SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Domain Adaptation by Revisiting Static Objects with a Transit Bus

Philip Neugebauer

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Domain Adaptation by Revisiting Static Objects with a Transit Bus

Domänenanpassung durch wiederholten Besuch Statischer Objekte durch einen Stadtbus

Author:	Philip Neugebauer
Supervisor:	Prof. Dr. Angela Dai
Advisor:	Dr. Christoph Mertz
Submission Date:	15.06.2023

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

P. Neugebauer

Munich, 15.06.2023 Philip Neugebauer

Acknowledgments

First and foremost, I want to express my sincere gratitude towards my advisor, Dr. Christoph Mertz and my supervisor, Professor Angela Dai, for the guidance and advice enabling the success of this thesis, and for their immediate support for any challenge that arose during the completion of this thesis.

I further want to thank my family for their strong support during my studies and thesis, supporting me by caring for me and providing advice and motivation. I also want to thank my friends for supporting me throughout the entire time of my thesis. Especially, it's hard for me to express how grateful I am for having the two colleagues, and friends, that I had in Christoph's research group: Anurag Ghosh and Anirudha Ramesh. Their advice both in work and life helped me to achieve a wonderful and successful time at the Carnegie Mellon University. I further want to thank my friends Oleksandr Golovnya, Min Ting Luong, Raoul Zebisch, Leonhard Winter, and Olivia Xu for their outstanding support.

I want to thank Navlab, the CMU Robotics Institute, and Dr. Christoph Mertz for hosting me for several months to do my research for this project. They generously provided me with the large BusEdge dataset. BusEdge is a project sponsored by the Mobility21 University Transportation Center (https://mobility21.cmu.edu/).

Lastly, I want to thank the German Academic Exchange Service (DAAD) for granting me a scholarship in the scope of the Program "IFI – Internationale Forschungsaufenthalte für Informatikerinnen und Informatiker - Programmlinie Masterstudierende".

Abstract

Modern Deep Learning-based Object Detection methods have made significant advancements in the past decade. However, a common limitation of these methods is their reliance on large amounts of labeled data. Especially for outdoor tasks, it is essential that the data covers multiple domains, such as varying weather and lighting conditions. However, while unlabeled data of these scenarios usually is readily available, labeled data is often scarce and might only cover some domains.

This thesis introduces a novel method that addresses this challenge by automatically labeling static objects in images under new domains. The approach leverages a sparse set of labels from one domain and unlabeled images from new domains captured in similar locations. The method works by locating a labeled object, revisiting the same object under a different domain, and transferring the label to the new domain. Evaluation of the proposed method utilizes data from the BusEdge system, a commuter bus equipped with cameras and sensors which captures data along its route. A potential use case of this bus is to automatically detect full trash cans that need to be emptied. An important conclusion of this thesis is that detection results can be improved by incorporating additional automatically generated labels from new domains.

Specifically, using the presented label generation method, labels of trash cans have been generated for new domains, such as cloudy or snowy weather. To enable a comprehensive analysis, two datasets under snowy conditions and a dataset under cloudy conditions were labeled by hand. These datasets complement an existing hand-labeled dataset in cloudy conditions, as well as a hand-labeled training dataset in sunny conditions. Training Object Detection models on a combination of hand-labeled data and automatically generated data yields remarkable improvements in detection results, compared to training on the hand-labeled data alone. The automatically generated data led to increases of up to +4.1 Average Precision (AP) and +6.7 AP on the two datasets representing the cloudy domain, and up to +1AP and +3.1AP on the datasets containing snowy conditions. The code for the developed method has been made publicly available.

Kurzfassung

Moderne Deep Learning basierte Objekterkennungsmethoden haben in den letzten zehn Jahren erhebliche Fortschritte gemacht. Eine häufige Einschränkung dieser Methoden ist jedoch, dass sie auf große Mengen an mit Labeln versehenen Daten angewiesen sind. Insbesondere für Aufgaben im Außenbereich ist es wichtig, dass die Daten mehrere Domänen abdecken, wie z. B. unterschiedliche Wetter- und Lichtbedingungen. Während jedoch Daten ohne Label für diese Szenarien in der Regel leicht verfügbar sind, sind Daten mit Label oft knapp und decken nur einige Bereiche ab.

In dieser Arbeit wird eine neuartige Methode vorgestellt, die sich dieser Herausforderung stellt, indem statische Objekte in Bildern in neuen Domänen automatisch mit Labeln versehen werden. Der Ansatz nutzt einen kleinen Satz gelabelter Daten aus einer Domäne und Bilder ohne Label aus neuen Domänen, die an den gleichen Orten aufgenommen wurden. Die Methode funktioniert, indem sie ein gelabeltes Objekt lokalisiert, dasselbe Objekt in einer anderen Domäne erneut betrachtet und die Label auf die neue Domäne überträgt. Die Bewertung der vorgeschlagenen Methode erfolgt anhand von Daten des BusEdge Systems, eines mit Kameras und Sensoren ausgestatteten Linienbusses, der auf seiner Strecke Daten aufzeichnet. Ein möglicher Anwendungsfall dieses Busses ist die automatische Erkennung von vollen Mülleimern, die geleert werden müssen. Eine wichtige Schlussfolgerung dieser Arbeit ist, dass die Objekterkennungsergebnisse durch die Einbeziehung zusätzlicher automatisch generierter Label aus neuen Domänen verbessert werden können.

Konkret wurden mit der vorgestellten Methode Label für Mülleimer in neuen Domänen automatisch generiert, wie z.B. unter bewölktem Wetter oder unter verschneiten Verhältnissen. Um eine umfassende Analyse zu ermöglichen, wurden zudem zwei Datensätze unter verschneiten Bedingungen und ein Datensatz unter bewölkten Bedingungen von Hand mit Labeln versehen. Diese Datensätze ergänzen einen bereits existierenden von Hand gelabelten Datatensatz unter bewölkten Bedingungen und einen von Hand gelabelten Trainingsdatensatz unter sonningen Bedingungen. Durch das Trainieren von Objekterkennungsmodellen mit einer Kombination aus von Hand gelabelten Daten und den automatisch generierten Daten konnten deutliche Verbesserungen erzielt werden, verglichen zu dem Training mit nur den von Hand gelabelten Daten. Die automatisch generierten Daten führten zu einer Steigerung der durchschnittlichen Präzision (AP) von bis zu +4,1 und +6,7 bei den beiden Datensätzen, die den bewölkten Bereich repräsentieren, und bis zu +1AP und +3.1AP bei den Datensätzen mit verschneiten Bedingungen. Der Code für die entwickelte Methode wurde öffentlich zugänglich gemacht.

Contents

1 Introduction

Deep Learning (DL) models have found huge popularity in recent years, particularly in fields such as Computer Vision, Natural Language Processing, Speech Recognition, and Recommender Systems. New models have been released at a fast pace, achieving remarkable advancements in their respective tasks. Deep Learning describes methods that aim to solve tasks without explicitly programming how to achieve these goals, but instead training a model to learn to achieve the task. It involves methods that utilize interconnected layers of mathematical functions which build a neural network to process data and make predictions, inspired by the workings of the human brain.

Multiple learning strategies exist in Deep Learning, including Unsupervised Learning for discovering patterns and clusters in data, Reinforcement Learning that utilizes feedback based on actions to learn, and Supervised Learning with labeled data. The choice of learning strategy depends on the task requirements, data availability, and other factors.

One of the major challenges in training DL models using Supervised Learning lies in obtaining sufficient training data. Many models achieve the optimal performance only when being provided with thousands of labels, which are typically annotated by hand. Large human labeled datasets exist, such as the COCO Dataset [1] with more than 200,000 labeled images and 80 classes, CIFAR-100 [2] with 60,000 labeled images and 100 classes, and ImageNet [3] with more than a million labeled training images spanning 1,000 object classes. However, although containing a huge number of classes, they still encompass only a fraction of the classes that might be needed for computer vision tasks.

To address this limitation, various approaches exist to reduce the reliance on a large number of labels for object detection. One such approach involves introducing new models that require fewer labels. Additionally, Data Augmentation techniques enable the expansion of the training set by performing label-preserving transformations on the existing data. Transfer Learning allows pre-training a model on different data, leveraging knowledge gained from one task or domain to enhance performance in a new task or domain. Domain Adaptation techniques enable the utilization of data from one domain to adapt and generalize to a different domain.

Moreover, training strategies like Active Learning and Semi-Supervised learning have been devised to minimize the number of labels that need to be manually annotated [4]. These strategies leverage algorithms that actively select the most informative samples for annotation or utilize a combination of labeled and unlabeled data to train the model effectively.

Despite these efforts, the demand for large amounts of training data remains in many cases. This ongoing requirement for extensive data continues to pose a significant challenge in training deep learning models.

This thesis presents a novel method for Domain Adaptation. Domains refer to distinct environments from which data is sourced, such as the location, weather condition, lighting condition, or perspective. In some instances, the available training data may be limited to just one or a few domains. For example, data for self driving cars might be limited to a small number of cities, but cars should be able to navigate in other places as well.

The absence of diverse training data poses difficulties when it comes to detecting objects that have not been encountered during training. For example, street signs in different regions may exhibit variations, or the training data might lack samples depicting objects in snowy conditions. Consequently, the task of accurately detecting such unseen objects becomes extremely challenging for neural networks, underlining the importance of Domain Adaptation. As a consequence, a large number of methods for Domain Adaptation have been introduced.

Data Augmentations play a crucial role in expanding the training data and exposing models to diverse scenarios, including scenarios from different domains. For instance, simulating different weather conditions allows models to learn robust representations capable of handling various environmental factors. Additionally, transferring image styles to new styles empowers models to adapt to different artistic renditions. Geometric transformations, such as rotations, translations, and adjustments of perspective, enrich the dataset with diverse spatial variations. Meanwhile, manipulating contrast and brightness levels adds flexibility to the model's response to variations in illumination.

Another prominent approach is Self-Training, where a model trained on one domain is applied to unsupervised data from another domain. Predictions made with high confidence by the model are utilized as pseudo-labels to retrain the model, gradually aligning it with the target domain [5].

Further approaches include leveraging tracking algorithms. The method by Tang et al. [6] tracks objects seen in images in videos to adapt from the image domain to the video domain. The method by Walsh et al. [7] works on objects that are static within a short time period and uses this assumption to generate further labels from different viewpoints and combines these to create 3D label suggestions which can speed up labeling efforts for new datasets.

This thesis focuses on addressing the specific problem of achieving Domain Adaptation for static objects. Static objects, such as buildings, landmarks, and road signs, play a crucial role in many computer vision applications, including autonomous driving, surveillance, and augmented reality. However, accurately detecting and recognizing static objects remains a challenging task due to changes in the domain, such as changing weather or lighting conditions or different perspectives. The method developed and applied in this thesis localizes the objects and the cameras to assign objects the same label while observing these objects under various conditions and perspectives. Using these generated labels, this thesis improves the accuracy and robustness of object detectors across domains.

To demonstrate the proposed method, data from the BusEdge system is used, a transit bus equipped with cameras and sensors. On its route, it captures images of trash cans, of which most are unlabeled, and the existing labels are only from the sunny and cloudy domain. Detecting objects on the route can be used to automatically detect overflowing trash cans, however, Object Detection models trained for this task are reliant on the provided labels.

Hence, this task is perfectly suited to show the potential of automatically generating data in the large number of new domains that the bus data covers and training an object detector on this data. This thesis demonstrates that high quality object labels can be automatically generated and used to achieve better results in detecting trash cans in domains in which no hand-labeled data exists.

1.1 Contribution

This thesis encompasses the following contributions:

- Development and implementation of a novel pipeline to automatically generate labels for Supervised Machine Learning based based Object Detection.
- Development, implementation, and execution of the pipeline on data recorded by a transit bus to generate waste bin labels for unseen weather and lighting conditions.
- Analytical and practical evaluation of the outcomes of the pipeline for waste bin label generation. For the evaluation, three test datasets were constructed and published with waste bin labels for the domains of snowy weather conditions at day and night, and for cloudy conditions based on the RoadBotics dataset[8].

1.2 Outline

The thesis is structured into the following parts:

- **Related Works**: Chapter 2 introduces works that aim at solving challenges similar to this thesis and illustrates how the presented work fits into the field of Domain Adaptation. Furthermore, it presents works that can be potentially used to implement parts of the proposed pipeline, and discusses their advantages and disadvantages.
- **Implementation**: Chapter 3 presents our proposed pipeline in detail and demonstrates our implementation of it.
- **Evaluation**: Chapter 4 showcases and analyzes the outcomes of our thesis, encompassing both theoretical examination and practical implementation for training DL models.
- **Discussion**: Chapter 5 analyzes and discusses the results of the evaluation, with a focus on which parts of the pipeline influence the number of labels that are generated.
- **Conclusion**: Chapter 6 summarizes the outcomes and insights of this thesis and presents potential improvements for the introduced method, based on the learnings from the evaluation and discussion.

2 Related Works

In this section we will cover works that contain similar approaches to the presented idea. In addition, we will give an exhaustive overview of recent works that can be used for parts of the implementation.

2.1 Deep Learning

Deep Learning is a subarea of Machine Learning, which in turn is a subarea of Artificial Intelligence. Artificial Intelligence describes any algorithm that mimics the behavior of a human, such as playing a game, detecting an object in an image, or replying to text. Machine Learning describes algorithms that work by using a set of input data on which the algorithm learns an objective, which could be finding clusters in the data, or predicting outputs for each datapoint that match outputs given as learning input, so called labels. A label can be sentiment of a text, information what an image depicts, or the location of an object in an image, to name a few examples. In the scope of this thesis, a label refers to the bounding box of an object in an image. Machine Learing includes algorithms that can detect linear correlations in data, such as state-vector machines or linear regression. Deep Learning combines many linear layers connected with non-linear activation functions, which allows it to detect nonlinear correlations in data. This allows DL to be used for use cases such as Natural Language Processing, Image Classification or Object Detection.

In recent years, Deep Learning has seen a surge in popularity which led to many novel DL models being released. However, these models generally achieve the best results on huge datasets that contain thousands of labels. While approaches such as One-Shot Learning or Few-Shot Learning exist, for example by Yu et al. [9], they are generally not suited for all use cases, but rather for specific use cases like face recognition.

The paper "A survey on deep learning and its applications" [10] grants further insights into the state and applications of Deep Learning.

2.2 Object Detection

Object Detection is one of the most researched areas of Computer Vision, next to Image Classification and Image Segmentation. It describes methods that detect the location and type of objects in images, whereas objects can be any category of entities that can be observed, such as physical tools, animals, or humans. Object Detection is used in fields such as Autonomous Vehicles, Medical Imaging, Environmental Monitoring, and Robotics.

Early Object Detection methods typically focused on handcrafted features. Examples include the Viola-Jones Algorithm [11] or Histogram of Oriented Gradients [12]. These algorithms require significant development times for each class that is to be detected and newer Deep Learning based methods significantly outperform these methods if provided with enough labels. DL based Object Detection methods can be generally divided into One-Stage Detectors, such as YOLO [13], Two-Stage Detectors, such as Faster R-CNN [14], and Transformer based Object Detectors, such as DeTr [15].

2.2.1 YOLO

"You Only Look Once: Unified, Real-Time Object Detection" has been published by Redmon et al. in 2015 [13] and outperformed other State of the Art (SOTA) methods. Furthermore, as it creates bounding boxes for objects, and regresses and classifies these in only one step, it achieves real-time performance. Further versions have been released since then, the latest being YOLOv8 [16] which outperforms previous models in terms of performance and accuracy.

2.2.2 Two-Stage Detectors

One of the most prominent Two-Stage Detectors is Faster R-CNN by Ren et al. Presented in 2015, it achieved State of the Art performance at almost real-time, and builds on its predecessors R-CNN [17] and Fast R-CNN [18]. Two-Stage detectors consist of a first stage in which they generate Regions of Interest (RoIs) that are likely to contain objects, and a second stage that performs Object Classification and bounding box regression on the proposed RoIs. In the case of Faster R-CNN, the first stage is accomplished by a fully convolutional Region Proposal Network (RPN) and the second stage uses a RoI pooling operation and Fully Connected Layers (Convs) to output class scores for each region and regresses their bounding boxes. The RPN produces feature maps that get reduced in scale at each layer but increase the semantic value. The layer with the lowest resolution and highest semantic score is then used to predict Regions of Interest.

Multiple convolutional networks can be used for the RPN, so called backbones. A common choice for the backbone is ResNet [19], as it contains skip layers which allow to build deep networks. Other choices include VGG [20] and MobileNet [21], which is famous for its fast inference time that makes it fitting for mobile applications. Different versions of ResNet exist, such as ResNet-50 and Resnet-101, whereas the number determines the number of layers that the network contains. These versions grant different trade-offs between accuracy and performance, as more layers allow it to achieve better accuracy at the cost of slower inference times.

Many works build on top of Faster R-CNN. Mask R-CNN by He et al. [22] introduces a further head that outputs a segmentation mask for each object with little overhead. Furthermore, it introduces RoIAlign to replace the RoI pooling operation of Faster R-CNN, to remove the harsh quantization of RoI pooling. Mask R-CNN does not only allow to perform tasks that require a segmentation mask, but even improves the results for object detection in some cases.

Feature Pyramid Networks (FPNs) by Lin et al. [23] can be combined with Faster R-CNN to improve object detection at varying scales, especially for small objects. FPN works by using not only the lowest layer produced by the RPN. Instead, it introduces upscaling based on the semantic features and skip layers between similar sized layers to generate predictions at varying scales.

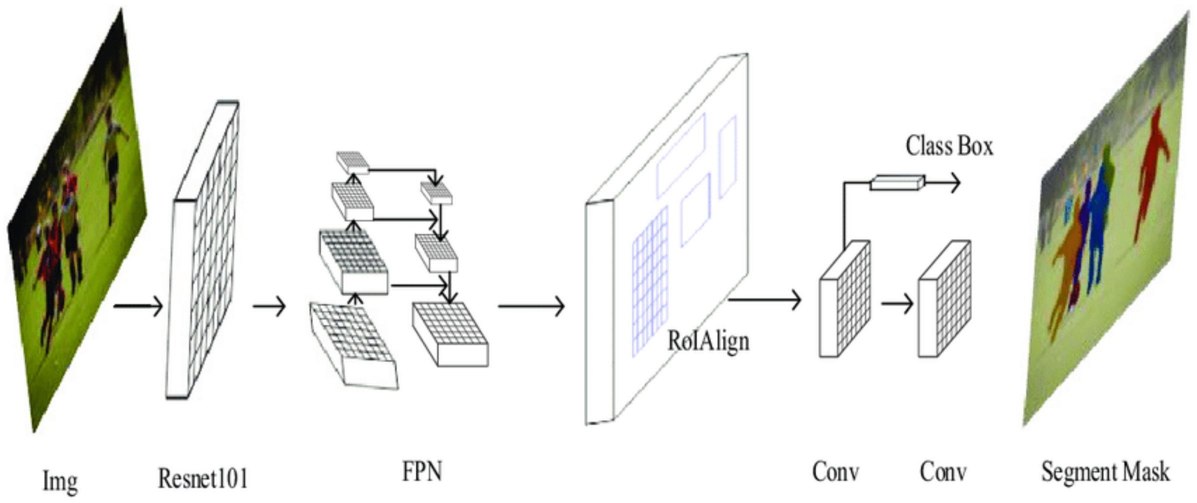


Figure 2.1: Overview of the Mask R-CNN + FPN architecture with a ResNet-101 backbone. Source: [24].

Good performance and popularity in the computer vision community make Mask R-CNN with ResNet + FPN backbone, as illustrated in Figure 2.1, a good choice to use it to benchmark our label generation method and show whether the generated labels can have a positive impact on object detection performance. As the focus of our method is on Object Detection, only the class boxes of the result are used, and not the segmentation mask.

2.2.3 Transformer-Based Object Detectors

Transformer Models have been introduced in 2017 by Vaswani et al. [25]. They found huge popularity in NLP applications, but also show good performance on Computer Vision tasks. In 2020, Carion et al. introduced the Detection Transformer (DeTr) [15]. It consists of a Transformer on top of a Convolutional Neural Network (CNN) and achieved State of the Art results. In 2021, Liu et al. introduced Swin [26], which again set new SOTA results on the ImageNet-1k [3] and COCO [1] datasets, by utilizing shifting windows in combination with a Transformer network. In the following years, newer versions of these models have been introduced, such as Co-DETR [27], Group DETR v2 [28], and FD-SwinV2-G [29] which improve the performance of the models.

Transformer based models have also been introduced for other Computer Vision tasks, such as Mask2Former [30] which improved SOTA results for panoptic segmentation, instance segmentation and semantic segmentation on the COCO dataset in 2022.

Despite the good performance of many Transformer based models, we chose Mask R-CNN to evaluate our results, as most other methods for Domain Adaptation also use Mask R-CNN, and it is therefore easier to compare results.

2.2.4 Further Object Detectors

In this thesis, we cover only the most prominent object detectors. Zaidi et el. [31] give a more holistic review of the current State of the Art detectors as of 2022. Zou et al. [32] provide a more extensive survey of the development of Object Detection models and methods related to Object Detection.

2.2.5 Detectron2

Detectron2 [33] is an Open Source Computer Vision Framework developed by Facebook AI Research (FAIR). Key features of Detectron2 include:

- **Extensibility**: Detectron2's modular design makes it easy to create costume networks, for example, by providing backbone networks, feature extractors, RPNs, and more. While it provides a number of data augmentations, loss functions, and evaluation metrics, it is simple to use custom augmentations, loss functions, or metrics.
- **State of the Art Models**: Detectron2's Model Zoo provides a collection of pre-trained SOTA models for various Computer Vision tasks, such as Mask R-CNN. Using these models, it is simple to use existing or custom datasets to train and benchmark models, for example, on the COCO dataset [1] that is widely used for object detection benchmarking.
- **Utility**: Detectron2 provides built-in techniques that allow to speed up training and inference times, for example, Automatic Mixed Precision significantly speeds up training while reducing memory footprint.
- **Scalability**: Detetectron2 provides a number of tools to allow large scale model training and is designed to scale well with large datasets. One feature it provides to achieve this is multi-node multi-process distributed training.

Detectron2 is widely adopted in research projects, benchmarking, and industry. Ease of use and importance in the field of object detection were deciding factors to choose Detectron2 to benchmark training effectiveness with and without data generated by our presented method.

2.3 Domain Adaptation

In many Deep Learning applications, training labels are available for one or few domains called the source domain. Meanwhile, the system is needed to work on other domains as well, for which no, few, or only weak labels exist, the so called target domain. Models trained on this data generally perform significantly worse in other domains than the ones for which many training labels exist. As a result, many methods exist that attempt to bridge the gap from one domain to another one. Depending on the availability of labels for other domains, different groups of methods for Domain Adaptation exist. According to Oza et al. [34], the following groups exist:

- If there exist a small number of labels for the new domains, Domain Adaptation via Semi-Supervised learning is possible, with algorithms such as self-training [5], co-training [35], or generative models [36].
- If there exist weak labels, i.e., noisy labels, incomplete data, or proxy or surrogate labels, Weakly Supervised learning can be used. An example for incomplete data for object detection would be that only classification labels exist for an image, instead of annotations which objects are contained and what their bounding boxes are. An example for proxy data would be using meta data.
- If there exist no labels at all for the target domain, Unsupervised Learning methods can be utilized.

This thesis assumes the case that there exist no labels for the target domain, and can be therefore categorized as part of Unsupervised Learning. While the generated labels can be seen as weak labels, Weakly Supervised learning could work on top of our method, rather than describing the functionality of our method. The following section therefore grants an overview over Unsupervised Domain Adaptation. The paper "A survey on semi-supervised learning" [37] grants an overview of Semi-Supervised Learning, while the paper "Weakly Supervised Object Localization and Detection: A Survey" [38] gives an overview of Weakly Supervised Learning.

2.3.1 Unsupervised Domain Adaptation

According to Oza et al. [34], Unsupervised Domain Adaptation can be divided in the following subcategories:

Domain Invariant Feature Learning

These strategies work by training the network in a way that it uses features that are less dependant on the source domain for detection. This is achieved by translating both data from the source and target domain into the feature space. Features that are common in both spaces are then prioritized over features that mainly occur only in the source space. A practical example could be to prioritize shape features over texture features when adapting from sunny weather conditions to cloudy conditions, as this change would presumably affect color more than shape of objects.

Pseudo-Label based Self-Training

These strategies work by training the model on the source data and apply it on the unlabeled target data to generate predictions. These predictions can be used as weak labels for re-training the network. This process can be repeated to generate further pseudo-labels and re-train the model on the growing dataset. As the generated labels can be noisy and can contain mistakes, filtering methods for the labels are usually used, for example, only the predictions with the highest probabilities get used. Furthermore, these methods are often combined with methods that use assumptions, such as methods that assume temporal consistency and use tracking to generate better labels. Finally, methods such as loss re-weighing can be used, for example, to weigh the pseudo-labels less, as they could contain wrong labels. Notable works of *Pseudo-Label based Self-Training* are presented in Section 2.3.2.

Image-to-Image Translation

These strategies work by translating images from the source domain to the target domain to train the model on the target domain. To achieve this, typically Generative Adversarial Networkss (GANs) or Variational Autoencoders are used. GANs consist of a Generator and a Discriminator model. The Generator attempts to generate data from the target domain based on the source domain that the Discriminator cannot differentiate from real data, while the Discriminator attempts to determine whether input samples are generated samples or real samples. Both models are trained in parallel based on the results with their respective tasks. A practical example is Domain Adaptation from day to night images. The Generator takes daytime images as input and generate similar looking images at nighttime, while the Discriminator receives these generated samples as well as with real nighttime data and would then predict whether a sample is real or generated. Finally, the trained Generator is used to transfer labeled images from the source to the target domain, while keeping the labeled object in the image. Using these generated samples with labels, the Object Detection model can be trained on source domain data and generated target domain data.

Domain Randomization

These strategies work in a similar way to *Image-to-Image Translation*, however, instead of adapting to one specific target domain, these methods aim to adapt to new domains in general. *Domain Randomization* works by introducing perturbations to the training data of the source domain, such as changing visual appearance, lighting conditions, object poses, textures, backgrounds, or camera viewpoints. This can be achieved, for example, by a style transfer network, such as the one presented by Huang et al. [39]. An example for this strategy is to generate new data under various lighting conditions based on the training data to achieve Domain Adaptation from daytime to nighttime. A model is then trained on the source data and all generated data. The generated data which could include lighting from dusk or dawn could then potentially improve results for night time detection, as it can allow the model to better understand lighting changes in general.

Mean Teacher Training

This strategy uses a student model that learns on a new domain and teacher model that attempts to prevent the student model from learning incorrect assumptions, using a consistency loss between predictions of both models. The general process for *Mean Teacher Training* is:

1. **Initialization**: The student model is randomly inititalized or trained using Supervised Learning. As the teacher is updated based on the student model using Exponential Moving Average (EMA) on the student's parameters, it is set to the student model at initialization.
2. **Exploration**: The student model learns on the target domain. This can be achieved using the previous methods, for example, by perturbing the source data to reflect the target domain.
3. **Consistency Check**: Predictions of both the student model and the teacher on unlabeled source and target data are compared to each other, the consistency loss describes how far these predictions diverge from each other.
4. **Update**: The student model is updated using both the supervised loss as well as the consistency loss. The consistency loss reduces the students sensitivity to input variations and therefore helps to stabilize the learning process by smoothing out the fluctuations and noise in the student model's parameter updates. The teacher model is updated by using EMA, meaning that it combines the newly learned parameters of the student model with the previous parameters of the student model.
5. **Iterative Training**: Steps 2. - 4. are repeated for multiple iterations, gradually refining the student and teacher models. While the student model adapts gradually to the target domain, the teacher model enforces consistency and guides the student model.

An example how this technique might improve stability for Domain Adaptation from sunny to snowy weather conditions is, that the student might start associating objects with a predominantly white color with snowy weather due to the visual patterns observed in the target domain. This assumption may be incorrect as objects in the source domain may also have a white color, but may not necessarily be associated with snowy weather. The teacher has not yet learned this assumption, and would therefore not make any mistakes based on this assumption. The consistency loss then helps the student model to correct this assumption.

Graph Reasoning

Techniques for *Graph Reasoning* utilize graph networks that can capture domain specific and domain invariant features. The method of Cai et al. [40] for Object Detection uses anchor boxes as nodes for the graph, the value of the node being the probability vector denoting the probability of that region belonging to one of the categories. The graph is then optimized using three losses, while utilizing *Mean Teacher Training*:

1. **Region-level consistency** helps to align predictions between teacher and student on a regional level. This can be achieved by aligning the features or bounding box predictions of similar object proposals across the domains.
2. **Inter-graph consistency** focuses on capturing consistency between the source and target domains at a higher-level representation. It aims to align the global characteristics across the domains, such as object category distribution or semantic patterns.
3. **Intra-graph consistency** emphasizes consistency within each domain in the student graph. This ensures that similar object proposals within the same domain exhibit consistent features or predictions.

Using these consistency losses, combined with student-teacher training shows improved results for Domain Adaptation.

Many of these techniques can also be combined with each other to increase their effectiveness. For example, *Mean Teacher with Object Relations* [40] combines *Graph Reasoning* with *Mean Teacher Training*, while still being mutually exclusive to the other techniques.

According to this categorization, our method fits best into the category *Pseudo-label based self-training*, as it generates pseudo-labels and then uses these labels to train and adapt to new domains.

2.3.2 Pseudo-Label based Self-Training

The paper "Shifting Weights: Adapting Object Detectors from Image to Video" [6] by Tang et al. describes a method that automatically generates labels for object detection in videos. The method works by using a detector trained on 2D images and detecting trajectories of objects with it in videos. These trajectories can be used to weakly label the object in each frame and these new labels can then be used to further train the object detector.

A similar approach is used to adapt models to new domains by RoyChowdhury et al. in the paper "Automatic Adaptation of Object Detectors to New Domains Using Self-Training" [5]. Their method applies a trained model on a dataset that contains no labels and uses the predictions as weak labels to re-train the model. Furthermore, their method uses estimated trajectories to generate further labels of objects that were missed by the detector. The generated labels can contain various weather conditions, light conditions, as well as view points, leading to a better domain adaption of the model on a domain for which only an unlabeled dataset exists. As the generated labels can be potentially incorrect, RoyChowdhury et al. also show how using a knowledge distillation loss, i.e., reducing the weight of these generated labels in training, can improve performance of the model. This approach especially helps to combine both hard and easy examples.

2.3.3 Dehazing

In addition to general Domain Adaptation methods, methods that adapt a model to one specific new domain promise better results for this domain. For example, various methods exist that tackle the challenge of dehazing, for which information about this specific problem can enhance adaptation.

Foggy weather conditions can pose a challenge to object detection models, as the haze reduces visibility and contrast in images. Furthermore, the haze depends on the per-pixel distance to the camera, light sources, and other influences, making it non-uniform and therefore non-trivial to remove. Various methods to tackle this challenge exist:

- "“Double-DIP”: Unsupervised Image Decomposition via Coupled Deep-Image-Priors" by Gandelsman et el. [41] allows to remove haze from images by dividing the image into multiple layers, for example, into a layer containing the haze and one that contains the image without haze.
- "Non-Local Image Dehazing" by Berman et al. [42] assumes that scenes mainly contain only a few hundred distinct colors. For example, two grass fields might have the same color, therefore any observed difference might come from a difference of distance to the camera which translates to different transmission coefficients. Therefore, the color difference between pixels can be used to determine the distance and transmission coefficients of each pixel.
- "Single image dehazing" by Fattal [43] formulates a refined image formation model, that defines how surface shading and light transmission influence how the image is captured. Using this model, the haze layer can be removed and a reliable transmission estimate can be created.
- "Enhanced Pix2pix Dehazing Network" by Qu et al. [44] introduces a GAN and an Enhancer model which are jointly trained. The Generator of the GAN creates a pseudo realistic image at a coarse level of detail, guided by the Discriminator. The Enhancer incorporates two enhancing blocks inspired by the receptive field model to refine the generated image and reduce haze in the image.

While these methods show good results for the task of image dehazing, their application is limited to this domain and does not allow for general Domain Adaptation.

2.4 Data Acquisition

This section presents two of the most common data acquisition methods, how they work and their advantages and limitations.

2.4.1 Manual Labeling Methods

Traditional manual labeling works by humans performing a labeling task for all data to create the needed labels. The tasks can be classifying an image, drawing bounding boxes around objects in images, or drawing segmentation masks for objects in images, among others. Having multiple annotators label the same objects can improve the accuracy of the labels and generally achieves a high accuracy for the tasks. However, it comes at the cost of extensive human effort, which can be time consuming and costly.

To ease the effort in labeling, numerous methods have been presented to make the labeling process faster, for example, by allowing to generate 3D labels with only one click in the center [45], or by generating high recall labels [7] which human annotators can either accept or reject, which can significantly reduce labeling time. However, despite these methods, labeling by hand often still requires significant human effort, as datasets can contain thousands of labels.

2.4.2 Synthetic Data

It is generally possible to imitate the data, for example, by using pre-existing models or by creating 3D models of the objects for which labels are needed. These virtual models allow to generate data with any viewpoint, simulated weather condition, lighting, background, and other settings, which makes them especially suitable for domain adaptation. This can especially help with the task of Domain Adaptation. In addition, this approach is especially useful when the data is difficult to obtain or good simulations already exist. However, synthetic data also introduces a new domain gap between the simulation and reality. Using additional methods to improve Domain Adaptation can help mitigate this issue and improve the accuracy of models trained on synthetic data.

A field in which synthetic data is useful is autonomous driving, where there are many scenarios that only occur infrequently, making it challenging to collect real-life samples. In these cases, synthetic data generated from specialized simulators like CARLA [46] and SYNTHIA [47], or even data from video games like GTA 5 [48], can be used. Alternatively, there are approaches that do not require a simulator, such as "Cut, Paste and Learn" [49], which cuts out labeled objects and pastes them into different backgrounds to generate more training data. However, using a simulator for synthetic data generation has been shown to produce better results [50] than the *Cut, Paste and Learn* approach. However, although simulations are getting increasingly realistic, modeling the real world is a hard task, for example, as there are many phenomena that can only be approximated due to computational constrains and many scenarios that only rarely occur.

2.5 Learning Strategies

Some learning strategies present solutions to reduce the number of labels needed for training. This section explains some of the most prominent strategies, their advantages and limitations.

2.5.1 Active Learning

Active Learning (AL) is a method for reducing the amount of time required for humans to manually label datasets by iteratively improving the model's predictions through a process of label suggestion and human verification. In the beginning, humans generate a small set of labels, which are used to train a model. This initial training allows the model to make predictions, which are then checked by a human, who categorizes the labels as correct or wrong. Using the newly generated labels from this step, the model is retrained to improve future predictions. By repeating this process of training the model and checking its predictions, more and better labels can be generated over time. Additionally, the model determines which images provide the highest training value if labeled. This leads to good results with reduced human labeling effort.

There are some challenges when using AL, one of which is the cold start problem. In the beginning, the model has only few reference points for training, leading to a high false positive rate in the predictions, which in turn leads to a lot of negative samples and few positive samples. This means that it might take many iterations until the model is able to achieve good results. Depending on the model size, retraining the model can also require some time, in which the human in the loop is not able to check predictions. This could lead to some extra time requirements for humans, or could require humans to be available at different times. Finally, the model might not learn well how to detect all kinds of object variations, as the model might never predict any samples that look vastly different from what is contained in the initial training data. For a more comprehensive overview of AL, the paper „A Survey of Deep Active Learning“ [51] provides further information.

2.5.2 One Shot Learning

One Shot Learning and similar approaches like Zero Shot Learning or Few Shot Learning have a similar goal to this project of making the most out of a sparse number of labels. The main idea behind some One Shot Learning methods is to utilize existing labeled data that is similar to the data that should be learned with only one image. An example is to identify people with only one image for training, by learning features on a dataset of other humans. For example, a model might learn important features that help to differentiate people, such as facial landmarks, and then uses these features to identify a new face. The main drawback of One Shot Learning is, that a dataset that is very similar to the new sample is required. In general, such a dataset is not present. Kadam et al. [52] give a a comprehensive review of Zero, One, and Few Shot Learning Approaches.

2.5.3 Transfer Learning

Transfer Learning has a similar approach to some One Shot Learning methods, however, it generally assumes that the new target data has more than one image, and the source data and target data can be further away from each other. Transfer learning learns general low level features on a big dataset. These low level features, while generally not having any semantic meaning to them, can be imagined as components of objects, such as edges, circles, wheels, etc. Using a pre-trained model that learned these features, it is shown that models can adapt faster to detect new objects. While low-level features are generally not comprehensible for humans, an easy way to imagine this training is, that to learn any unseen object such as a scooter, the model would use the existing learned features. For the sample of the scooter, it would learn that it's "two wheels connected by a horizontal edge, with a second vertical edge attached to one of the wheels", instead of starting to learn how to detect edges and wheels.

While Transfer Learning can greatly improve accuracy and speed up training, it requires the existence of a huge dataset for pre-training that is similar to the target data, and the training can take tremendous amounts of time. While there exist such pre-trained models readily available, these models have been trained only on few different datasets and might therefore not be well suited for all tasks. Furthermore, a small dataset with target data is required, that might not always be available.

In this thesis, we choose to use Transfer Learning, due to the simplicity of the strategy, the availability of models that have been pre-trained on street level data, and as these models perform well on the given task. A further overview over the state of Transfer Learning is given by Zhuang et al. [53].

2.6 Similar Approaches

In this section, methods that have a different goal than this thesis, but have a similar approach, are presented. As these methods contain similar components, they can be insightful for this thesis.

2.6.1 Automatic Label Generation

The method presented in the paper "Leveraging Temporal Data for Automatic Labeling of Static Vehicles" by Walsh et al. [7] contains many similarities to the method presented in this thesis. The paper describes how it is possible to generate new labels by localizing an object in the 3D space in a sequence of images, using a 3D detector that has been pre-trained on a labeled dataset. By localizing objects, the method is designed to improve the proposals generated by the detector to generate high recall labels. These suggestions can help human annotators to label a new dataset more efficiently. This thesis extends this approach to also work across different sequences and different seasons.

2.6.2 Scene Change Detection

Scene change detection has a similar approach to this work, as it also compares the same locations over time. Therefore, many steps of localization of the camera and scene geometry are similar, as well handling different image viewpoints. However, an annotated dataset for changes that can occur is usually used for change detection methods, making them unsuitable for label generation.

The paper "Street-view change detection with deconvolutional networks" by Alcantarilla et al. [54] describes a full pipeline for scene change detection. The pipeline consists of localizing the camera and densely mapping the scene, as well as reprojecting new camera images to previously seen camera poses using the dense reconstruction and performing change detection on the 2D images. It presents and compares two different algorithms for the change detection, named *CDNet* and *FCN-CD* that are both based on deconvolutions. The output is a segmentation mask for all changes that occur in the scene.

Alcantrilla et al. present and use the VL-CMU-CD dataset that they created by labeling changes in the VL-CMU dataset [55][56]. It contains 152 RGB and depth image sequences for change detection and 1,362 registered image pairs with manually annotated structural change segmentation masks. Most annotated changes belong to the classes bins, signs/traffic-signs, and vehicles.

For localization of the camera poses and creating a sparse reconstruction of the scenes, multi-sensor fusion SLAM is used. The presented method combines 2D image feature measurements, GPS measurements per camera, and odometry data, each with an individual factor and an assumed underlying Gaussian noise distribution.

New sequences can be registered with the existing reconstruction across different times of day and seasons by extracting and tracking A-KAZE features [57] and then refining the registration by matching these features using Random sampling and consensus (RANSAC) and a three-point algorithm [58]. Afterwards, global Bundle Adjustment is performed to bring the camera poses and sparse 3D reconstruction into a common reference frame.

Alcantrilla et al. build a dense 3D reconstruction by first computing DAISY descriptors [59] and SEEDS superpixels [60] and then estimating a plane for each superpixel using RANSAC on which they perform optimization. The results get refined using the method described in "Fast Global Image Smoothing Based on Weighted Least Squares" by Min et al. [61]. This process is repeated on different scales with the previous results being propagated to the next level.

Using the 3D reconstruction and localization, 2D images can be reprojected to new poses. This allows to generate image pairs that are from exactly the same pose. On these image pairs, change detection can be performed in 2D space, for which the paper presents two methods *CDNet* and *FCN-CD*.

CDNet [54] is a Convolutional Neural Network based on Deconvolutional Networks with the main idea to stack contraction and expansion blocks with 1.4 million parameters. It has a good trade-off between size and performance and its small size makes it suitable for mobile applications.

FCN-CD [54] is a CNN based on Deconvolutional Networks as well, especially based on Fully Convolutional Networks [62]. It achieves higher precision than *CDNet*. However, it has 134.5 million parameters, making it almost 10 times larger than the *CDNet* and therefore it is more prone to overfitting and requires more memory.

2.7 Localization and Scene Reconstruction

An important part of this thesis is localizing the cameras of the images from the source and target distributions and to create a reconstruction of the scenes surrounding the objects for which labels are to be classified. In this section, we explore methods that are suitable for this task and discuss their benefits and drawbacks.

2.7.1 Localization

A number of methods exist that focus on localization [63, 64]. Ding et al. [63] shows that it is advantageous to first build a map using additional Lidar data in order to subsequently perform localization using camera only. However, to keep the presented method simple, only one algorithm will be used to compute both the location of the vehicle and to create a reconstruction.

2.7.2 SLAM

Simulatanous Localization and Mapping (SLAM) is a technique that uses sensor data to both localize the position of the sensor suite and map the surrounding area. It is usually used in robotics and autonomous driving, as most SLAM methods are designed to run in realtime. Most implementations require ordered camera streams. There are different subcategories for SLAM, depending on the sensors they require. While many implementations utilize Lidar sensor data, Visual SLAM only requires cameras and Visual-Inertial SLAM uses cameras and odometry data. In addition, there are approaches to combine GPS with SLAM, such as [65]. While utilizing Lidar can improve results in many cases, many setups do not contain one, as current Lidar systems are expensive. As a consequence, we aim to build a method that does not require a Lidar sensor. Consequently, the following sections focus on presenting methods that do not require a Lidar.

2.7.3 Visual and Visual Inertial SLAM

The survey "Visual and Visual-Inertial SLAM: State of the Art, Classification, and Experimental Benchmarking" by Servières et al. [66] grants a recent overview from 2021 over the SOTA methods for Visual and Visual Inertial SLAM. It compares the methods Vins-Mono [67], ROVIO [68], ORB-SLAM2 [69], DSO [70], and LSD-SLAM [71] and performs benchmarks for these methods on the EuRoC MAV dataset and on a visual-inertial dataset that they created specifically for urban pedestrian navigation.

Among the tested methods, the ORB-SLAM2 algorithm achieves the best results in most categories. This makes it particularly promising for the method presented in this thesis, which aims to map surrounding areas of labeled objects and localize the camera. ORB-SLAM2 exhibits several desirable properties in this context:

- ORB-SLAM2 demonstrates the robustness among the tested methods on the IRSTV dataset, showing no issues related to the environment or initialization.
- It effectively handles challenges encountered in urban spaces, such as glass reflections and pedestrian motion.
- It has the ability to detect and close loops. This is essential when revisiting the locations of the labels.

- Even in scenarios without loops, ORB-SLAM2 yields precise results, with a low Absolute Positioning Error (APE) Root Mean Square Error (RMSE) of 1.1% over the traveled distance.
- ORB-SLAM2 excels in trajectory reconstruction, which aligns well with our use case, as live pose estimation is not required.

The alternative methods have shown drawbacks in the benchmarks by Servières et al. [66], that make them less preferable for the given use case:

- **Vins-Mono** achieves good results, but ORB-SLAM2 generally surpasses its performance. Moreover, initialization issues may arise if additional methods, such as the one proposed by Fu et al. [72], are not utilized. While Hu et al. [73] demonstrates improved results for Vins-Mono by optimizing the photometric parameters, it remains unclear how Vins-Mono compares to other SLAM methods with these adaptations. Therefore, we will rely on more established methods.
- **DSO**'s native implementation lacks a loop closure method and necessitates GPU acceleration. Although a method to include loop closure has been proposed by Gao et al. [74], DSO is better suited for live pose estimation rather than trajectory reconstruction, making it less compatible with our offline approach.
- **LSD-SLAM** heavily relies on initialization, which poses a significant challenge in practical scenarios.
- **ROVIO** performs well for applications focusing on local pose estimation. However, it lacks high accuracy in reconstructing trajectories, which is a requirement for our method. Additionally, textureless areas can pose problems for ROVIO.

Yin et al. proposed a novel SLAM method called BioSLAM [75], which utilizes a dual-memory mechanism inspired by the human brain's memory replay mechanism. The dual-memory mechanism consists of a dynamic part for learning new observations and a static part for managing new and old memories. The introduced method works on visual and Lidar data. The authors show that BioSLAM outperforms SOTA place recognition methods in a number of test cases for the visual and Lidar-based implementation. However, this method is relatively new and our test data does not include Lidar data. Therefore, we will focus on more established methods and leave the exploration of BioSLAM's potential to future research.

Several papers show how SLAM methods can be improved by including further data:

- **Multi-IMU**: Zhang et al. show in [76] that using multiple Inertial Measurement Units (IMUs) improves localization by noticeable margins without adding a lot of computational cost. However, not all systems, contain multiple IMUs.
- **Multi-session Maps**: Bürki et al. show in [77] that it is possible to use multi-session maps to improve localization accuracy.

- **Multi-Agent**: There are several approaches that have demonstrated the effectiveness of combining data from multiple concurrent agents to improve positioning accuracy, such as [78], [79], and [80].

However, such data might not always be available and these methods might add additional complexity to the proposed method.

2.7.4 SLAM Frameworks

Frameworks for SLAM can help to integrate a SLAM method and methods that work on top of it. The framework Maplab by Schneider et al. [81] is a popular framework that grants a lot of flexibility and provides many methods that are already implemented. The methods include different implementations for loop closures, incorporating GPS and other priors, dense reconstruction, and combining maps. In addition, Maplab is able to handle large-scale environments and lifelong mapping by using map sparsification.

2.7.5 Structure for Motion

Structure for Motion (SfM) has the same goal as Visual SLAM to localize the camera positions and reconstruct the surrounding area. However, usually different assumptions are made for SLAM and for SfM, as both focus on different use cases. SLAM is mainly used for Autonomous Navigation, therefore many methods consider resource constraints and assume a logic behind the movement of the vehicle. In comparison, a typical use case for SfM is to reconstruct a famous architecture based on crowd-sourced images. As a consequence, most SfM implementations work on unordered images and only some methods consider resource constraints.

Structure for motion usually works by performing the following steps:

1. **Feature Extraction**: In the first step, features are detected in each image which are points or lines with a well-defined position in image space, e.g. at corners or edges. Famous examples of feature detectors include SIFT [82], SURF [83], A-KAZE [57], and ORB [84]. Differences between detectors lie in how features are chosen. The feature points are then used to obtain descriptors. Descriptors are feature vectors that contain information about the local neighbourhoods of the most representative features and therefore form a compact representation of the image that removes most of the redundant data while keeping most of the interesting data.

2. **Feature Matching**: In the next step, the descriptors of each image get compared to find matches. If images have a lot of matches, it means that they are similar and likely means that they are showing the same objects that can be used as reference points to localize the camera positions. The method RANSAC [85] is often used for finding the commonality between two sets of feature points and for outlier rejection, due to its high robustness.

3. **Computing the Fundamental, Essential and Projection Matrices**: Using the computed matched features for an image pair, the fundamental matrix can be calculated. The fundamental matrix describes the relationship between two images that contain points of the same scene. It can be used to compute the essential matrix. The essential matrix captures the geometric relationship between the camera positions where the two images were captured. Finally, it can be used to compute the projection matrix. The projection matrix allows to transform a point in the 2D image space to a ray in the 3D world space on which the point lies.

4. **2 View 3D Reconstruction**: The projection matrix of two images can be used to get an estimation for the 3D position of the matched points. Triangulation can be used to further refine these estimates. One limitation of this reconstruction is, that there is an ambiguity for the scale of the reconstruction. Without the usage of any other sensors or knowledge, it is impossible to calculate the real size of the reconstruction.

5. **Multi-View Reconstruction**: There are two main approaches for registering more than two cameras, known as incremental and global Structure from Motion methods. Incremental SfM methods register new views to previous views one at a time, using the same algorithm as for two views, which can lead to higher projection errors for individual images but may be less influenced by those errors overall. In contrast, global SfM methods register all images simultaneously, making them more susceptible to outliers but generally more scalable and efficient.

6. **Sparse Bundle Adjustment**: Bundle Adjustment (BA) is a technique that can be used to refine the localization and reconstruction results. In the case of incremental SfM, BA is applied after each increment, while for global SfM, it is only applied once at the end of the reconstruction process. Bundle Adjustment works by optimizing the reconstruction error for all reconstructed 3D points. As in general the points do not perfectly line up when seen in different images, this equation is non-linear. The involved parameters - camera intrinsics, poses, and 3D point positions - are generally optimized iteratively.

2.7.6 COLMAP

COLMAP [86][87] is an incremental SfM and Multi-View Stereo pipeline. While it can be used out of the box to localize the camera positions and create a sparse or dense reconstruction, it allows to exchange different parts of the pipeline, for example, by importing features from any feature detector. In the base configuration, it uses the SIFT [82] feature extractor and offers different options for feature matching, such as Exhaustive Matching or Sequential Matching. It can then create a sparse reconstruction and export it into different formats. Finally, it can create a dense reconstruction in the form of a point cloud which it can use to estimate a dense surface using Poisson [88] or Delaunay reconstruction. COLMAP is Open Source, is actively supported and has a broad community.

We chose to use COLMAP for our implementation, over the other presented methods such as SLAM methods, for it's good accuracy, for being Open Source and actively maintained, and because modifications are possible. Furthermore, it is easy to implement and work with COLMAP, as exhaustive documentation is available and it is platform independent. Finally, its ability to work on unordered sets of images allows for a broader use of the presented method.

3 Implementation

The implementation is divided into two pipelines that resemble each other: a Reconstruction Pipeline that creates reconstructions of the labeled data and a Label Generation Pipeline that registers unlabeled data against the reconstructions to generate new labels.

To create the reconstructions, the data is grouped based on location and COLMAP used to create a sparse reconstruction for each group of images. The Label Generation Pipeline consists of sampling the unsupervised data into groups similar to the reconstruction groups, based on proximity to any of the reconstruction groups, merging the image groups into the reconstructions, and generating labels based on the matches created in the merging process. To demonstrate the process of the proposed method, this thesis presents how labels for waste bins are generated, using camera data gathered from a commuter bus. A visual overview of the implementation can be seen in Figure 3.1.

All code related to this project has been published on github[1]. The datasets that were manually labeled or modified in this thesis have been uploaded to Kaggle[2].

3.1 BusEdge Dataset

To showcase and evaluate our implementation, we use camera data from a metro commuter bus that is equipped with five cameras on the outside. A similar bus is shown in Figure 3.2. The bus is driving between Pittsburgh and Washington County, generally two times a day. It provides a large amount of video data, combined with measurements from sensors such as a GPS and IMU module. In addition, the bus contains an onboard computer, a *RoadRecorder 9000* by *Safety Vision* which features an *Intel Core i7-8700t 2.40GHz* CPU and 5TB of storage. This allows the system to collect and store all data until it gets collected by a person and stored on a large server.

The *BusEdge* system by Ye et al. [4] allows to use the data of the bus and perform object detection. As the bus is only equipped with a CPU, large neural networks could not be used with real time performance. Sending over all data from the bus to a server in the cloud would require a high bandwidth data stream which could be costly. Therefore, the BusEdge system introduces an *Edge System* in which the onboard computer is used to filter the data and send images over to a cloud server if the lightweight model running on the onboard computer estimates a high probability that the image contains an object that should be detected. The cloud server then analyzes these images using a large neural network for more precise predictions. An overview of the BusEdge system can be seen in Figure 3.3.

[1] `https://github.com/pneug/label-generation`

[2] `https://kaggle.com/datasets/3650f5a2004c51d9dc1833c7a4075951729c75e464eca0cc7253970cd15c8524`

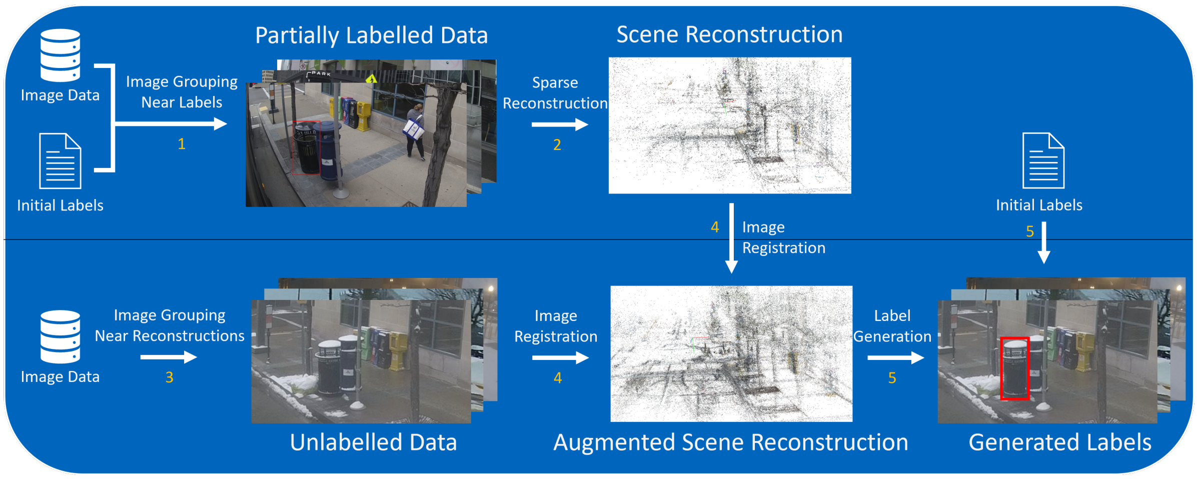


Figure 3.1: Overview of the Reconstruction Pipeline (top) and Label Generation Pipeline (bottom).

1. Partially labeled images are split into groups of images that are nearby a group of labels.
2. A sparse reconstruction is created for each group of labeled objects that helps to relocate unlabeled nearby images.
3. Images from unlabeled data get grouped based on their proximity to a group of labels, using the GPS data.
4. Using the sparse reconstructions created in step 2. and the grouped images corresponding to the sparse reconstructions, the 3D positions of feature points of the unlabeled images are determined.
5. Using the 3D positions of feature points inside the given labels and their corresponding position in the unlabeled images, new labels are generated.

Figure 3.2: A bus similar to the one in this images has been equipped with cameras, sensors, and a mobile computer to capture the data used in this implementation. Source: Rotondo et al. [89].

Most of the data of the bus is unlabeled, making it unsuitable for object detection. Ye et al. therefore present an active learning approach, called *Auto-Detectron*. This approach still requires manual labeling, which, as the authors state, could potentially be reduced by using self-supervised or semi-supervised methods. The method has been tested by detecting street signs and thrash cans with a minimum of 71 labels and 40 positive labels and a maximum of 394 labels, of which 130 are positive.

Waste bin detection on the bus has been further refined by Rotondo et al. [89] and Storm et al. [90]. Furthermore, these works classify the state of the trash can, whether it is full, empty, or there is a garbage bag next to the trash can. The presented models achieve high precision and recall, especially for large waste bins. However, the models are only tested on a dataset in cloudy conditions that is similar to the used validation set. In the following, we will use the dataset by Rotondo et al. to show that our method is able to improve waste bin detection results, both in the cloudy domain, as well as in the snowy domain.

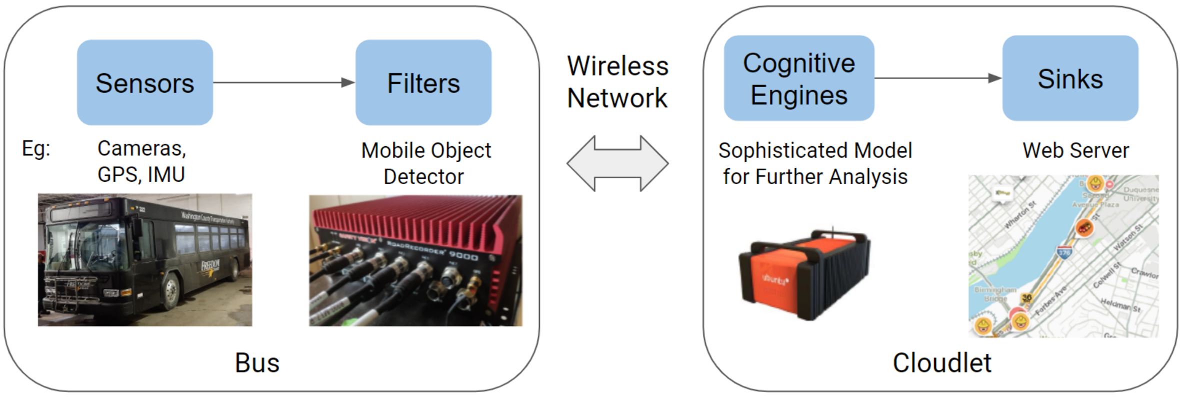


Figure 3.3: Overview of the BusEdge system that has been developed by Ye et al. [4]. Source: Ye et al. [4].

3.2 Reconstruction Pipeline

The first part of our pipeline uses images of and around labeled objects to create sparse reconstructions. The reconstructions can then be used to match new unlabeled images and generate labels for these images.

3.2.1 Input

The pipeline presented in this thesis requires a small number of labeled images that contain a GPS location. Furthermore, it needs a few overlapping images with different viewpoints of the surroundings of the labels. These images are not required to be labeled.

To showcase the method, we use a dataset that contains labels for waste bins [91]. The dataset contains 7974 images and 1563 labels. The images are from the BusEdge system. The annotations are labeled at one frame per second (fps). As our method works on static objects, we reduced the dataset to contain only static waste bins. Furthermore, as we want to show that only a small number of labels are necessary for our method, we removed most duplicate labels for the same objects. This resulted in 307 labeled objects. The camera images had insufficient overlap, therefore we added unlabeled data from the same source with 5fps, as well as data of another camera of the same bus, for the same time. Specifically, we use the partially labeled data from the forward-facing camera at the right back of the bus and the data from the forward facing camera centrally in the front of the bus.

3.2.2 Image Grouping

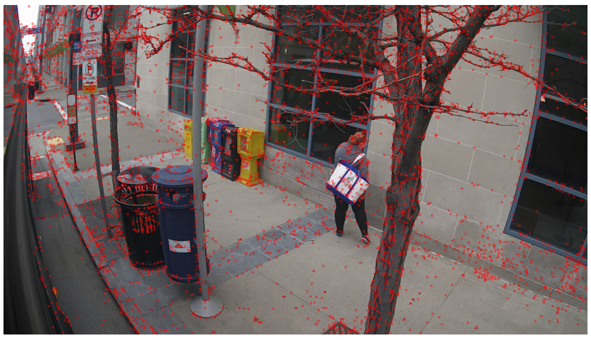
To create a 3D reconstruction of the scene around an object, COLMAP is used. COLMAP requires "a set of overlapping images of the same object, taken from different viewpoints" [92].

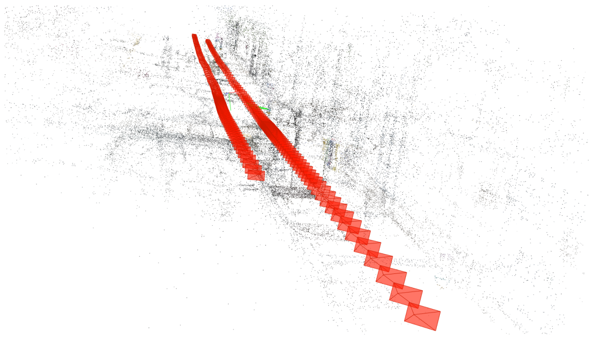
The method presented in this thesis uses the time stamps of the labeled images to group nearby labels together. If a label has been taken less than 3s apart from another label, they get grouped together. This leads to labels of the same object and labels from the same location getting grouped together. Instead of reconstructing the scene for each label, we instead reconstruct it for each group to save computation time.

To allow to reconstruct the 3D geometry based on Multi-View Stereo, a set of images of the scene are required. Using only the small number of labels per group showed to be insufficient, therefore, further unlabeled images from the same data of which some images are used, both from the same camera as well as from a second camera. The location and angle between images used for reconstruction should slightly change for optimal results, therefore our method uses the GPS data to ensure that images are only considered if the bus is moving, which greatly reduces the number of similar images. The method takes approximately 1 image per m^2. A comparison to using velocity data instead showed slightly favorable results for using GPS data, however, this might depend on the given situation, for example, the GPS accuracy might be lower in urban settings than in rural areas. We found that a total number of 130 to 250 images yields good results. Taking less images might result in a worse reconstruction, while taking more images would add extra complexity. During evaluation, it became evident that the timestamps of the camera images and the IMU and GPU data are not well synced, but instead have a roughly 3s time difference. This negatively influences the sampling process, as images are sampled less uniformly and at slightly wrong locations. As this only became apparent at a late stage during the thesis, time constraints would only allow to repeat the second part of the method with the time delay being taken into account. An analysis of the bus data of multiple dates revealed that the time delay of 3s is constant for all dates, therefore the time stamps can be adjusted accordingly by shifting them by 3s.

3.2.3 Sparse Reconstruction

Next we use COLMAP to build sparse reconstructions for each sequence. This process can be divided into three parts:


(a) **Extracted Feature Points**: The red dots show SIFT feature points that COLMAP sampled in the image.


(b) **Sparse reconstruction**: The reconstruction for the scene, each point is a 3D feature point. The red pyramids show the camera position and rotation for each image, the resulting red lines indicate the trajectories of the two cameras used for taking the pictures.

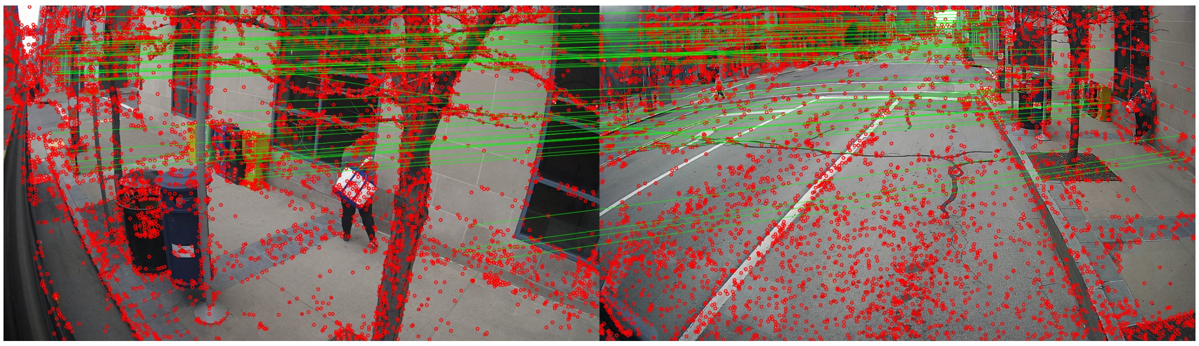

(c) **Mapped Feature Points**: The green lines show the correspondences between feature points contained in both images.

Figure 3.4: Reconstruction process

- **Feature Extraction**: In the first step, COLMAP generates feature points for each image. The default settings are used, if not mentioned otherwise. One exception are the parameters for the cameras: the method uses COLMAP's OPENCV camera option as it showed better results than the default SIMPLE_PINHOLE. As it is known that all data is from only two cameras, the images are divided into two folders so that COLMAP can automatically estimate the camera parameters per camera. To increase the number of generated feature points, we set *SiftExtraction.estimate_affine_shape*, as well as *SiftExtraction.domain_size_pooling* to true. Sample results of this process are shown in Figure 3.4a

- **Feature Matching**: Next, COLMAP attempts to match the feature points of all images. As we implemented measures to keep the number of images low, we use Exhaustive Matching which attempts to match all images with each other and leads to the best reconstruction results. Other matching strategies might use additional information like GPS data to match only some images with each other, which could potentially lead to worse results, but could reduce the processing time needed. We use the default parameters, except for setting *SiftMatching.guided_matching* to true to increase the number of matched feature points. This process is visualized in Figure 3.4c.

- **Mapping**: In the final step to obtain a sparse reconstruction, COLMAP incrementally reconstructs the scene by using Structure for Motion and the matches between the images. To refine the results, COLMAP also uses Bundle Adjustment. To obtain more 3D points by the algorithm, we set *Mapper.tri_ignore_two_view_tracks* to 0 and *Mapper.tri_min_angle* to 0.5. A sample for the resulting sparse reconstruction that is created in this step is presented in Figure 3.4b

3.3 Label Generation Pipeline

After creating the reconstructions around the given labels once, the reconstructions can be used to generate labels given unlabeled data of the same locations. The Label Generation Pipeline takes image data as input and outputs labels based on the given image data.

3.3.1 Input

The label Generation takes unlabeled images as input, as well as the given labels and the reconstructions from the Reconstruction Pipeline. The given images can be unordered and distorted.

3.3.2 Image Grouping

Image grouping works similar in the Label Generation Pipeline as in the Reconstruction Pipeline. A difference is, that as sequence groups were already obtained in the first pipeline, only the image sampling is performed in the second pipeline, based on GPS location. Similar to the Reconstruction Pipeline, the method samples 1 image per m^2. For most scenes, this results in a small number of images around the target location and shows a good trade-off between number of generated labels and processing time. However, given a specific use case, it might make sense to change the sampling scheme. For the label Generation Pipeline, the time delay of the camera data, and GPS and IMU data, described in section 3.2.2, has been taken into account and the data has been synchronized.

3.3.3 Image Matching

To match the new images with the reconstructions, COLMAP is used again. The first two steps, Feature Extraction and Feature Matching, work similar to the ones in the Reconstruction Pipeline. For the third step, we use the Image Registration functionality by COLMAP, instead of the Mapper, as this allows to register the newly added images against the existing model. To refine these results, our pipeline uses COLMAP's bundle adjustment.

3.3.4 Label Generation

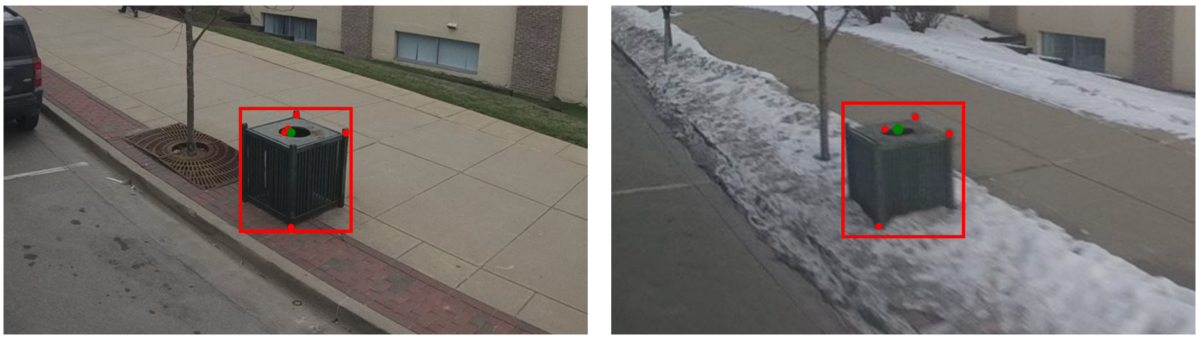

(a) Source label of (b) (b) Generated label for a new domain

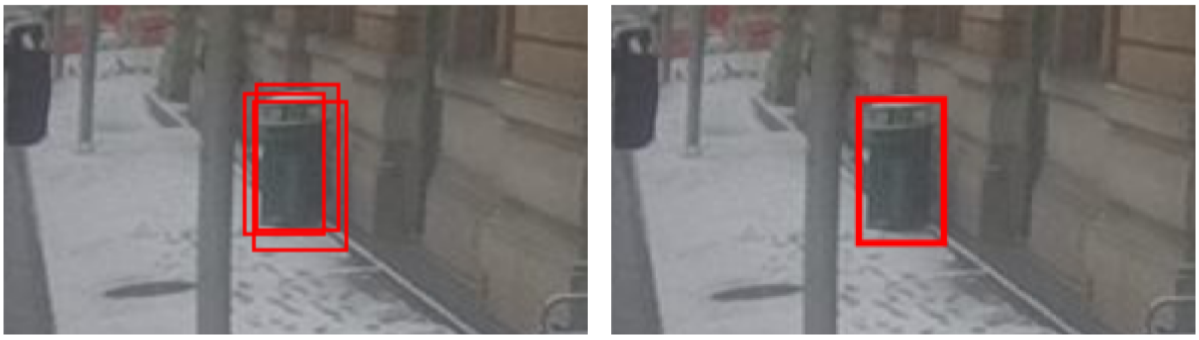
Figure 3.5: **New Label Generation**: The offset between the median (green) of the feature points (red) in the matched source and target image are used to position the new bounding box around the target object. The factor between the distances of the 3D mean points to the cameras of the source and target image is used to calculate the size of the generated bounding box.

To generate new labels, the presented pipeline checks which of the 2D feature points of the labeled images are inside one of the bounding boxes of the labels. If any of these points have been matched with any images of the unlabeled data, they can be used to calculate the position of the object in the new image.

In the first step, we collect feature points in the labeled data that coincide within the bounding boxes of one of the labels. As the shape of the object might be less than the bounding box, points from outside of the object could be potentially sampled. However, we take measures in a later step to mitigate the effects of sampling a point outside of the object.

In the next step, the algorithm finds all feature points in the unlabeled data that have been matched with the collected feature points in the labeled data. The algorithm determines the median position of the features in the source and target images. Using the median ensures that points on the outside of the bounding box, which might potentially lie outside of the target object, have less impact on the resulting bounding box. The offset can be used to determine the position of the new box. Requiring a certain number of shared feature points has shown to lead to higher quality outputs, however, it also reduces the number of output labels. A value of three required shared feature points has been chosen. By measuring the mean distance of the 3D points to the camera position of the source and target image, obtained by the reconstruction, a scale factor for the bounding box can be determined. This process is visualized in Figure 3.5.

The source labels can potentially contain multiple labels for the same object. This can lead to multiple labels being generated for the same object in the target image. Most Object Detection models require that each object should be labeled exactly once. To achieve this, these labels are combined, based on their overlap. For our dataset of static waste bins, only



(a) Multiple bounding boxes can be created per object if the object is visible in multiple source images. However, object detectors generally expect one label per object.

(b) Using a weighted mean formulation, all overlapping objects are merged into a single box.

Figure 3.6: Reduction of overlapping generated bounding boxes.

few objects have overlapping bounding boxes if labeled correctly. Therefore, our method combines all labels that overlap. For each group of boxes that supposedly show the same object, the algorithm needs to decide which bounding box to use. To achieve this, the mean of the boxes is used, weighted on the number of shared features of each box. The method has been compared to another method of taking the box that contains the most shared features, however, no significant difference could be observed in the results of both methods. An example of this step is shown in Figure 3.6.

The number of generated output labels varies a lot per object, which can lead to an imbalanced dataset. Furthermore, many of the labeled images are similar to each other, providing little new insight to the model. A simple method to reduce similar looking images is to only take one image per reconstructed scene and camera. However, this method also removes some visually distinct images, resulting in less generated labels than possible. We therefore developed a second method, called Extended Sampling, to increase the number of sampled visually distinct images. This method takes the scene, camera, timestamp, and source image into account. However, this method could potentially generate images that are less distinct from each other. As this is hard to measure, and the method only generates slightly more labels, we chose to use the first method for the implementation.

4 Evaluation

This chapter evaluates the results obtained from the presented Label Generation Pipeline. The first part analyzes the quality of the generated labels and proposes methods to improve the results and assesses their performance. The second part analyzes the practical use of the generated labels by comparing training results with and without the generated labels.

To showcase how well the label generation works, we generated waste bin labels using the labels of static trash cans given in the Waste Bin Detection Dataset to generate labels of the trash cans in other domains.

Specifically, we picked data from one bus route, from Washington, PA, to Pittsburgh and back, from each month between February 2021 to January 2022. The data is from various times of the day to maximize differences between the data. This leads to data from various weather conditions, lighting conditions, and exact routes. Furthermore, dynamic objects in the scene, such as cars or pedestrians, can change the appearance of the scenes and lead to occlusions. Information about the individual chosen data can be seen in Table 4.1.

Date	Time	Weather	Ground Condition	Lighting
18.02.2021	6:41am - 10:18am	Cloudy	Snowy+Wet	Night+Dawn+Day
25.03.2021	2:13pm - 5:54pm	Cloudy+Rainy	Partially Wet	Day
05.04.2021	2:47pm - 6:37pm	Partially Cloudy	Dry	Day
18.05.2021	2:13pm - 5:59pm	Sunny	Dry	Day
05.06.2021	12:09pm - 6:30pm	Sunny	Dry	Day
07.07.2021	6:46am - 10:11am	Partially Cloudy	Dry	Day
09.08.2021	6:45am - 10:21am	Partially Cloudy	Dry	Day
15.09.2021	6:36am - 10:24am	Cloudy	Dry	Dawn+Day
15.10.2021	6:19pm - 8:02pm	Cloudy	Dry	Day+Dusk+Night
02.11.2021	1:13pm - 5:55pm	Partially Cloudy	Dry	Day
15.12.2021	1:02pm - 7:49pm	Cloudy	Dry	Day+Dusk+Night
21.01.2022	2:34pm - 5:27pm	Sunny	Snowy+Icy	Day

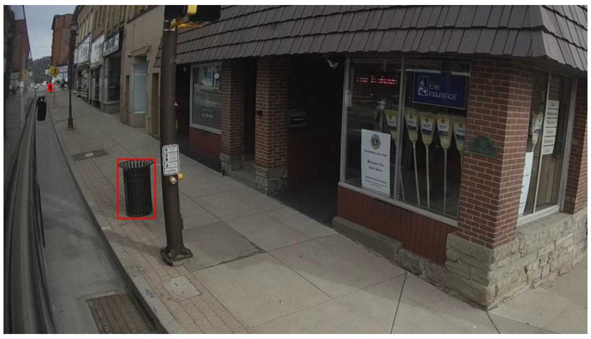
Table 4.1: Metadata of the camera data that was used for label generation. Time represents the time-span from the first to the last used image of the data, in which the bus is near a trash can.

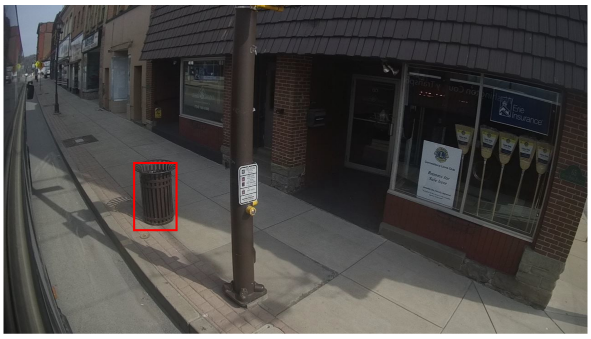
4.1 Generation Results

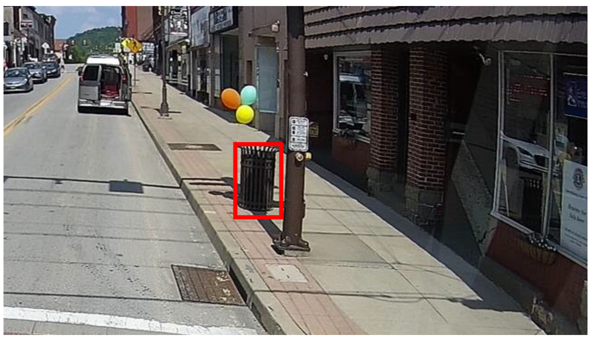
The generated labels show many strengths and weaknesses. A particular strength of the Generation Pipeline is the ability to capture the same objects under a large number of changing conditions. In the following, special cases are shown. Specifically, samples that show diverse conditions that can be captured and samples in which no or incorrect labels are generated will be discussed.

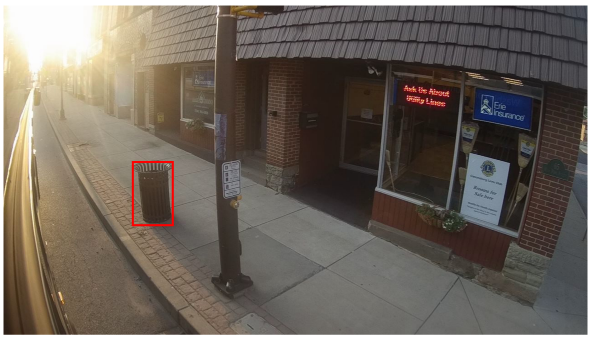
4.1.1 Learnable Domains

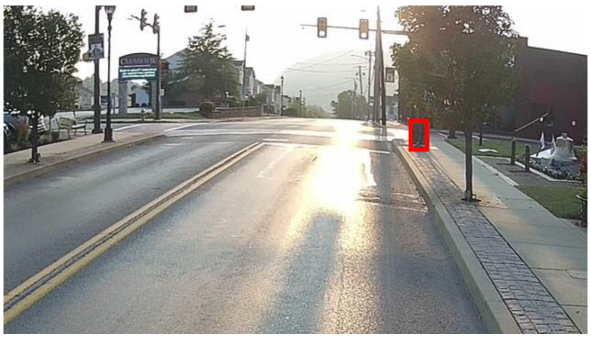
Figures 4.1 - 4.9 show samples of various domains for which our method is able to generate labeled images. While some of these domains were expected, other domains such as blurriness by motion revealed additional potential of the presented method.


(a) Source label for (b), (c), and (d)


(b) Generated label with higher sun exposure


(c) Generated label with the sun coming from a different direction


(d) Generated label with the sun impacting the visibility


(e) Generated label with shiny roads due to a low sun angle

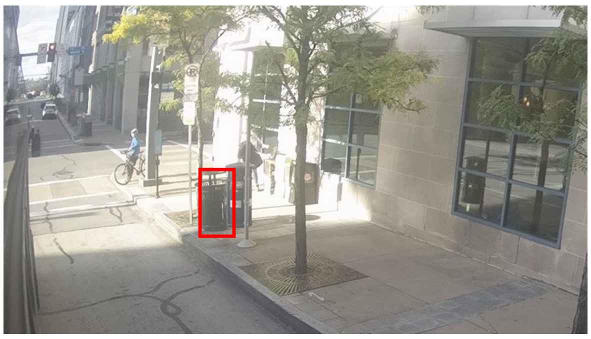
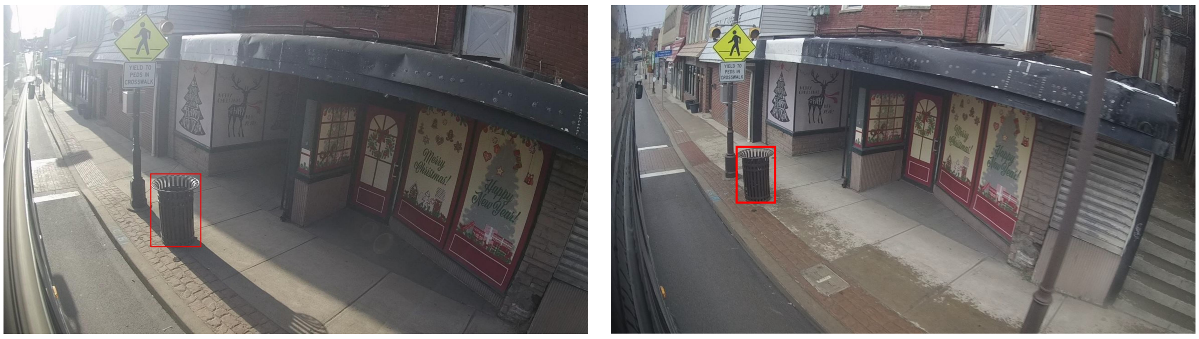
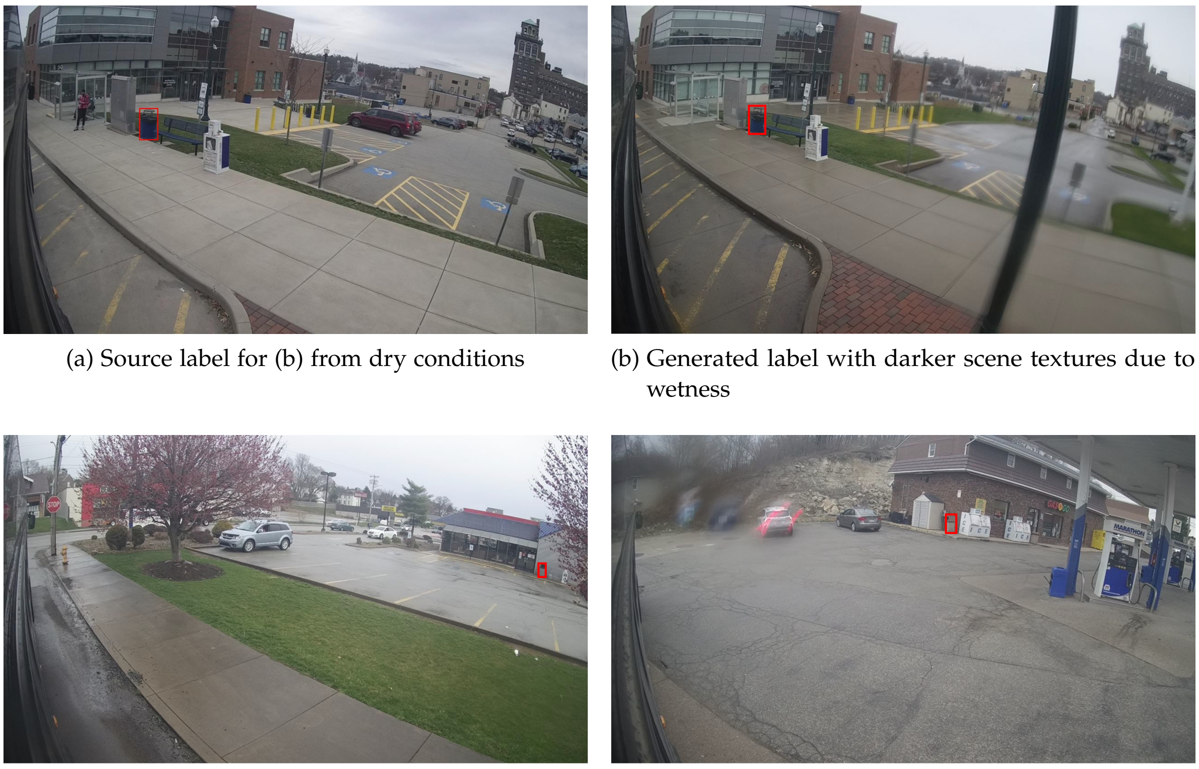

(f) Generated label with a reflection of a waste bin in a window

Figure 4.1: **Generated data under sunny conditions**: While for these samples the domain is the same as for for the given data, the generated labels can potentially still allow to train for new scenarios, such as having different levels of sun intensity or angles of the sunshine depending on the time of day and time of year. The sun can also have influence on the image quality, i.e., a ray can overlay big portions of the image. Finally, shiny reflections can let the scene appear differently and can even lead to mirror-like reflections in windows, which could easily confuse a Neural Network.



(a) Source label (b) The same object under cloudy conditions

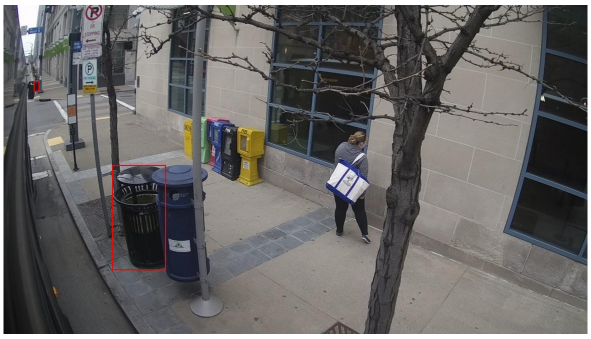
Figure 4.2: **Generated data under cloudy conditions**: With cloudy conditions, shadows seen in the sunny data do not show up or are only marginal and less reflections occur.

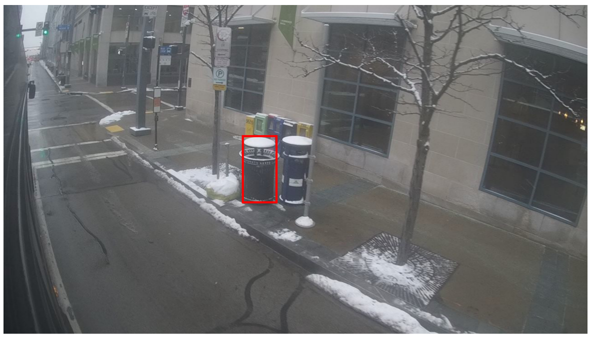


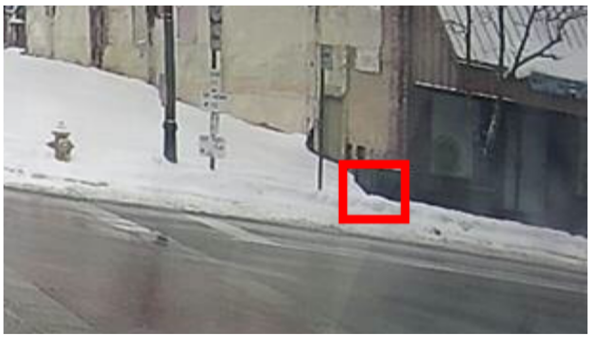
(a) Source label for (b) from dry conditions (b) Generated label with darker scene textures due to wetness

(c) Generated label containing shiny surfaces due to wetness (d) Generated label with water on the lens impairing the vision

Figure 4.3: **Generated data under rainy conditions**: Rain can both influence the visuals of the scenery as well as the image quality. Wetness caused by the rain can lead to a change in the texture of some objects, i.e., parts of the scene can have a darker texture (b) than on dry days (a), or can be shiny (c). Furthermore, wetness of the camera may potentially lead to obstructions of the view (d).


(a) Source label for (b)


(b) Generated label with the lid of the waste bin appearing white instead of Black


(c) Generated label with snow partially blocking the view onto the object

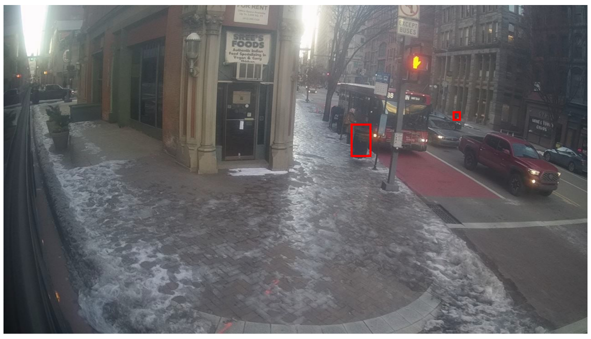
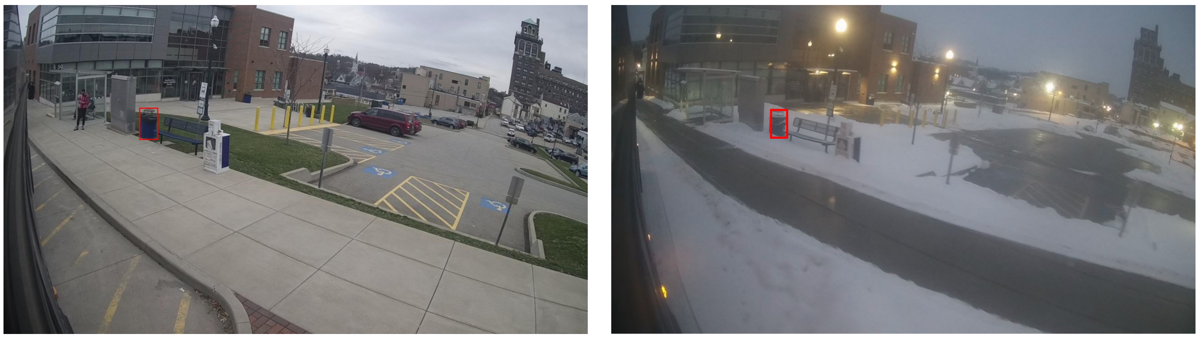

(d) Generated label with roads appearing partially shiny due to being frozen

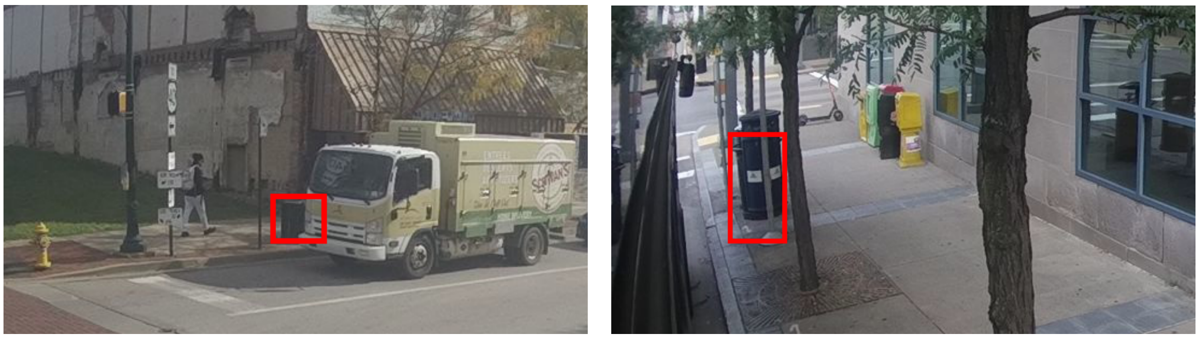
Figure 4.4: **Generated data under snowy conditions**: The influence of snow can be manifold: it can change the appearance, obstruct the view, and cause reflections of both the target objects and surrounding objects. These changes largely differ between days, i.e., there can be more or less snow, wind can move the snow to different places, and streets can be frozen, wet, or dry. This leads to a huge difference in appearance and obstructions, not only between data from sunny days and snowy days, but also between images from different snowy days.



(a) Source label for (b) during the day

(b) The scene from (a) during night

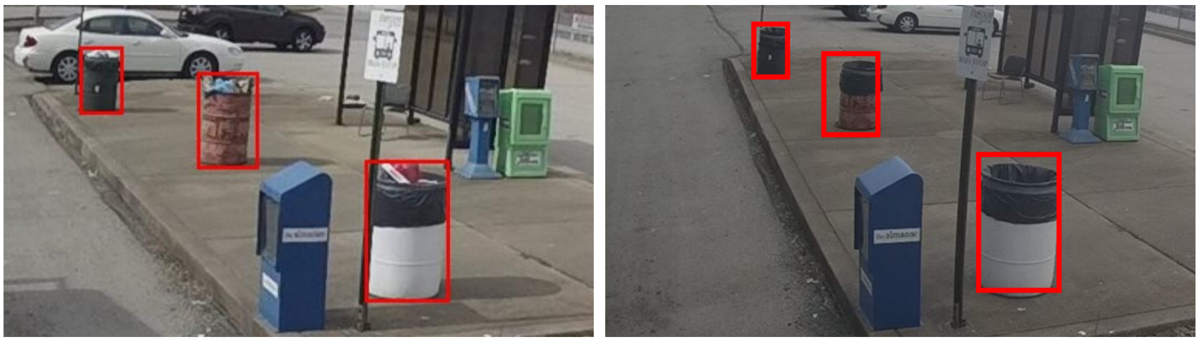
Figure 4.5: **Change of time**: Appearance can change significantly between time of day, especially between day and night. Including data by night in the training, for example, by generating it, could therefore have significant benefits. While it is more difficult to generate samples for these cases, as the appearance of the scene can be quite different, the sample above shows that it is possible in favorable cases.



(a) Generated label with a light occlusion, caused by a dynamic object

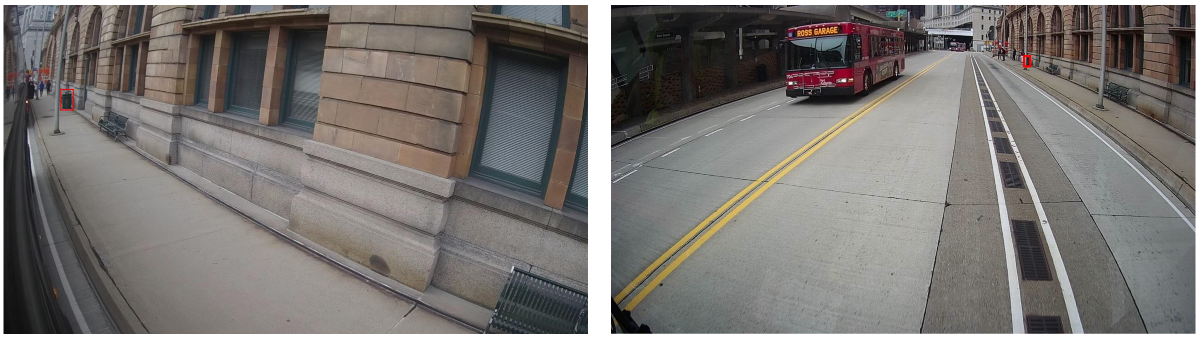
(b) Generated label with a heavy occlusion, caused by a change of view of the camera and static objects

Figure 4.6: **Generated data with occlusions**: Occlusions can be hard to learn, as objects that lead to the appearances of the occlusion can be manifold, and the level of occlusion can differ between light and heavy occlusions.



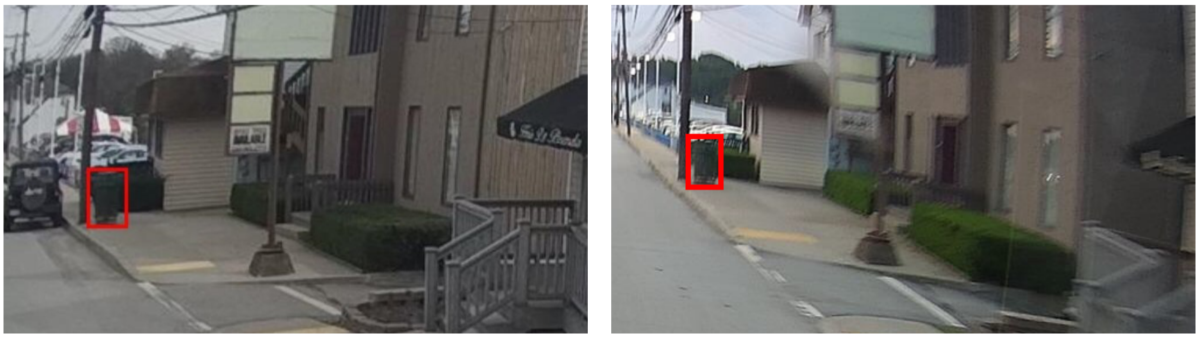
(a) Source label containing three full waste bins (b) Generated label in which the waste bins are empty

Figure 4.7: **Change of appearance**: Causes for a change in the appearance of objects can be manifold. Examples are degradation over time, vandalism, or natural occurring changes such as a plant growing. For neural networks, appearance changes can pose a major problem, especially if the training data covers only some of the appearances that can occur. For the sample of waste bins, changes in appearance have been observed, due to changes of weather or a change of fullness (a).



(a) Source label (b) The same object from a further distance, different camera and different angle

Figure 4.8: **Change of view**: Generated data can have an unseen view onto an object. This can reveal how an object looks like from a different angle or distance. Close up views of an object could potentially reveal more details in the texture of the object, while distant views can provide additional training data for this case in which it is generally harder to detect an object.



(a) Source label

(b) Generated label with motion blur caused by high speed of the bus

Figure 4.9: **Change of speed**: Motion Blur can occur when a camera moves at a high speed or has a long exposure time, or if an object moves at a high speed. It usually blurs the moving object along the axis of movement, or the entire image according to the movement of the camera. There are solutions to simulate motion blur and learn how to recover the unblurred image [93]. However, as unblurring images is of ill-posed nature, using real training data might have benefits over using synthetic data or approaches to unblur the images.

4.1.2 Analysis of Faulty Generated Labels

The box generation does not produce optimal results in all cases, due to various causes. The assumptions may not be fulfilled, for example, an object might not be static, but individual parts of the generation pipeline can fail as well. This section analyzes the failures that can occur, and which parts are prone or less prone to errors. This will provide insights on how to improve the pipeline to achieve better results in the future.

In many cases of failure, the results from the method are imprecise and the generated bounding box only roughly fits the target object. It is difficult to determine the exact cause for these cases, but generally it seems to be caused by the 3D position of the Feature Points of the reconstruction being imprecise, as shown in Figure 4.10 (a) and (b). Another, less often occurring cause is that the Feature Points used to locate the target object lie next to the object or on an occlusion, shown in Figure 4.12 (a) and (b).

Another group of wrong labels are False Negatives, cases in which a target object has not been labeled. Figure 4.11 shows the two most common reasons for this: the label generation method could only locate some of the source labels, or the source image contains only some of the objects contained in the target image. Most of these cases can be avoided with the methods presented in Section 4.2.

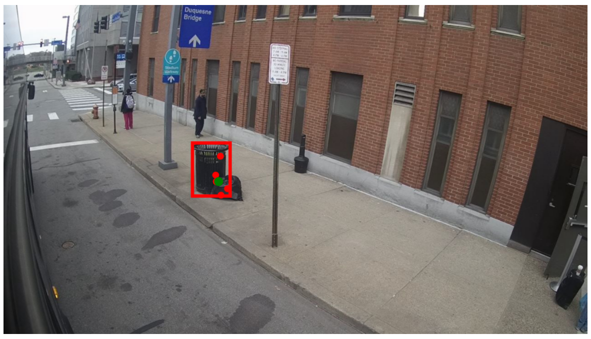
Lastly, False Positives can occur, labels that don't contain a target object. While these cases only rarely occur, it is still important to be aware that these cases are possible. Causes for these cases include that the staticness assumption does not hold, the target object is entirely occluded, or that the scene reconstruction is majorly wrong. Such samples are shown in Figure 4.12 (c) and (d), Figure 4.12 (e) and (f), and Figure 4.10 (c) and (d).

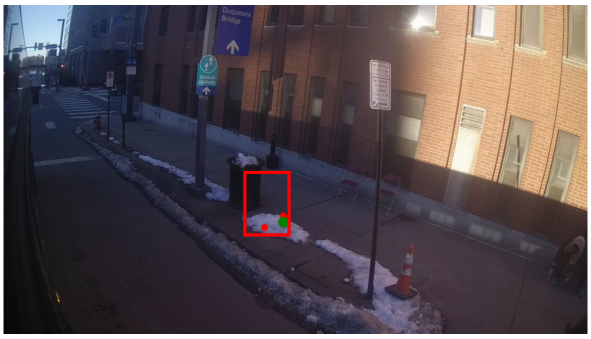
4.2 Analysis of Label Quality

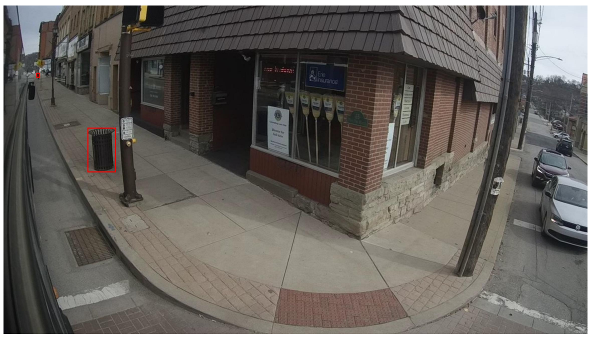
This section analyzes the results of the presented method. Furthermore, it presents approaches to improve the quality of the generated labels and evaluates the effectiveness of these approaches. Finally, we assess the impact of varying parameters of the pipeline, such as the required number of shared feature points between source and target labels.

4.2.1 Setup

For this evaluation, we use the generated data from the unlabeled bus data from 18.02.2021 and 21.01.2022, as these days feature snowy conditions and therefore grant insights into an important and difficult switch in domain from sunny to snowy conditions. To evaluate the correctness of the automatically generated labels in the snowy domain, labels have been drawn by hand for each image for which a label has been automatically generated. This was a challenging task, as in many cases, it is hard for a human to determine the labels, as objects can be small and resolution low. Furthermore, there are cases in which it is ambiguous whether an object should be labeled, as objects can be far in the distance, or almost completely occluded, and are therefore only barely visible. Even for a human, it proved to be beneficial to observe the scene during varying conditions, similar to the label generation method, to


(a) Source label for (b).


(b) Generated label from (a). Although the Feature Points in the source and target image look visually and geometrically different, COLMAP matched these points with each other, leading to an imprecise label. This is the prevalent cause of imprecise labels.


(c) Source label for (d).

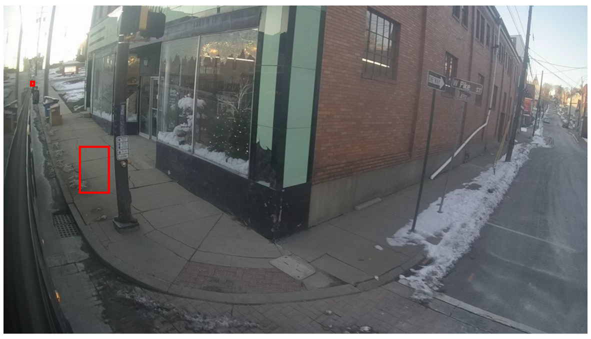
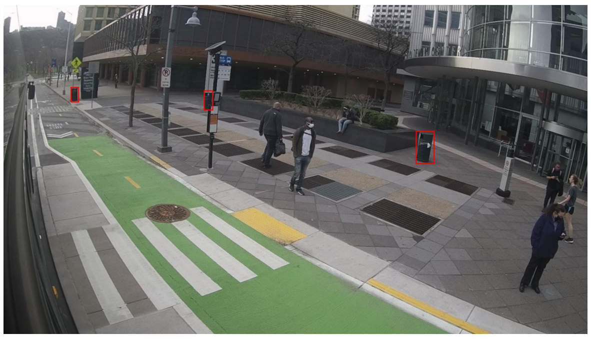
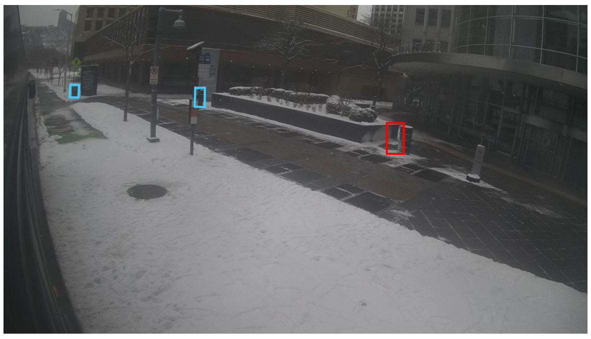

(d) Generated label from (c). The scenes in the source and target image look similar and are in proximity to each other. As a result, the reconstruction by COLMAP has mistakenly identified both scenes as the same. Subsequently, False Positve (FP) labels have been generated.

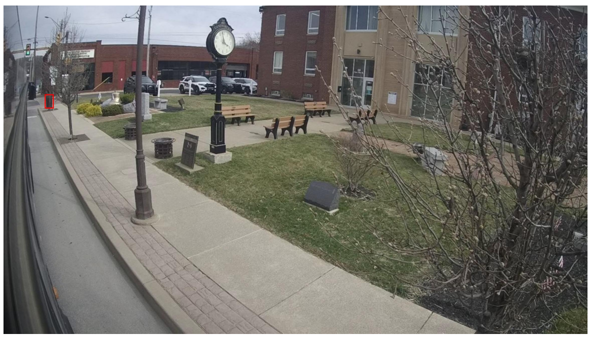
Figure 4.10: **Issues in the reconstruction**: These issues occur due to COLMAP not reconstructing the scene perfectly. While imprecise Feature Points commonly account for imprecise labels, the reconstruction only seldom has major faults.



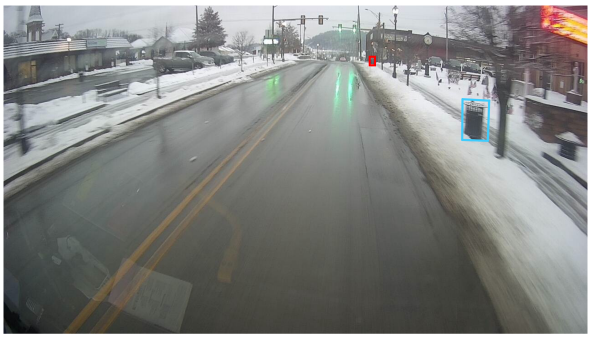
(a) Source label for (b), containing three labeled trash cans.



(b) Generated label from (a). Only one of the three trash cans that are contained in the image has been labeled, as the label generation method only creates a label if it can locate the trash can in the target image with high probability. The missing labels are marked in blue.

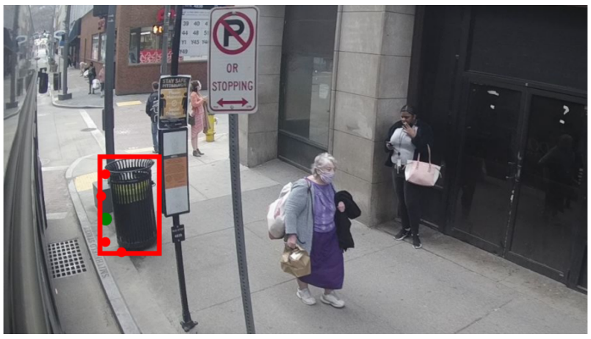


(c) Source label for (d), containing one labeled trash can.

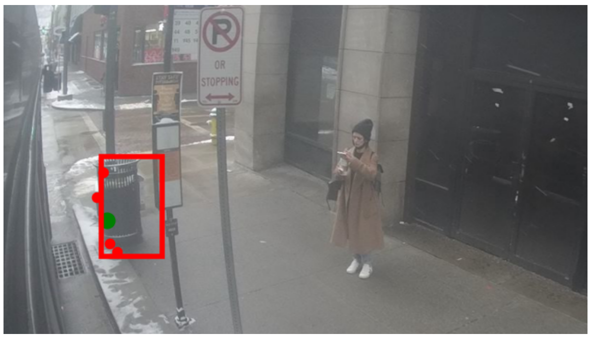


(d) Generated label from (c). As the source only shows one of the two trash cans containted in the target image, the other one is missing a label. The missing label is marked in blue.

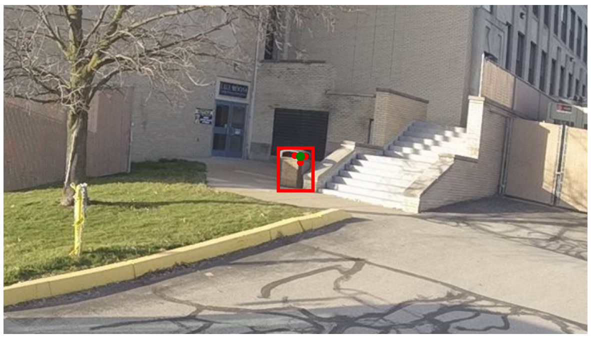
Figure 4.11: **Multiple objects per image**: Cases in which multiple objects are contained in the target image can potentially lead to missing labels. However, Section 4.2.3 and 4.2.4 present solutions how to avoid most of these cases.



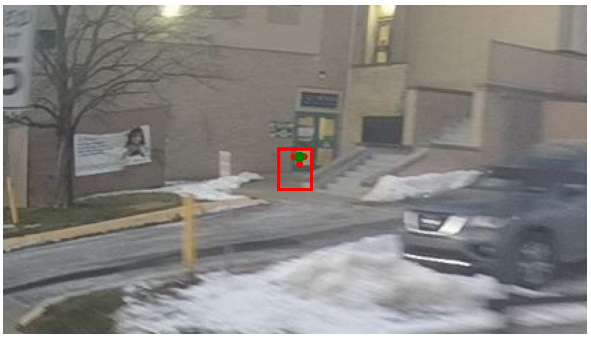
(a) Source label for (b). The Feature Points (red) are not on the target object in this sample, leading to an imprecise label.



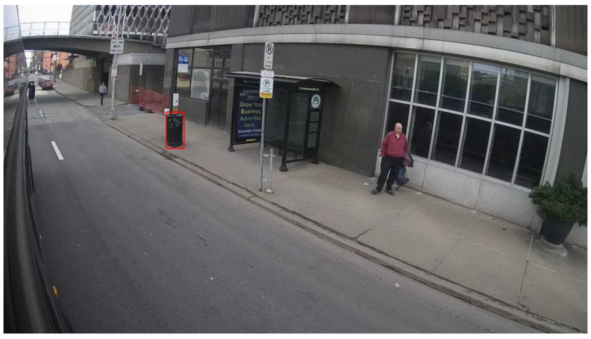
(b) Generated label from (a). In this case, in addition to the Feature Points lying outside of the target object, most Feature Points have been incorrectly matched. As a result, the generated bounding box is imprecise.



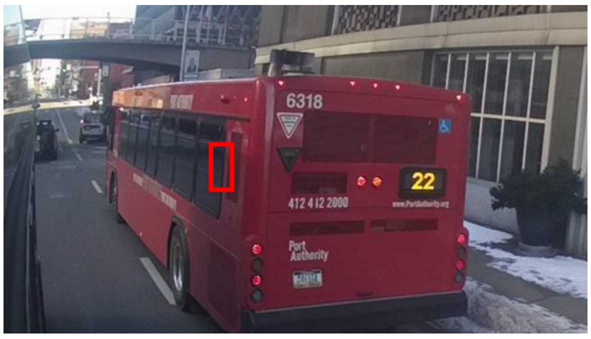
(c) Source label for (d).



(d) Generated label from (c). The trash can that can be seen in the source image has been removed.



(e) Source label for (f).



(f) Generated label from (e). In this sample, an occlusion leads to a False Positive.

Figure 4.12: **Further samples of wrong labels**: While these cases only rarely occur and are generally filtered out by requiring multiple shared feature points, it is interesting to note that in theory, these cases can occur.

determine the location of a label. Despite these challenges, the results show clear trends among the data, revealing insights into the quality of the labels.

For classification of the generated label quality, we use the following categories per label:

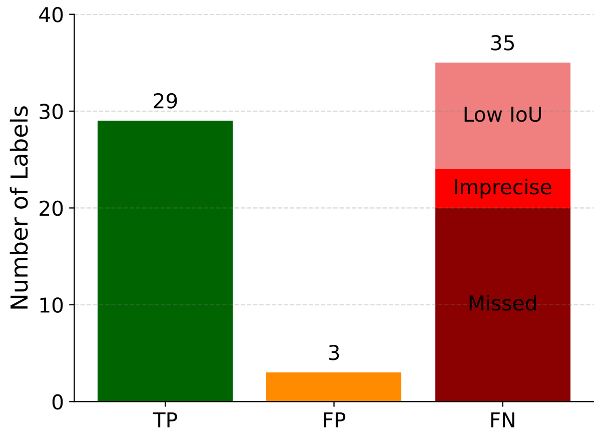
- **True Positive (TP)**: These bounding boxes sufficiently capture the object for most Object Detection tasks.
- **Low IoU**: These labels only partially contain the target object, i.e., the label only has a low Intersection over Union (IoU) with the ideal label. This categorization helps to identify whether a target object has not been captured with a high overlap due to the label completely missing or the generated label being to inaccurate.
- **Imprecise**: These labels have no overlap with any target object, but are in the vicinity of one. Similarly to *Low IoU*, these labels indicate that the cause behind missing the object lies in imprecision of the box generation method.
- **False Positve (FP)**: These labels are not in the vicinity of any of the target objects, indicating a major fault in the reconstruction pipeline or that an object has been removed.
- **Missing**: Objects that are in the target image but no generated label exists for them are categorized as *Missing*.

There are cases in which such a precise distinction might not be needed and might lead to extra complexity in diagrams. We therefore denote the cases *Low IoU*, *Imprecise*, and *Missing* as *False Negatives (FNs)*. Furthermore, the *Accuracy* of a set of images with generated labels is defined as the number of *FPs* divided by the total number of images.

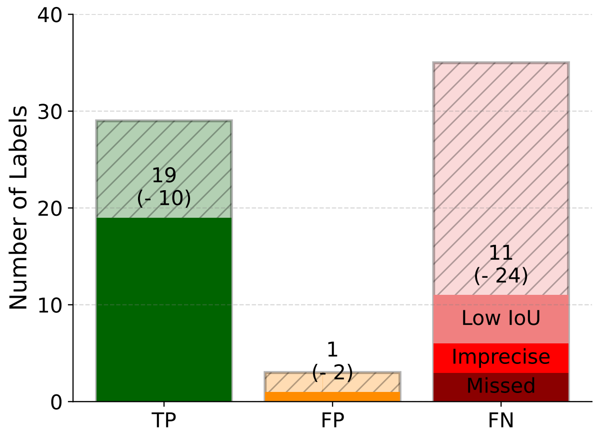
4.2.2 Label Quality without Enhancements

Figure 4.13 presents the results of the pipeline as described in the implementation. The data contains only few False Positives case, in which, for example, the staticness assumption was not given. In this case, the trash can has been permanently removed. For the False Negative cases, three main reasons have been identified:

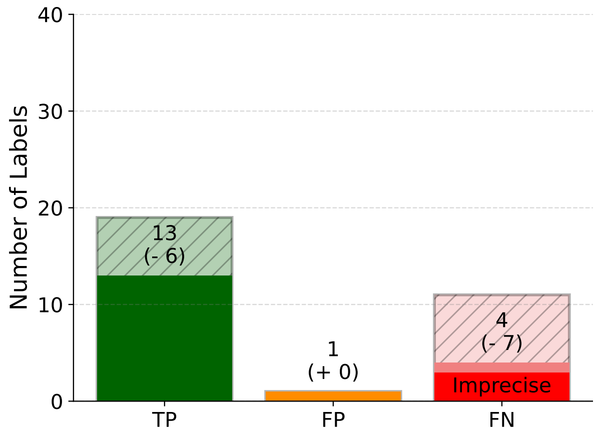
- The generated labels were inaccurate and only loosely matched with the correct labels or were nearby a correct label without any intersection.
- The object was not visible in the source image.
- The object was visible in the source image, but not enough features have been matched between the source and target location, so that no label has been generated.



(a) Results of the pipeline without modifications. The pipeline produces more false samples than true samples, signalling the need of methods to improve the label quality.



(b) Results of the pipeline with Enhancement 1, compared to the pipeline without modifications. The method increases the label quality by 18%, at the cost of reducing the number of good labels by 34.5%.



(c) Results of the pipeline with Enhancement 1 + 2, compared to the pipeline with Enhancement 1. This method increases the label quality by 10.9%, however, it reduces the number of good labels by 31.6%.

Figure 4.13: **Classification of the generated results from the pipeline**: Modifications can improve the quality of the labels, at the cost of missing some of the correct labels.

4.2.3 Enhancement 1: Requiring a Minimum Number of Labels

The label quality can be improved by requiring that the generated image contains at least as many annotation boxes as any of its source images. This effectively removes FN cases that come from labels not being matched. However, it can also reduce the number of TPs. The results are shown in Figure 4.13. While with this method 2 of the correctly labeled images are not contained anymore, it successfully eliminates 17 of the FN cases caused by missing matches, reducing the number of labels categorized as *Missed* from 20 to 3. This results in an overall improvement of the accuracy for images from 43% to 61%. For most applications, label quality is more important than label quantity, therefore we use this method as the baseline to evaluate further methods.

4.2.4 Enhancement 2: Using Only the Source Camera

For the sample of the trash can label generation using data from a transit bus, further FN cases can be avoided by only using data from the same camera with the same positioning on the bus as the one used for the source labels. Using this camera, the perspectives of the source and target images are generally closer to each other, resulting in less objects that can be seen in the target image but not in the source image. However, samples by cameras other than the source camera could prove especially useful in the training process, as the change in perspective could lead to more diverse samples. Additionally, this method might not be suitable for data from other sources, for example, when using crowd sourced images. Results of this method can be seen in Figure 4.13. All missed cases could be avoided, and the accuracy is increased from 61% to 72%, compared to the results when only Enhancement 1 is applied. However, the number of TP labels is reduced by 32%, making this method only suitable in some use cases.

4.2.5 Influence of the Box Generation Parameters

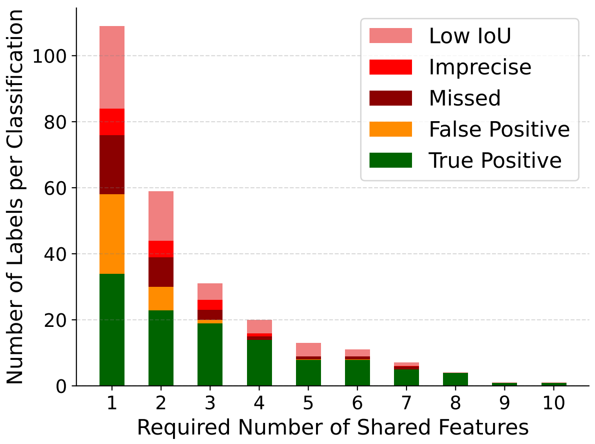
In the Implementation in Section 3.3.4, we propose to require three feature points to be shared between an annotation from the source image and target image to generate a label. However, this parameter can be adjusted, depending on the required level of accuracy of the labels, to generate less but higher quality labels or more but lower quality labels. Figure 4.14 (a) shows the effect of changing this parameter: setting it to low values generally produces more TP labels but has a lower accuracy, while setting it to high values leads to generating less labels but of higher quality.

Figure 4.14 (b) illustrates the influence of the method to determine the mid point of the Feature Points on the accuracy, for different values of the parameter *Required Number of Shared Features*. Using the median shows a tendency of producing better results.

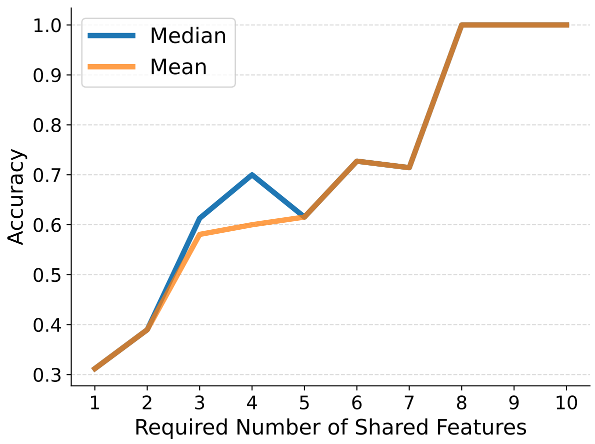
Figure 4.14 (c) presents the pareto front for different values of the parameter *Required Number of Shared Features* and Enhancement 1 or Enhancement 1+2. The front reveals the optimal choices of parameters given a required quality level of the labels. For low accuracy, Enhancement 1 with 1-4 required shared feature points achieves the best results, while for higher accuracy Enhancement 1+2 with 3-7 required shared feature points performs best.

4.3 Training on the Generated Data

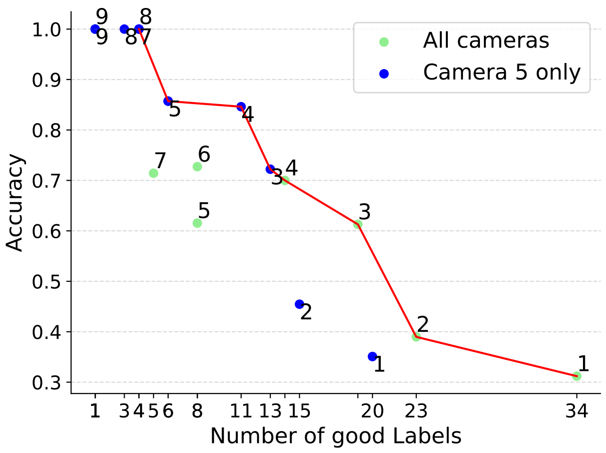
Determining the output of a deep learning model trained on a given dataset by only observing the model and the input data is generally not possible. Therefore, this section shows the performance achieved by training a well established model on the original hand-labeled data and the data with automatically generated labels. These experiments can indicate whether



(a) Label quality in relation to the parameter *Required Number of Shared Features*. Lower values lead to more labels being generated, while higher values increase the accuracy.



(b) The influence of the method to determine the mid point of the Features on label quality appears to be insignificant for the label quality.



(c) Pareto Front of Enhancement 1 (All cameras) and Enhancement 1+2 (Camera 5 only), showing the front with the best combinations of the parameter *Required Number of Shared Features* (number next to the points) and either Enhancement 1 or 1+2.

Figure 4.14: **Influence of parameters of the pipeline on label quality**: While the parameter *Required Number of Shared Features* provides a trade-off between label quality and number of labels, the effect of the choice of the method to determine the mid point of the shared feature points seems to be marginal.

the generated data is working well for real use cases or whether the quality of the generated data is insufficient.

4.3.1 Training Setup

We chose to use a Faster R-CNN model for comparing training on the original and the generated data, as it is widely established in the object detection field. We used the R-50-FPN backbone model, which is based on the MSRA's original ResNet-50 model, combined with a Feature Pyramid Network. This backbone is broadly used in the field of object detection and segmentation and therefore grants good insights into the general suitability of the generated data for training models. The model has been pre-trained for around 37 COCO epochs on the COCO dataset[1]. The COCO dataset contains 80 classes, consisting of street and houshold objects, animals, persons, food, and more. There is no waste bin class or similar in the COCO labels.

For the training, Detectron2 is used, as it allows to quickly train established models, such as the Faster R-CNN on new data in COCO format. COCO format is a json file that contains all data needed for training, such as information about the images and their annotations. The Model Zoo of Detectron2 provides models pre-trained on the COCO dataset. For the training, we use Early Stopping with a validation frequency of 500 batches. Early Stopping periodically applies inference on a validation dataset to measure whether further training achieves better results on this unseen data or worse results due to overfitting on the training data, and stops the training in case of overfitting. Deep Learning models overfit after a varying number of steps, i.e., they learn features that are specific to the training data, instead of learning general features. To reduce training time, only samples that contain at least one annotation are used for training and validation. We set the maximum training length to a high number, so that the training will always be stopped by Early Stopping. This resulted in an average length of around 2000 steps. We use a batchsize of 4 and *BATCH_SIZE_PER_IMAGE* of 128. For augmentations, *ResizeShortestEdge* and *RandomFlip* have been used which are the default augmentations for Detectron2 and therefore represent typical training behavior.

Multiple metrics to measure the object detection performance of a model exist. The precision denotes the number of TP predictions divided by the number of total predictions (i.e., True Positives plus the False Positives): $Precision = TP/(TP + FP)$. A predicted bounding box is considered correct if it has the highest overlap with a ground truth bounding box. To measure the overlap, the Intersection over Union is used. The IoU of two boxes is defined by $IoU = AreaOfIntersection/AreaOfUnion$; Area of Intersection is the area in which the two boxes intersect, Area of Union is the area that is the total area covered by the two boxes combined. The IoU is in a range between 0, no overlap between the boxes, and 1, the boxes are identical.

Choosing different values for the threshold IoU for object detection determines which predictions are counted as correct or incorrect: Using a high IoU, only predictions that almost perfectly match the target label are counted. In contrast, by using a low IoU, boxes with only a small overlap are still identified as correct.

The Recall is denoted as $Recall = TP/(TP + FN)$ and measures how many of the target objects have been correctly predicted. The Average Precision combines both Precision and Recall, by calculating or estimating the Area under the Precision-Recall Curve. To obtain the Precision-Recall Curve, the predictions are sorted by their confidence. The first value is then drawn into a diagram, next the Precision and Recall of the first and second prediction are used, etc. This generally results in a curve in which the Precision is decreasing with increased recall values. Different definitions for the Average Precision exist, that generally calculate the area under the Precision-Recall Curve or of a smoothed version of the curve.

This thesis uses the COCO mean Average Precision (AP) metric, as defined by the COCO competition, to measure the performance of the model on different inputs and test sets. To measure the Average Precision, this metric interpolates the Precision-Recall Curve and uses 101 points to estimate the area under the curve. It then calculates the mean of the Average Precision for 10 different IoU values between 0.5 and 0.95 with a step size of 0.05. AP values range between 0 and 100, with 100 representing perfect performance. The mean Average Precision is less noisy than using only the Average Precision, and takes both loosely matching predictions as well as closely fitting matches into account, but ranks close matches higher. This metric is the primary COCO competition metric. This thesis uses the AP metric, as it is commonly used and is well suited for most use case scenarios.

4.3.2 Datasets

To showcase the performance with and without the generated labels, various test sets are required. As this thesis has a focus on Domain Adaptation, these datasets have been captured under different domains. For the sample of waste bins, domains can be different weather conditions, lighting conditions, different objects and different views. This section presents the test datasets that are used for evaluation.

Waste Bin Detection Sunny

The work of Rotondo et al. [89] presents two novel datasets that have been created for trash can detection. The first dataset contains images from the BusEdge system captured predominantly in the sunny domain, with dynamic and static trash cans labeled. The second second contains similarly labeled images in the cloudy domain. The Rotondo et al. use the first dataset for training and divide the second dataset into a validation and test part. For this theses, this split is used as well. As this thesis covers label generation for static objects only, we adapted the datasets and removed all labels that contain dynamic waste bins. Furthermore, to simulate a scenario in which labels are scarce, labels containing the same object have been reduced to a small number for the train set. This dataset is be referred to as the Sunny dataset, or the dataset in sunny domain, in this thesis. These modifications result in a total of 7144 images of which 204 contain at least one instance of a trash can. A sample of the dataset is shown in Figure 4.15

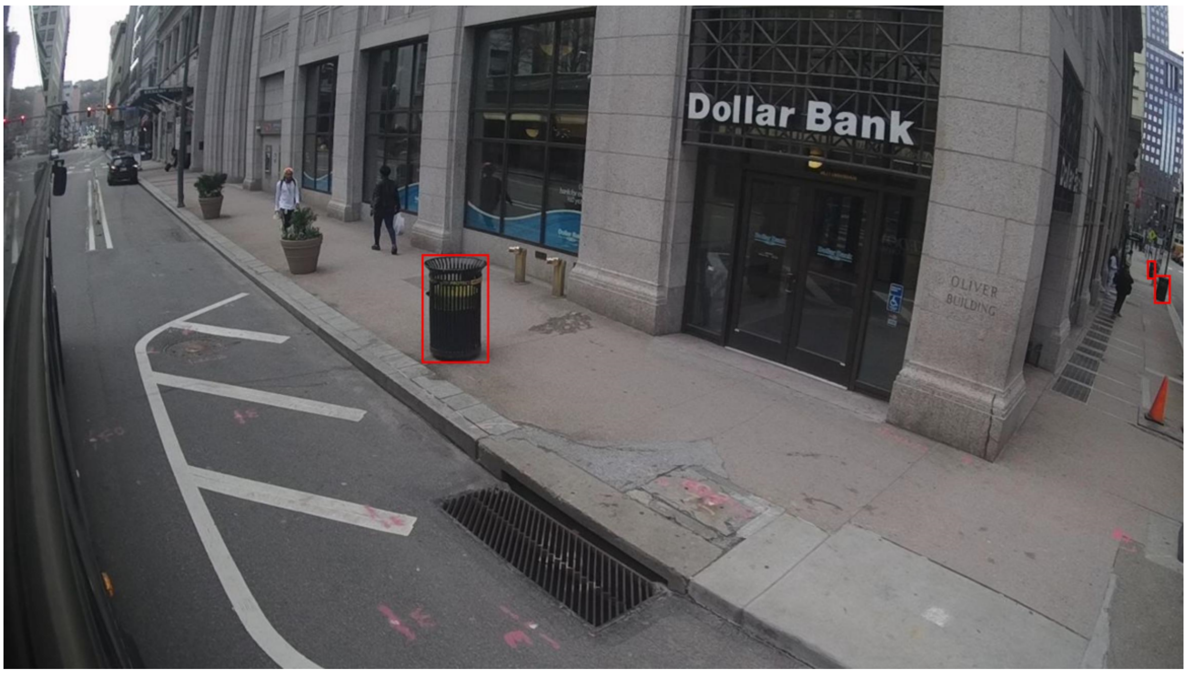
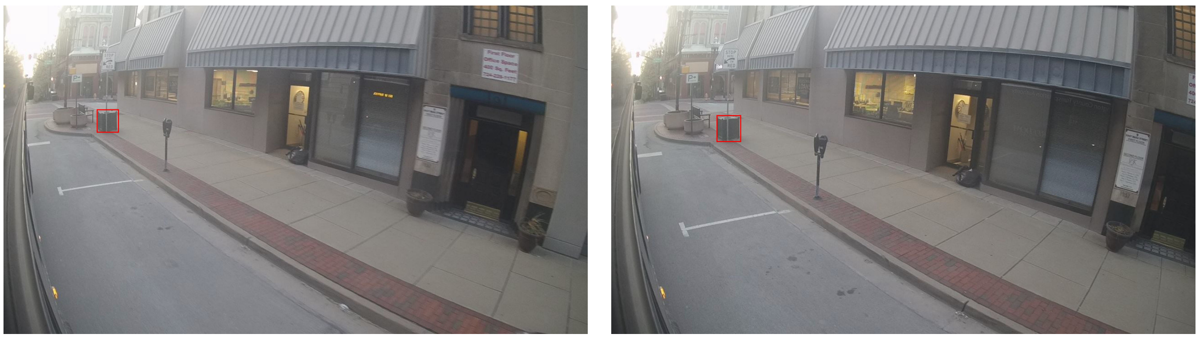


Figure 4.15: **Sample from the Sunny dataset**: A common challenge is the detection of objects that appear small in the image, such as the trash cans on the right side of the image.



(a) Test Set sample

(b) Validation Set sample

Figure 4.16: **Samples from the Cloudy$_{\text{Val}}$ and Cloudy$_{\text{Test}}$ datasets**: Many images of the test set (a) and validation set (b) of the Waste Bin Detection Dataset look similar to each other, therefore generating little insights about the generalization of a model. To generate better insights, we labeled further data with more differences.

The datasets containing only labels of static objects in the cloudy domain are further refered to as Cloudy$_{\text{Val}}$ and Cloudy$_{\text{Test}}$. A limitation of these datasets is, that both datasets are very similar to each other, as they are from the same camera, time and in some cases similar location. The same objects that are in the validation set can also appear in the test set with only slightly different angles. Figure 4.16 shows two of these samples that mostly resemble each other. To gain better insights into the generalization of a model, we labeled additional data from different sources, times and weather conditions. The Cloudy$_{\text{Val}}$ dataset contains 3438 images, of which 354 images contain a trash can. Similarly, the Cloudy$_{\text{Test}}$ dataset contains 3437 images, of which 338 contain a trash can.

Snowy Datasets

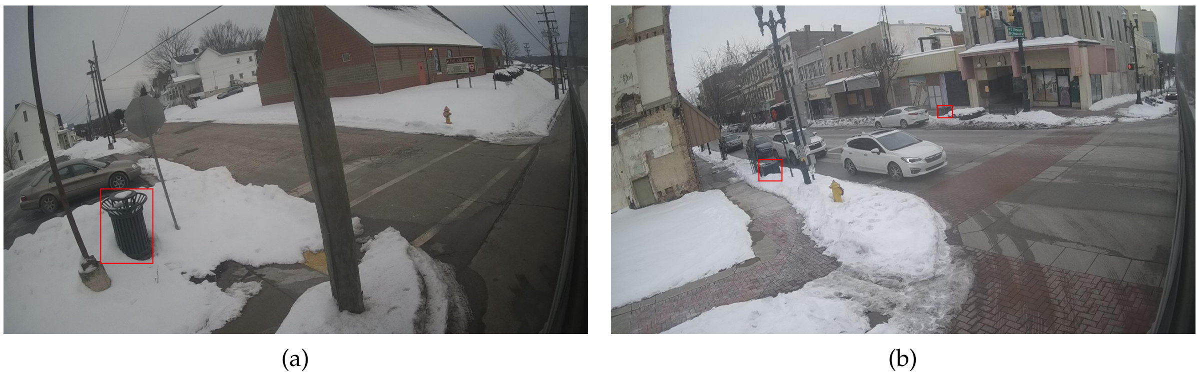


Figure 4.17: **Samples from the Snowy$_{\text{Day}}$ dataset**: Snow can heavily impact the visuals of the objects to be detected. In (a), a small layer of snow leads to the top of the waste bin looking partially white instead of black, while in (b) the snow heavily occludes part of the waste bin, which changes the visible shape of the object.

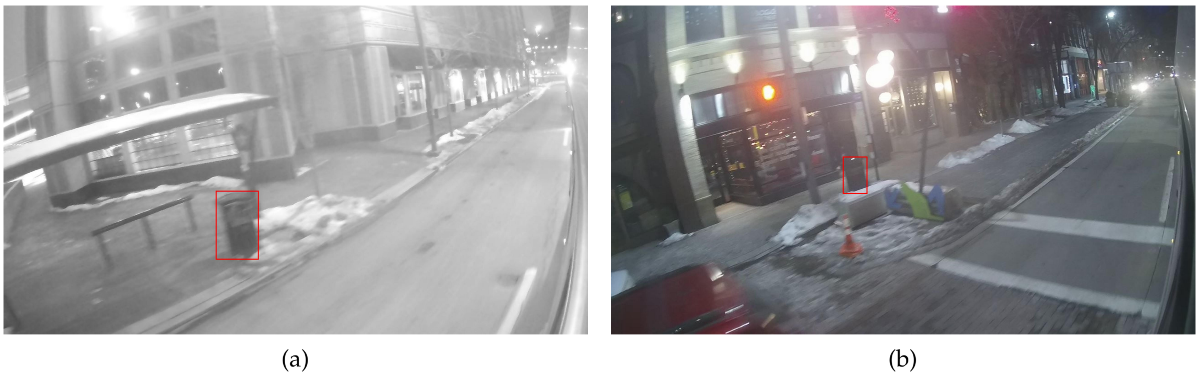


Figure 4.18: **Samples from the Snowy$_{\text{Night}}$ dataset**: Night images bring numerous challenges that are not or less present during the day. Specifically, the saturation can vary between images, letting some images appear without colors (a), and some images with some degree of color (b). Furthermore, the images are often more blurry than during the day.

We labeled data from the bus on a snowy day that has not been used for label generation, 17.02.2021, and for which the amount of snow varies to the data used for generation. The data comes from a different camera that points back from the right front of the bus, therefore has a different angle to both the original and generated training data. The images are from the bus going from Washington, PA, to Pittsburgh and back. We labeled approximately 3 images per waste-bin from different angles and distances and discarded further images of the same objects to improve balance in the dataset. As there is a longer stop in Pittsburgh, the first half of the data is taken during the day and the second half during night. We therefore split the data into a Snowy-Day dataset and a Snowy-Night dataset. The Snowy-Day dataset contains 236 object labels and 4214 images, while the Snowy-Night dataset contains 119 object labels and 4349 images. The discrepancy of labels in both datasets is caused by the much worse visibility during night, making it harder for a human to identify waste-bins with a high probability.

Roadbotics based Dataset

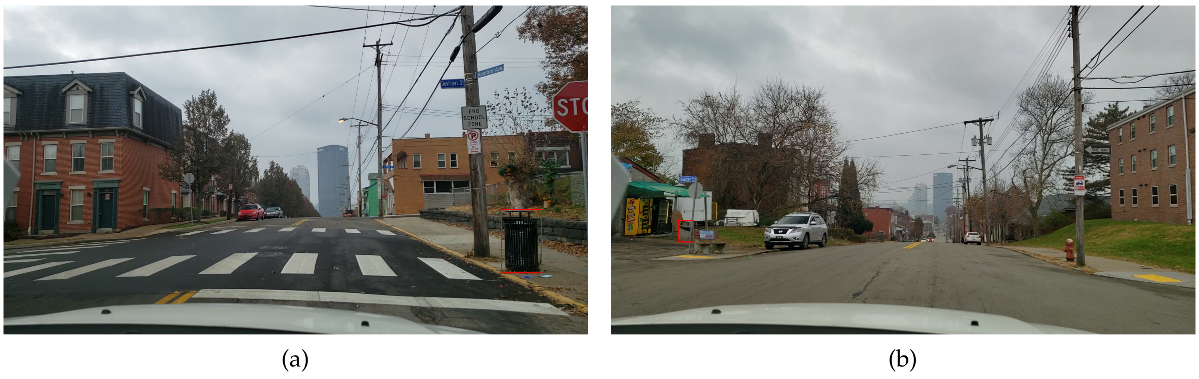


Figure 4.19: **Samples from the RoadBotics based dataset**: The RoadBotic based dataset brings images from a new perspective, camera, and covers locations and objects that are not contained in any of the training data. While some objects look similar to the objects found in the training data, such as in (a), others can also have a new appearance (b), making it a good test to analyze whether a model is only remembering seen objects or can also generalize to new but similar objects.

Roadbotics published a dataset [8] that contains video data taken by a car in different cities throughout the US. The dataset does not contain labels for waste bins. Therefore, we labeled 33 minutes of video data at 1 frame per second by hand. To keep the dataset balanced, we again labeled only around 3 images per object and discarded all other images that contain the same object, resulting in 83 object labels and 1377 images. We chose to take data from Pittsburgh, so that the waste bins generally look similar, however, little or no waste bins occur both in this test set and the initially labeled or generated training data. This dataset provides insights into how the models perform on data taken from another vehicle and camera and therefore better presents how well a model is able to generalize.

Generated Datasets

To measure the impact of using generated data to train object detectors, multiple datasets consisting of automatically generated data have been created. For most of these datasets, only generated data from the snowy domain is used. This is in order to measure the impact of adding generated data from the target domain to the training. These datasets use the data obtained by the bus on 18.02.2021 and 21.01.2022, the same data that has been used for the evaluation in the previous section. The datasets vary in whether they contain similar images or not and in whether they have been supervised by a human:

- $\textbf{Generated}_{\textbf{snowy}}$: This dataset contains all labels that have been generated for the given data in snowy domain. This means that the data is most exhaustive, containing 334 images, however, many images are visually highly similar.
- $\textbf{Generated}_{\textbf{snowy,unique}}$: To reduce the number of visually similar images, this dataset contains only one image per group of objects and camera. This reduces the number of images to only 17 images, however, these images are visually highly distinct.
- $\textbf{Generated}_{\textbf{snowy,reviewed}}$: To measure the impact of the quality of the labels, we removed all low quality images from $\textit{Generated}_{snowy}$ by hand. Furthermore, highly similar images were removed by hand. These modifications result in this dataset with 41 generally visible distinct images.

The forth dataset containing generated data is $\textbf{Generated}_{\textbf{all domains}}$. To create this dataset, our pipeline has been applied to all data that has been described in Table 4.1. The source data, and therefore the generated labels, are from a huge variety of domains.

For all automatically generated datasets, we used Enhancement 1 and five required shared feature points.

Overview

Table 4.2 grants an overview over all hand-labeled and generated datasets used for evaluating training performance for object detection.

4.3.3 Training Results

To measure the impact of using the generated data for training, different combinations of hand-labeled data and generated data have been explored. Table 4.3 reveals that by only training on the generated data, the trained model performs significantly worse on all test datasets than training on the hand-labeled dataset from the sunny domain. This is presumably caused by the low number of generated labels and lower quality of these labels compared to the hand-labeled images.

Dataset	Labeled Objects	Images w. Trash Cans	Total Images	Domain
Sunny	307	204	7144	Sunny, Day
$\text{Cloudy}_{\text{Val}}$	465	354	3438	Cloudy, Day
$\text{Cloudy}_{\text{Test}}$	458	338	3437	Cloudy, Day
$\text{Snowy}_{\text{Day}}$	236	168	4214	Snowy, Day
$\text{Snowy}_{\text{Night}}$	119	101	4349	Snowy, Night
RoadBotics	83	58	1377	Cloudy, Day
$\text{Generated}_{\text{snowy}}$	347	334	334	Snowy, Day
$\text{Generated}_{\text{snowy,unique}}$	17	17	17	Snowy, Day
$\text{Generated}_{\text{snowy,reviewed}}$	41	41	41	Snowy, Day
$\text{Generated}_{\text{all domains}}$	109	104	104	Multiple

Table 4.2: Overview of all datasets that have been used for training and testing.

However, when combining the hand-labeled data with the generated data, clear improvements can be achieved over using only the hand-labeled data, as shown by the data in Table 4.4.

In this set of experiments, the impact of using different sets of generated data in combination with hand-labeled data is analyzed. The following insights are revealed from the test results:

- Almost all sets of generated data improve the results on every test, showing that the data is well suited even for general Domain Adaptation.
- Combining the hand-labeled data with the generated data from various domains achieves the best results overall on the Cloudy dataset and the RoadBotic based dataset. Furthermore, it outperforms the other datasets that were not reviewed by a human on the $\text{Snowy}_{\text{Day}}$ dataset. This shows that in general, generating more data of any domain can improve the results. Hence, this method is not only well suited for Domain Adaptation, but also label generation for the same domain.
- Using the full generated data outperforms both the variant of the data in which only visually highly distinct images were automatically selected, as well as the variant in which low quality labels and labels that look visually similar were removed by hand. This indicates that even though labels might look mostly similar to each other, they can still have a positive impact on the performance of the model.

In summary, incorporating automatically generated data in the training data can substantially improve the object detection results for new domains. This means that our method can improve results without requiring manual labeling.

Train Dataset \ Test Dataset	$\text{Cloudy}_{\text{Test}}$	$\text{Snowy}_{\text{Day}}$	$\text{Snowy}_{\text{Night}}$	RoadBotics
Sunny	**38.4**	**22.6**	**29.4**	**23.7**
$\text{Sunny+Generated}_{\text{snowy}}$	27.4	13.0	16.3	19.4
$\text{Sunny+Generated}_{\text{snowy,unique}}$	4.9	5.3	6.7	9.7
$\text{Sunny+Generated}_{\text{all domains}}$	32.3	17.4	25.5	22.0

Table 4.3: **Using only generated data for training**: This table shows the results of training on data generated under snowy conditions and generated from various domains, and compares the results to training on the hand-labeled data from sunny domain. Training only on the generated data of the target domain significantly decreases performance, likely due to the lower number of labels. Each cell contains the Average Precision achieved when trained on the dataset in the row and tested on the dataset in the column, the $\text{Cloudy}_{\text{Val}}$ dataset is used for validation. Higher AP values are better; AP values marked in bold mark the best result on this test set among these experiments.

Train Dataset \ Test Dataset	$\text{Cloudy}_{\text{Test}}$	$\text{Snowy}_{\text{Day}}$	$\text{Snowy}_{\text{Night}}$	RoadBotics
Sunny	38.4	22.6	29.4	23.7
$\text{Sunny+Generated}_{\text{snowy}}$	40.5	22.6	**32.5**	30.2
$\text{Sunny+Generated}_{\text{snowy,unique}}$	39.2	22.9	28.3	26.2
$\text{Sunny+Generated}_{\text{snowy,reviewed}}$	40.2	22.5	30.7	28.3
$\text{Sunny+Generated}_{\text{all domains}}$	**42.5**	**23.6**	27.8	**30.4**

Table 4.4: **Training on hand-labeled + generated data**: This table shows the results of different combinations of generated data and hand-labeled data on object detection performance. The results show that adding generated data to the hand-labeled data improves detection performance in most cases. Each cell contains the Average Precision achieved when trained on the dataset in the row and tested on the dataset in the column, the $\text{Cloudy}_{\text{Val}}$ dataset is used for validation. Higher AP values are better; AP values marked in bold mark the best result on this test set among these experiments.

5 Discussion

The evaluation in the previous section shows that the proposed method is able to generate high quality labels for object detection. However, it also highlights various shortcomings of the method. This chapter elucidates one of the biggest limitations, analyses its causes, and presents potential improvements.

5.1 Number of Generated Labels

The evaluation proves that generally, higher amounts of generated data can improve the object detection results. However, the presented implementation produces labels for only a small fraction of the provided unlabeled images. To understand what is causing this, all parts of the proposed pipeline need to be investigated. The following overview analyzes which parts of the pipeline have an influence on the number of generated labels, based on the data from the BusEdge system from 21.01.2022:

- In the first step, nearby labels are grouped together to reduce workload for later steps and unlabeled images from similar locations are sampled. This results in a total of 112 groups, referred to as scenes. These might either contain multiple images of the same objects or of objects in close proximity to each other. While this step in itself does not directly influence the number of labels that get produced, it has an impact on the performance of the later steps. Especially, inaccuracy of the GPS can lead to problems at a later step, as well as the time difference between the clock of the GPS and IMU sensor and the clock of the cameras. By factoring in the miss-match in the time steps, more accurate sensor data might be obtained. Furthermore, an exploration of different algorithms to sample nearby unlabeled images might improve the reconstruction results. Alternative algorithms include using GPS or velocity measurements to sample images.
- The method to sample unlabeled images for label generation leads to unsatisfactory results. It is based on the GPS range within which the source labels in the respective scene are located. While this generates satisfactory results for scenes that contain multiple, distributed labels, some scenes showed to have a narrow range in which the source labels are located. For 42 scenes no nearby images have been sampled, reducing the number of scenes for which labels can be generated to 72. Investigation revealed that for 16 of these 42 scenes no images have been sampled due to the source scene containing only one image, meaning that all labels lie in an area of 0m^2. No images are sampled in this case, as the current implementation samples images only inside the area in which source labels are located. Improving the sampling mechanism, for example,

by sampling in a wider area for these scenes, could therefore potentially increase the number of generated labels.

- In the Image Registration step, the current implementation ignores scenes with more than 200 images to be registered, as these scenes would considerably increase the workload and lead to longer generation times. In the given case, this reduces the number of scenes used for labeling by 4, to 68. Choosing a different method to sample unlabeled images could avoid this reduction, for example, by sampling only every second image in these cases.
- The label generation method relies on features that lie inside of a bounding box of an object in the source image and that are observed in the target image. For the used labeled data, no feature points lie inside of one of the source bounding boxes in 9 scenes, meaning that they cannot assist in generating labels for any unlabeled images matched to them. Furthermore, for the used unlabeled data from 21.01.2022, for 24 scenes, no shared features are contained in the extended reconstruction. Only this would allow to generate labels based on them. This can have various reasons, such as the object not being visible in the unlabeled data or the appearance shifting too much between domains to find similar feature points. As a result, for this data, only 35 scenes can be used for label generation. Adapting the reconstruction process might result in more shared features, for example, by using a different feature detector than SIFT. Using a different method to create the labels could also potentially improve how many labels are generated, for example, the method could directly use information from the 3D points and 3D camera locations to generate labels.
- To obtain high quality labels, Section 4.2 presents methods and choices of parameters to increase label quality at the cost of reducing the number of generated labels. For example, requiring the target image to contain as many annotated objects as the source image reduces the number of scenes used for label generation by 7 to 28. Setting the required number of shared feature points to 3 further reduces it by 16 scenes to 12 scenes.
- Each scene can potentially produce multiple labels of different objects, with different perspectives and at different times of day. With the method Extended Sampling presented in Section 3.3.4, out of the 12 scenes, a total of 15 labels were generated.

Figure 5.1 shows an overview of the main causes why only a small number of the potential labels are generated. Multiple sets of unlabeled data can be used to generate additional labels, for example, from different days or conditions. While these labels might contain the same objects, varying conditions between data can lead to these images having diversity and therefore can make this data useful for model training. This can be especially useful to generate labels from rarely occurring events.

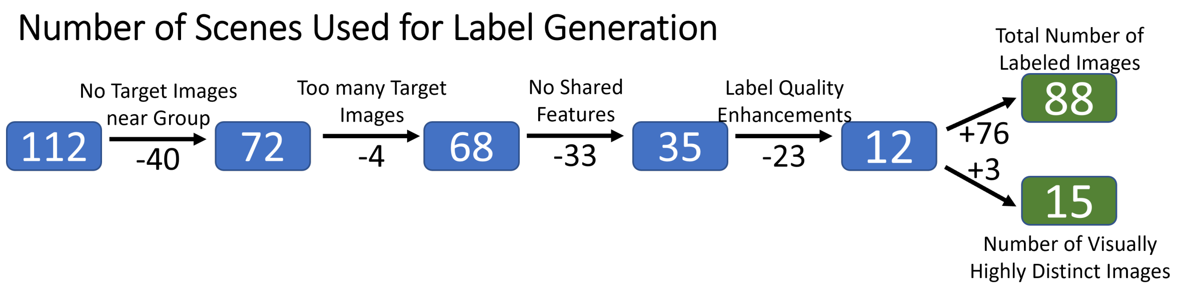

Number of Scenes Used for Label Generation
112
No Target Images near Group
-40
72
Too many Target Images
-4
68
No Shared Features
-33
35
Label Quality Enhancements
-23
12
+76
+3
Total Number of Labeled Images
88
15
Number of Visually Highly Distinct Images

Figure 5.1: **Number of scenes used for label generation**: The initial 204 labeled images get grouped into 112 scenes. Due to the listed reasons, only 12 of these scenes can be used for label generation, for the data from the bus from 21.01.2022. As each scene can contain images of multiple trash cans and from varying angles and times of day, a total of 88 labeled images could be obtained, or 15 labeled images that are visually highly distinct.

6 Conclusion

6.1 Summary

In this thesis a novel method for automatic label generation has been designed, implemented, and tested. The focus was on automated label generation in new domains where labeled images are unavailable. This was achieved by locating the position of objects that are labeled in the source domain and observed in both the source and target domains. The effectiveness of the developed method has been demonstrated using data from the BusEdge system, a transit bus equipped with cameras, and a GPS and IMU sensor. Using a dataset that contains static trash cans in sunny conditions, the method was applied to automatically generate labels for new domains, such as cloudy or snowy conditions. The demonstration proves that it is indeed possible to generate labels for new domains by revisiting labeled objects. The analysis in Section 4.2 reveals optimal parameters to generate the highest amount of labels given a required level of accuracy. Remarkable improvements have been shown by incorporating these automatically generated labels into the training process of Object Detection models. Notably, on the datasets that contain images in the cloudy domain, an increase of up to +4.1AP was achieved on the $\text{Cloudy}_{\text{Test}}$ set and +6.4AP on the RoadBotics based dataset. In the snowy domain, improvements of up to +1AP were obtained on the $\text{Snowy}_{\text{Day}}$ dataset and +3.1AP on the $\text{Snowy}_{\text{Night}}$ dataset.

These remarkable improvements highlight the potential of the presented method to benefit many applications of Computer Vision that involve detecting static objects. Moreover, by cropping the labeled images to focus solely on the detected objects, the method can be extended to other areas of Computer Vision, such as Object Classification.

6.2 Outlook

The results of the evaluation show remarkable improvements when using the automatically labeled images in the training process. However, to fully exploit the potential of the proposed method larger datasets need to be constructed. The discussion in Section 5 shows that many parts of the pipeline can be optimized. Optimizing these parts will allow to generate more labels with higher diversity. Furthermore, performance optimizations such as more efficient selection of unlabeled images might allow the pipeline to work more efficiently and reduce computational workload. Generating more labeled data and using bigger test sets will allow to make better statements about how much our method can improve Domain Adaptation in different use cases. It will be especially interesting to apply this method to other domains and objects.

List of Figures

List of Tables

Acronyms

AL Active Learning.

AP COCO mean Average Precision.

BA Bundle Adjustment.

CNN Convolutional Neural Network.

Conv Fully Connected Layer.

DL Deep Learning.

EMA Exponential Moving Average.

FN False Negative.

FP False Positve.

FPN Feature Pyramid Network.

GAN Generative Adversarial Networks.

IMU Inertial Measurement Unit.

IoU Intersection over Union.

RANSAC Random sampling and consensus.

RoIs Regions of Interest.

RPN Region Proposal Network.

SfM Structure for Motion.

SLAM Simulatanous Localization and Mapping.

SOTA State of the Art.

TP True Positive.

Bibliography

[1] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: `1405.0312 [cs.CV]`.

[2] A. Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: `10.1109/CVPR.2009.5206848`.

[4] C. Ye, J. Dolan, R. Iyengar, J. Harkes, J. Wang, E. Ziqiang, Fang, H. Turki, J. Chiang, X. Zhang, Y.-C. Kuo, Y.-T. Lin, and C. Ofodike. "BusEdge: Efficient Live Video Analytics for Transit Buses via Edge Computing". In: 2021.

[5] A. RoyChowdhury, P. Chakrabarty, A. Singh, S. Jin, H. Jiang, L. Cao, and E. Learned-Miller. "Automatic Adaptation of Object Detectors to New Domains Using Self-Training". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[6] K. Tang, V. Ramanathan, L. Fei-fei, and D. Koller. "Shifting Weights: Adapting Object Detectors from Image to Video". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. Burges, L. Bottou, and K. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: `https://proceedings.neurips.cc/paper/2012/file/26e359e83860db1d11b6acca57d8ea88-Paper.pdf`.

[7] S. Walsh, J. Ku, A. D. Pon, and S. L. Waslander. "Leveraging Temporal Data for Automatic Labelling of Static Vehicles". In: *2020 17th Conference on Computer and Robot Vision (CRV)*. 2020, pp. 134–141. DOI: `10.1109/CRV50864.2020.00026`.

[8] RoadBotics. *RoadBotics Open Data Set*. `https://www.roadbotics.com/2021/03/15/roadbotics-open-data-set/`. Accessed on 23.01.2023.

[9] H. Yu, I. Mineyev, L. R. Varshney, and J. A. Evans. *Learning from One and Only One Shot*. 2022. arXiv: `2201.08815 [cs.CV]`.

[10] S. Dong, P. Wang, and K. Abbas. “A survey on deep learning and its applications”. In: *Computer Science Review* 40 (2021), p. 100379. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2021.100379. URL: https://www.sciencedirect.com/science/article/pii/S1574013721000198.

[11] P. Viola and M. Jones. “Rapid object detection using a boosted cascade of simple features”. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990517.

[12] N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[13] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[14] S. Ren, K. He, R. Girshick, and J. Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’15. Montreal, Canada: MIT Press, 2015, pp. 91–99.

[15] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. *End-to-End Object Detection with Transformers*. 2020. arXiv: 2005.12872 [cs.CV].

[16] G. Jocher, A. Chaurasia, and J. Qiu. *YOLO by Ultralytics*. Version 8.0.0. Jan. 2023. URL: https://github.com/ultralytics/ultralytics.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV].

[18] R. Girshick. “Fast R-CNN”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[19] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. In: *arXiv preprint arXiv:1512.03385* (2015).

[20] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[21] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV].

[22] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. "Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: `1703.06870`. URL: `http://arxiv.org/abs/1703.06870`.

[23] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. "Feature Pyramid Networks for Object Detection". In: *CoRR* abs/1612.03144 (2016). arXiv: `1612.03144`. URL: `http://arxiv.org/abs/1612.03144`.

[24] J. Li, R. Zhang, Y. Liu, Z. Zhang, R. Fan, and W. Liu. "The Method of Static Semantic Map Construction Based on Instance Segmentation and Dynamic Point Elimination". In: *Electronics* 10 (Aug. 2021), p. 1883. DOI: `10.3390/electronics10161883`.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

[26] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: `2103.14030 [cs.CV]`.

[27] Z. Zong, G. Song, and Y. Liu. *DETRs with Collaborative Hybrid Assignments Training*. 2023. arXiv: `2211.12860 [cs.CV]`.

[28] Q. Chen, J. Wang, C. Han, S. Zhang, Z. Li, X. Chen, J. Chen, X. Wang, S. Han, G. Zhang, H. Feng, K. Yao, J. Han, E. Ding, and J. Wang. *Group DETR v2: Strong Object Detector with Encoder-Decoder Pretraining*. 2022. arXiv: `2211.03594 [cs.CV]`.

[29] Y. Wei, H. Hu, Z. Xie, Z. Zhang, Y. Cao, J. Bao, D. Chen, and B. Guo. *Contrastive Learning Rivals Masked Image Modeling in Fine-tuning via Feature Distillation*. 2022. arXiv: `2205.14141 [cs.CV]`.

[30] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar. *Masked-attention Mask Transformer for Universal Image Segmentation*. 2022. arXiv: `2112.01527 [cs.CV]`.

[31] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee. "A survey of modern deep learning based object detection models". In: *Digital Signal Processing* 126 (2022), p. 103514. ISSN: 1051-2004. DOI: `https://doi.org/10.1016/j.dsp.2022.103514`. URL: `https://www.sciencedirect.com/science/article/pii/S1051200422001312`.

[32] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye. "Object Detection in 20 Years: A Survey". In: *Proceedings of the IEEE* 111.3 (2023), pp. 257–276. DOI: `10.1109/JPROC.2023.3238524`.

[33] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. *Detectron2*. `https://github.com/facebookresearch/detectron2`. 2019.

[34] P. Oza, V. A. Sindagi, V. VS, and V. M. Patel. *Unsupervised Domain Adaptation of Object Detectors: A Survey*. 2021. arXiv: 2105.13502 [cs.CV].

[35] M. Chen, K. Q. Weinberger, and J. Blitzer. "Co-Training for Domain Adaptation". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper_files/paper/2011/file/93fb9d4b16aa750c7475b6d601c35c2c-Paper.pdf.

[36] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. *Semi-Supervised Learning with Deep Generative Models*. 2014. arXiv: 1406.5298 [cs.LG].

[37] J. E. van Engelen and H. H. Hoos. "A survey on semi-supervised learning". In: *Machine Learning* 109.2 (Feb. 2020), pp. 373–440. ISSN: 1573-0565. DOI: 10.1007/s10994-019-05855-6. URL: https://doi.org/10.1007/s10994-019-05855-6.

[38] D. Zhang, J. Han, G. Cheng, and M.-H. Yang. *Weakly Supervised Object Localization and Detection: A Survey*. 2021. arXiv: 2104.07918 [cs.CV].

[39] X. Huang and S. Belongie. *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization*. 2017. arXiv: 1703.06868 [cs.CV].

[40] Q. Cai, Y. Pan, C.-W. Ngo, X. Tian, L. Duan, and T. Yao. *Exploring Object Relation in Mean Teacher for Cross-Domain Detection*. 2019. arXiv: 1904.11245 [cs.CV].

[41] Y. Gandelsman, A. Shocher, and M. Irani. ""Double-DIP": Unsupervised Image Decomposition via Coupled Deep-Image-Priors". In: June 2019, pp. 11018–11027. DOI: 10.1109/CVPR.2019.01128.

[42] D. Berman, T. treibitz, and S. Avidan. "Non-Local Image Dehazing". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[43] R. Fattal. "Single Image Dehazing". In: *ACM Trans. Graph.* 27.3 (Aug. 2008), pp. 1–9. ISSN: 0730-0301. DOI: 10.1145/1360612.1360671. URL: https://doi.org/10.1145/1360612.1360671.

[44] Y. Qu, Y. Chen, J. Huang, and Y. Xie. "Enhanced Pix2pix Dehazing Network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[45] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari. *Training object class detectors with click supervision*. 2017. DOI: 10.48550/ARXIV.1704.06189. URL: https://arxiv.org/abs/1704.06189.

[46] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[47] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3234–3243. DOI: `10.1109/CVPR.2016.352`.

[48] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan. *Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks?* 2016. DOI: `10.48550/ARXIV.1610.01983`. URL: `https://arxiv.org/abs/1610.01983`.

[49] D. Dwibedi, I. Misra, and M. Hebert. *Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection*. 2017. DOI: `10.48550/ARXIV.1708.01642`. URL: `https://arxiv.org/abs/1708.01642`.

[50] T. Bu, X. Zhang, C. Mertz, and J. M. Dolan. "CARLA Simulated Data for Rare Road Object Detection". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 2794–2801. DOI: `10.1109/ITSC48978.2021.9564932`.

[51] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang. "A Survey of Deep Active Learning". In: *ACM Comput. Surv.* 54.9 (Oct. 2021). ISSN: 0360-0300. DOI: `10.1145/3472291`. URL: `https://doi.org/10.1145/3472291`.

[52] S. Kadam and V. Vaidya. "Review and Analysis of Zero, One and Few Shot Learning Approaches". In: *Intelligent Systems Design and Applications*. Ed. by A. Abraham, A. K. Cherukuri, P. Melin, and N. Gandhi. Cham: Springer International Publishing, 2020, pp. 100–112. ISBN: 978-3-030-16657-1.

[53] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. *A Comprehensive Survey on Transfer Learning*. 2020. arXiv: `1911.02685 [cs.LG]`.

[54] P. F. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi. "Street-view change detection with deconvolutional networks". In: *Autonomous Robots* 42.7 (Oct. 2018), pp. 1301–1322. ISSN: 1573-7527. DOI: `10.1007/s10514-018-9734-5`. URL: `https://doi.org/10.1007/s10514-018-9734-5`.

[55] H. Badino, D. Huber, and T. Kanade. "Visual topometric localization". In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. 2011, pp. 794–799. DOI: `10.1109/IVS.2011.5940504`.

[56] H. Badino, D. Huber, and T. Kanade. "Real-time topometric localization". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 1635–1642. DOI: `10.1109/ICRA.2012.6224716`.

[57] P. Fernández Alcantarilla.
"Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces". In: Sept. 2013. DOI: `10.5244/C.27.13`.

[58] M. Kaess, K. Ni, and F. Dellaert. "Flow separation for fast and robust stereo odometry". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 3539–3544. DOI: `10.1109/ROBOT.2009.5152333`.

[59] E. Tola, V. Lepetit, and P. Fua.
"DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo".
In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.5 (2010), pp. 815–830. DOI: `10.1109/TPAMI.2009.77`.

[60] M. Van den Bergh, X. Boix, G. Roig, and L. Van Gool.
"SEEDS: Superpixels Extracted Via Energy-Driven Sampling".
In: *International Journal of Computer Vision* 111.3 (Feb. 2015), pp. 298–314. ISSN: 1573-1405. DOI: `10.1007/s11263-014-0744-2`.
URL: `https://doi.org/10.1007/s11263-014-0744-2`.

[61] D. Min, S. Choi, J. Lu, B. Ham, K. Sohn, and M. N. Do.
"Fast Global Image Smoothing Based on Weighted Least Squares".
In: *IEEE Transactions on Image Processing* 23.12 (2014), pp. 5638–5653.
DOI: `10.1109/TIP.2014.2366600`.

[62] J. Long, E. Shelhamer, and T. Darrell.
"Fully convolutional networks for semantic segmentation".
In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
Los Alamitos, CA, USA: IEEE Computer Society, June 2015, pp. 3431–3440.
DOI: `10.1109/CVPR.2015.7298965`.
URL: `https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298965`.

[63] X. Ding, Y. Wang, R. Xiong, D. Li, L. Tang, H. Yin, and L. Zhao.
"Persistent Stereo Visual Localization on Cross-Modal Invariant Map".
In: *IEEE Transactions on Intelligent Transportation Systems* 21.11 (2020), pp. 4646–4658. DOI: `10.1109/TITS.2019.2942760`.

[64] M. Zhang, Y. Chen, and M. Li. "Vision-Aided Localization For Ground Robots".
In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 2455–2461. DOI: `10.1109/IROS40897.2019.8968521`.

[65] D. Kiss-Illés, C. Barrado, and E. Salamí.
"GPS-SLAM: An Augmentation of the ORB-SLAM Algorithm". In: *Sensors* 19.22 (2019). ISSN: 1424-8220. DOI: `10.3390/s19224973`.
URL: `https://www.mdpi.com/1424-8220/19/22/4973`.

[66] M. Servières, V. Renaudin, A. Dupuis, and N. Antigny. "Visual and Visual-Inertial SLAM: State of the Art, Classification, and Experimental Benchmarking".
In: *Journal of Sensors* 2021 (Feb. 2021), p. 2054828. ISSN: 1687-725X.
DOI: `10.1155/2021/2054828`. URL: `https://doi.org/10.1155/2021/2054828`.

[67] T. Qin, P. Li, and S. Shen.
"VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator".
In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.
DOI: 10.1109/TRO.2018.2853729.

[68] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart.
"Robust Visual Inertial Odometry Using a Direct EKF-Based Approach". en. In:
2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS);
Conference Location: Hamburg, Germany; Conference Date: September 28 - October 2,
2015. Zürich: ETH-Zürich, 2015. DOI: 10.3929/ethz-a-010566547.

[69] R. Mur-Artal and J. D. Tardós. "ORB-SLAM2: An Open-Source SLAM System for
Monocular, Stereo, and RGB-D Cameras".
In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262.
DOI: 10.1109/TRO.2017.2705103.

[70] J. Engel, V. Koltun, and D. Cremers. *Direct Sparse Odometry*. 2016.
DOI: 10.48550/ARXIV.1607.02565. URL: https://arxiv.org/abs/1607.02565.

[71] J. Engel, T. Schöps, and D. Cremers.
"LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *Computer Vision – ECCV 2014*.
Ed. by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars.
Cham: Springer International Publishing, 2014, pp. 834–849. ISBN: 978-3-319-10605-2.

[72] P. Fu, S. Li, T. Yu, X. Zhu, W. Yao, and X. Chen. "An Initialization Method for
Monocular Visual-Inertial Odometry Based on Uncertainty State".
In: *2022 4th International Conference on Communications, Information System and Computer
Engineering (CISCE)*. 2022, pp. 522–528. DOI: 10.1109/CISCE55963.2022.9851127.

[73] L. Hu, L. Sun, Y. Wang, K. Yue, Z. Li, and Z. Yan.
"Online Photometric Calibration of Optical Flow Visual-Inertial SLAM System".
In: *2020 12th International Conference on Communication Software and Networks (ICCSN)*.
2020, pp. 13–17. DOI: 10.1109/ICCSN49894.2020.9139073.

[74] X. Gao, R. Wang, N. Demmel, and D. Cremers.
"LDSO: Direct Sparse Odometry with Loop Closure".
In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018,
pp. 2198–2204. DOI: 10.1109/IROS.2018.8593376.

[75] P. Yin, A. Abuduweili, S. Zhao, C. Liu, and S. Scherer.
BioSLAM: A Bio-inspired Lifelong Memory System for General Place Recognition. 2022.
DOI: 10.48550/ARXIV.2208.14543. URL: https://arxiv.org/abs/2208.14543.

[76] M. Zhang, X. Xu, Y. Chen, and M. Li. "A Lightweight and Accurate Localization
Algorithm Using Multiple Inertial Measurement Units".
In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1508–1515.
DOI: 10.1109/LRA.2020.2969146.

[77] M. Bürki, L. Schaupp, M. Dymczyk, R. Dubé, C. Cadena, R. Siegwart, and J. Nieto. "Vizard: Reliable visual localization for autonomous vehicles in urban outdoor environments". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1124–1130.

[78] W. Wen, X. Bai, G. Zhang, S. Chen, F. Yuan, and L.-T. Hsu. "Multi-Agent Collaborative GNSS/Camera/INS Integration Aided by Inter-Ranging for Vehicular Navigation in Urban Areas". In: *IEEE Access* 8 (2020), pp. 124323–124338. DOI: `10.1109/ACCESS.2020.3006210`.

[79] R. Dubois, A. Eudes, and V. Frémont. "Sharing visual-inertial data for collaborative decentralized simultaneous localization and mapping". In: *Robotics and Autonomous Systems* 148 (2022), p. 103933. ISSN: 0921-8890. DOI: `https://doi.org/10.1016/j.robot.2021.103933`. URL: `https://www.sciencedirect.com/science/article/pii/S0921889021002177`.

[80] W. Wen, X. Bai, G. Zhang, S. Chen, F. Yuan, and L.-T. Hsu. "Multi-Agent Collaborative GNSS/Camera/INS Integration Aided by Inter-Ranging for Vehicular Navigation in Urban Areas". In: *IEEE Access* 8 (2020), pp. 124323–124338. DOI: `10.1109/ACCESS.2020.3006210`.

[81] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart. "Maplab: An Open Framework for Research in Visual-Inertial Mapping and Localization". In: *IEEE Robotics and Automation Letters* 3.3 (July 2018), pp. 1418–1425. DOI: `10.1109/lra.2018.2800113`. URL: `https://doi.org/10.1109%5C%2Flra.2018.2800113`.

[82] D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 1573-1405. DOI: `10.1023/B:VISI.0000029664.99615.94`. URL: `https://doi.org/10.1023/B:VISI.0000029664.99615.94`.

[83] H. Bay, T. Tuytelaars, and L. Van Gool. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Ed. by A. Leonardis, H. Bischof, and A. Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

[84] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: `10.1109/ICCV.2011.6126544`.

[85] M. A. Fischler and R. C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Readings in Computer Vision*. Ed. by M. A. Fischler and O. Firschein. San Francisco (CA): Morgan Kaufmann, 1987, pp. 726–740. ISBN: 978-0-08-051581-6.

DOI: https://doi.org/10.1016/B978-0-08-051581-6.50070-2. URL: https://www.sciencedirect.com/science/article/pii/B9780080515816500702.

[86] J. L. Schönberger and J.-M. Frahm. "Structure-from-Motion Revisited". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[87] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. "Pixelwise View Selection for Unstructured Multi-View Stereo". In: *European Conference on Computer Vision (ECCV)*. 2016.

[88] M. Kazhdan and H. Hoppe. "Screened Poisson Surface Reconstruction". In: *ACM Trans. Graph.* 32.3 (July 2013). ISSN: 0730-0301. DOI: 10.1145/2487228.2487237. URL: https://doi.org/10.1145/2487228.2487237.

[89] E. Rotondo and C. Mertz. "Detecting and Classifying Waste Bin Garbage Levels Along Transit Bus Routes". In: *RISS Working Journal* (2021).

[90] T. Storm and C. Mertz. *Detecting and Classifying Bus Stop Trash Cans*. 2022.

[91] E. Rotondo and C. Mertz. *Waste Bin Detection Dataset (RISS2021)*. https://www.kaggle.com/datasets/elirotondo/waste-bin-detection-dataset-riss-2021. Accessed on 23.01.2023.

[92] J. L. Schönberger. *COLMAP Tutorial*. https://colmap.github.io/tutorial.html. Accessed on 01.05.2023.

[93] D. Gong, J. Yang, L. Liu, Y. Zhang, I. Reid, C. Shen, A. van den Hengel, and Q. Shi. "From Motion Blur to Motion Flow: A Deep Learning Solution for Removing Heterogeneous Motion Blur". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.