Downstream Task Self-Supervised Learning for Object Recognition and Tracking

Abubakar Siddique
Marquette University

DOWNSTREAM TASK SELF-SUPERVISED LEARNING FOR OBJECT RECOGNITION AND TRACKING

by

Abubakar Siddique

A Dissertation submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy

Milwaukee, Wisconsin

May 2023

ABSTRACT
DOWNSTREAM TASK SELF-SUPERVISED LEARNING FOR OBJECT RECOGNITION AND TRACKING

Abubakar Siddique

Marquette University, 2023

This dissertation addresses three limitations of deep learning methods in image and video understanding-based machine vision applications. Firstly, although deep convolutional neural networks (CNNs) are efficient for image recognition applications such as object detection and segmentation, they perform poorly under perspective distortions. In real-world applications, the camera perspective is a common problem that we can address by annotating large amounts of data, thus limiting the applicability of the deep learning models. Secondly, the typical approach for single-camera tracking problems is to use separate motion and appearance models, which are expensive in terms of computations and training data requirements. Finally, conventional multi-camera video understanding techniques use supervised learning algorithms to determine temporal relationships among objects. In large-scale applications, these methods are also limited by the requirement of extensive manually annotated data and computational resources.

To address these limitations, we develop an uncertainty-aware self-supervised learning (SSL) technique that captures a model's instance or semantic segmentation uncertainty from overhead images and guides the model to learn the impact of the new perspective on object appearance. The test-time data augmentation-based pseudo-label refinement technique continuously trains a model until convergence on new perspective images. The proposed method can be applied for both self-supervision and semi-supervision, thus increasing the effectiveness of a deep pre-trained model in new domains. Extensive experiments demonstrate the effectiveness of the SSL technique in both object detection and semantic segmentation problems.

In video understanding applications, we introduce simultaneous segmentation and tracking as an unsupervised spatio-temporal latent feature clustering problem. The jointly learned multi-task features leverage the task-dependent uncertainty to generate discriminative features in multi-object videos. Experiments have shown that the proposed tracker outperforms several state-of-the-art supervised methods. Finally, we proposed an unsupervised multi-camera tracklet association (MCTA) algorithm to track multiple objects in real-time. MCTA leverages the self-supervised detector model for single-camera tracking and solves the multi-camera tracking problem using multiple pair-wise camera associations modeled as a connected graph. The graph optimization method generates a global solution for partially or fully overlapping camera networks.

ACKNOWLEDGMENTS

Abubakar Siddique

I would like to express my sincere gratitude to my advisor Dr. Henry Medeiros, whose guidance and support throughout my dissertation journey have been invaluable. His expertise, encouragement, and patience have been a constant source of inspiration and motivation. I would also like to thank my second supervisor, Dr. Majeed Hayat, for his valuable feedback and insights in every situation in the last two years of my graduate life. Their contributions have greatly enriched my research and helped shape my dissertation into its final form.

I am grateful to my family and friends especially my mother Anwara Begum, my father Montaz Miah, and my wife Nasrin Sultana for their unwavering support throughout my academic journey. Their love and encouragement kept me going during difficult times. Finally, I thank all the lab mates Philipe, Reza, Sam, Yev, Xie, Scott, and Jamir who generously shared their time and experiences with me. Without their contributions, this dissertation would not have been possible. Thank you all for helping me achieve this important milestone in my academic career.

Finally, I would like to thank the Graduate School and all of the Marquette University administration especially our Office of International Education staff Susan Whipple, Michael Groen, and our department office staff Ms. Katie Tarara for their great support throughout the entire doctoral program time.

This dissertation includes material that has been previously published in three peer-reviewed papers where I was the first author. The research that forms the foundation of this dissertation was directed and supervised by Dr. Henry Medeiros, who is also listed as a co-author in all three publications.

DEDICATION

I would like to dedicate this work to my ever-supporting family and friends, especially my mother, Anwara Begum, my father Montaz Miah, my wife, Nasrin Sultana, and lovely daughter Alveera Siddiqa, who helps in every difficulty of my life. I would also like to dedicate this dissertation to my supervisors, Dr. Henry Medeiros and Dr. Rezaul Karim Mazumder, for their encouraging motivations in different stages of my life. Finally, I dedicate this achievement to my sisters, especially my elder sister, who was my first school guardian and helped me to continue school under challenging situations.

Thank You

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Image and video understanding are fundamental problems in computer vision applications that aim to enable machines to interpret and make sense of visual data like humans. These problems involve recognizing objects, people, actions, events, and scenes within images and videos. Image recognition tasks as shown in Fig. 1.1-(a) include object detection [1], instance [7], semantic [8], or panoptic segmentation [9] to localize an object, recognize its boundaries, and predict the pixel category for everything in an image frame. Pose and depth estimation techniques [10] predict the key points/poses and depth for objects in an image. The typical challenges

Figure 1.1: Examples of image/video recognition tasks in a multi-view WILDTRACK [11] video datasets. (a) classification, detection, and instance segmentation, (b)-(c) single-camera tracking and two-camera temporal association.

of image understanding tasks are the vast variability in object appearance, pose, scale, and the presence of clutter and occlusions in the image. On the other hand, the video understanding problem involves the temporal connection of predictions on individual images. For example, the multiple-object tracking [12] establishes the

temporal relationship among objects, and cross-view data association [11] connects multi-view information in a video sequence. Fig. 1.1 (b)-(c) show an example of a video understanding task where the temporal association across two cameras uses individual image recognition results and maintains unique target identities across the views.

Modern machine-learning techniques address these tasks using large amounts of labeled data. However, image and video understanding techniques to solve complex, real-world tasks without leveraging manual labels or using a few manual labels are a challenging problem in modern computer vision applications. Popular approaches to tackle the necessity for large datasets include: i) self-supervised learning (SSL) [13], semi-supervised learning (Semi-SL) for object detection [14] and instance segmentation tasks [15] in challenging overhead perspectives, ii) SSL for generalizing panoptic segmentation [16] tasks across challenging datasets, iii) uncertainty-aware multi-task learning for spatio-temporal discriminative embeddings to solve multi-object tracking and segmentation [17], and iv) unsupervised multi-view association for video surveillance [18] and 3D pose estimation [11].

Using manual labels is one popular strategy for designing deep convolutional neural network (CNN) models for specific tasks. However, this strategy is more challenging when the data domain changes frequently over time and locations, as is the case for computer vision applications, such as video surveillance, self-driving vehicles, or agricultural automation. The limitations of existing fully supervised works in these applications motivate us to develop self-supervised/unsupervised learning approaches to solve image/video understanding problems. One of the most popular techniques to address the domain change problem is SSL, which adapts a model to make predictions in a new domain without accessing manually generated labels. In image/video recognition problems, multi-task learning is another strategy where the data or task-dependent uncertainties help a supervised [10] or unsupervised model

[17] to converge better in comparison with a single-task strategy.

The ultimate goal of image/video understanding algorithms is to solve higher-level problems (a.k.a. downstream tasks), such as multiple-object tracking using multiple cameras [19]. Still, these algorithms may also require extensive manual annotations. This dissertation aims to solve image understanding problems, such as multi-object detection and segmentation, and video understanding problems, such as multi-view tracking and association problems, in a self-supervised/unsupervised way. A pre-trained model based on publicly available datasets must be sufficiently robust to solve a similar situation in new challenging datasets. Domain adaptation methods [20] have become popular in addressing this issue. SSL [21] and Semi-SL algorithms [22] are widely used to transfer model knowledge to new data domains instead of training the model from scratch using expensive manual annotations. These domain adaptation methods are currently used to solve challenging downstream tasks and backbone feature learning.

1.1 Problem Statements

The main contributions of this dissertation are novel learning techniques to mitigate the need for extensive manual annotations. More specifically, it addresses the following five problems.

1.1.1 Problem statement #1

For downstream tasks, dataset-specific pre-trained convolutional neural networks perform poorly in unfamiliar camera perspectives. In proposal-based object detectors [1, 6], multi-scale features from a backbone network are used to solve downstream tasks by leveraging supervised learning techniques. Hence, CNN-based detectors struggle to predict previously learned target categories under unfamiliar camera perspectives. Accurate localization and precise shape prediction are

essential for high-level decision-making problems in crowded surveillance applications. For monocular [23] or multi-view tracking [11] applications, the challenges of overhead perspectives, partial visibility, variations in camera angles in different camera networks [18] degrade the consistency in the association of temporal information. We propose a data augmentation technique where multiple inferences of the same input frame generate a new augmented target distribution and estimate the unknown locations of possible targets. Objectives 2 and 3 explain how these augmented detection distributions address the problems introduced by unfamiliar overhead camera perspectives using a new self-supervised technique.

Most deep learning-based detection [14] and segmentation models [7] perform with high accuracy when trained using large manually annotated datasets [24]. Modern deep learning models leverage data augmentation to learn more examples during training and perform transfer learning into unseen domains. However, most augmentation techniques improve backbone feature learning, which only indirectly enhances the performance of downstream tasks. Although simple geometric transformations can improve downstream task predictions, a systematic approach to guide the deep model from transformation-based augmentation is rare. Hence, in Objectives 2 and 3, we explain how we can effectively transfer knowledge from deep CNN model learning to solve the object detection problem in unfamiliar datasets.

1.1.2 Problem statement #2

In multi-view scene understanding, detection or segmentation models need a significant amount of human annotations to learn about different perspectives or domains. Due to the change of perspective challenge, pre-trained models [7] trained on publicly available large-scale datasets [24] have difficulty detecting partially occluded or small objects. In most computer vision applications, the typical approach is to use human-annotated labels to fine-tune the model for new

data domains. However, this approach relies on the tedious and expensive human annotation procedure and deployment-specific training data. For example, the dramatic variability in video surveillance systems and the dependency on camera-specific adjustments require deployment-specific fine-tuning of surveillance models using manual labels. Obtaining these expensive manual labels is currently the main barrier to the widespread adoption of models based on deep neural networks for such applications. To overcome this challenge, we intend to use the algorithm proposed in Objective 1 to devise a novel self-supervised algorithm that updates the model using automatically generated pseudo-labels and the network prediction uncertainties on unseen data.

1.1.3 Problem statement #3

For semantic prediction, transfer learning is challenging when the data domain is unknown to the pre-trained model. For semantic or instance segmentation applications [7, 9], a common approach is to use manual labels to train a deep CNN. To achieve high-quality predictions from supervised models, sometimes computationally expensive post-processing algorithms [25] are necessary. However, such models cannot generalize for substantially different datasets, especially in the presence of varying illumination, camera perspective, or background clutter. We intend to devise an SSL method using automatically generated augmented semantic pseudo-labels, which increase the model's sensitivity to objects of interest. The proposed active learning strategy (Objective 3) reduces the semantic uncertainties for challenging datasets and mitigates the semantic labeling cost. The proposed method can either avoid the costly post-processing approach without degrading performance or incorporate post-processing algorithms to further improve prediction accuracy.

1.1.4 Problem statement #4

Conventional multi-object segmentation and tracking approaches leverage supervised learning and resort to separate motion and appearance models to perform association, but unsupervised jointly learned multi-task models can improve tracking consistency. Multi-object segmentation and tracking algorithms [26, 12] usually employ a supervised learning technique to generate discriminative embedding features and then apply an association technique based on sophisticated target motion models. The dependency on human annotations to train such models limits the applicability of existing methods to real-world problems. Again, the supervised learning of individual motion and appearance models fails to yield satisfactory tracking consistency [27]. However, for many real-world applications, task-dependent uncertainty-aware learning of joint spatio-temporal embeddings makes it possible to use unsupervised clustering to perform data association. Spatio-temporal embeddings increase the multiple-object tracking system's performance in challenging scenarios where target appearances and motions change abruptly with time. To address these challenges, we propose a new task-dependent, uncertainty-aware joint embedding learning technique that eliminates the need for expensive manual labels. To perform temporal association, we propose a novel unsupervised spatio-temporal clustering algorithm (Objective 4) that leverages domain knowledge as a constraint graph. This algorithm can be easily applied with any pre-trained multi-task feature extractor, especially in scenarios where multiple objects have similar motion and appearance patterns.

1.1.5 Problem statement #5

For multi-camera networks, real-time multi-object tracking is challenging, as modern deep learning architectures have high computational

requirements when performing detections in high-resolution complex visual data. For tracking multiple objects in a multi-camera network, a typical strategy [28, 11] is to use a single camera tracker (SCT) and then perform multi-camera association either in 2D or 3D space. Even though appearance features are widely employed to perform association across cameras, in overhead camera views, cross-camera matching becomes challenging, and computational complexity increases accordingly. To address these limitations in multi-camera tracking, we intend to devise a framework (Objective 5) to perform real-time single-camera detection, tracking, and multi-camera association. Optimizing memory resources and parallelizing the internal detection and tracking processes makes it possible to process the multi-camera surveillance task in real time. Parallel computing strategies, optimized data, and network weight representations make it possible to maintain the overall real-time multi-camera tracking performance.

1.2 Objectives

In this dissertation, we address five key challenges: i) perspective variations in multi-object detection and segmentation, ii) semi-supervised/self-supervised/unsupervised learning for multiple object detection, segmentation, and tracking, iii) SSL for semantic and panoptic segmentation, iv) unsupervised multi-view association, v) real-time multi-camera multi-object tracking in image/video understanding problems. These challenges are captured in the five research objectives described below.

Objective 1: Develop a test-time data augmentation algorithm for enhancing the performance of region proposal-based detectors.

This objective aims to improve the performance of state-of-the-art object detection and segmentation algorithms by increasing their sensitivity to targets of interest. Current detectors and post-processing algorithms cannot detect targets under

substantial perspective distortions. Hence, our goal is to develop a multiple inference-based clustering/voting algorithm to identify the targets in geometrically distorted scenarios without resorting to additional training in a manner that applies to any detector as a robust post-processing algorithm. This unsupervised clustering-based data augmentation algorithm also reduces false detections. It motivates us to devise a self-supervised algorithm that uses the outcome of Objective 1 as a high-quality pseudo-label generation step during self-supervised training.

Objective 2: Devise a self-supervised learning technique to overcome the dependency on human annotations in multi-view scene understanding.

Detectors based on deep CNNs are usually trained using labeled data to solve the detection problem in specific applications. The model needs to be fine-tuned using newly labeled data to be deployed in a new application or camera network. However, the manual labeling task is expensive and tedious for large-scale applications. To address the labeling cost and transfer learning challenges in a new domain or unseen data distributions, we devise an SSL algorithm that uses automatically generated pseudo-labels to update the model. The ultimate goal of SSL is to transfer the learning from one domain to another in real-world applications simply by using the unlabeled data and the initial model weights.

Objective 3: Devise a self-supervised learning strategy for semantic segmentation.

We explore the effectiveness of our self-supervised approach in semantic segmentation without relying on extensive labeled data and computationally expensive post-processing approaches. Our goal is to robustly transfer knowledge from the initial model to any challenging dataset without performing the tedious semantic labeling work and generalizing the learning for any dataset.

Objective 4: Design an unsupervised spatio-temporal feature learning technique based on task-dependent uncertainty.

Unsupervised learning is another area of machine learning where a model learns without leveraging labeled data. In multi-object tracking and segmentation applications, labeled data generation for each track is costly and time-consuming. To address these issues, we propose to devise a spatio-temporal clustering approach to track and segment individual target instances in a video sequence. Since embedded feature extraction and tracklet associations are independent of detector training, we can easily employ this unsupervised track generation approach with any pre-trained multi-task predictor without using manual track labels. Thus, our multiple-object tracking and segmentation algorithm can be applied to any dataset as long as the detector performs well in a new domain. Our SSL approach from Objective 2 can enhance the detector performance in new scenarios.

Objective 5: Implement a real-time multi-camera multi-object tracking system.

We apply our proposed SSL-based models in a multi-camera tracking algorithm. We devise a real-time multi-camera tracklet association (MCTA) approach by efficiently leveraging both central processing unit (CPU) and graphics processing unit (GPU) computations. The objective of the real-time MCTA is to combine Objective 1 and Objective 2 for real multi-camera tracking systems where creating labeled data for each facility can be challenging. Regarding the consistency of multiple object identities across cameras and the computational complexity for large-scale camera networks, our goal is to apply MCTA in other application domains such as unmanned aerial vehicle (UAV) object tracking, self-driving vehicles, and video analytic-based monitoring systems.

1.3 Dissertation Contributions

So far, the outcomes from the research described in this dissertation have been published in the form of the three following peer-reviewed papers:

1. **A. Siddique**, R. J. Mozhdehi, H. Medeiros, "Unsupervised Spatio-temporal Latent Feature Clustering for Multiple-object Tracking and Segmentation", in British Machine Vision Conference, 2021.
2. **A. Siddique**, H. Medeiros, "Tracking Passengers and Baggage Items using Multiple Overhead Cameras at Security Checkpoints", in IEEE Transactions on Systems, Man, and Cybernetics: Systems, Dec. 2022.
3. **A. Siddique**, A. Tabb and H. Medeiros, "Self-Supervised Learning for Panoptic Segmentation of Multiple Fruit Flower Species," in IEEE Robotics and Automation Letters, vol. 7, no. 4, pp. 12387-12394, Oct. 2022.

Unsupervised Spatio-temporal Clustering for Multiple-object Tracking and Segmentation. In [17], we proposed an unsupervised spatio-temporal latent feature clustering algorithm to improve tracking consistency without leveraging manual annotations for multiple-object tracking and segmentation. We extended this method to solve the occlusion problem by using uncertainty-aware latent features in a robust Re-ID approach. We also exploited an SSL technique to enhance the multiple-object detections for multiple object tracking and segmentation (MOTS) tasks. Our end-to-end algorithm reduces tracking failures in a new domain for MOTS applications. Our research is based on self-supervised/unsupervised learning techniques, and it works in unseen video datasets for which obtaining manual annotations is very tedious and expensive.

Self-Supervised Learning for Detection. Another application of this work is a computer vision-based video analytics approach to automate the screening process and reduce the cognitive load of Transportation Security Officers (TSOs) at the air-

port security checkpoint. The contribution was part of a multi-institutional project called Correlating Luggage and Specific Passengers (CLASP). The passengers and their corresponding items, such as handbags, suitcases, and backpacks, are tracked and associated throughout airport security checkpoints equipped with multiple overhead cameras. We employed a pre-trained multi-object detector model and proposed an SSL technique [18] where we use test-time data augmentation, proposal regression, and unsupervised clustering-based pseudo-label generation to update the initial detection model. Our learning technique reduces the instance uncertainty during the transfer of knowledge from one security checkpoint to another without resorting to significant amounts of human labels. Our method can be easily applied to fully self-supervised or semi-supervised cases depending on the scenario complexity in terms of perspective distortion, scale, and appearance variations.

Self-Supervised Learning for Multi-species Fruit Flower Segmentation. We developed another SSL technique to address the semantic uncertainty prediction of a multi-task model [9] for multi-species fruit flower segmentation and counting applications. This method is also helpful in estimating the blooming statistics of an orchard to optimize fruit production. To transfer model knowledge from one orchard to another, we proposed a rotation-based test-time data augmentation strategy for pseudo-label generation and a similar augmentation approach for the model update. We also employ a semantic refinement strategy [25] to enhance the quality of the pseudo-labels and then update the model using the robust automatically generated labels with rotation invariance property, i.e., prediction scores and segmentation contours remain stable even though we rotate the original input frames. We have found a significant improvement over this application's recent baselines and propose a simple strategy to count the flowers in multiple orchards. Our method only uses labeled data to initialize the multi-task model on a single orchard dataset. Still, our fully self-supervised approach reduces the dependency on tedious and expensive

human labels for multi-species flower datasets.

Multi-Camera Tracklet Association. We devise a real-time unsupervised multi-view data association technique to solve multiple object tracking and association problems in overhead camera networks. The proposed tracking-by-detection algorithm consists of a self-supervised detector and single-camera tracker, including a robust Re-Identification (Re-ID) module. Cross-camera association is the core component of multi-camera tracking. We use 2D projections of the single camera tracks and solve the bipartite association problem for camera pairs with overlapping fields of view. The one-to-one matching of trajectories over the camera pairs in the entire network to determine the hand-off of the passengers' identities across cameras is obtained using a graph optimization method. Since our approach only uses the trajectories of the targets and the image transformation-based homography without relying on camera calibration and overhead target appearances, our cross-camera association algorithm is scalable, i.e., we can increase the network size based on the available computational resources. This approach is applicable to different fully/partially overlapping networks without restrictions on the camera layout and the need for expensive human labels.

Real-time Multi-Camera Tracking. Finally, to apply the core components of this dissertation to a real-world application, we propose to develop a real-time multi-camera tracking (MCT) algorithm. This work is also part of the CLASP project, where multiple institutions are responsible for developing algorithms to address different aspects of the problem. Along this direction, we developed online versions of our semi-supervised detector, self-supervised SCT, and MCTA that process input video frames as multi-camera batches using several GPUs. We assign a GPU process to each camera to generate real-time multi-object detection and tracking. To combine the results from each GPU process, we transfer the processed data into another parallel CPU process to perform multi-camera association in real-time.

1.4 Dissertation Organization

This dissertation consists of seven chapters. In chapter 1, we introduce the fields of computer vision in real-world applications and the four main problems addressed by the proposed methods in this dissertation. Chapter 2 contains the background related to some computer vision concepts, machine learning techniques to process visual data, design methodologies for convolutional neural network models, and the summary of the existing learning techniques in multi-object detection, segmentation, single-camera, and multi-camera tracking applications. Chapter 3 describes the novel data augmentation-based pseudo-label generation technique and the proposed uncertainty-aware self-supervised learning algorithm for multi-object detection and instance segmentation tasks. Chapter 4 focuses on applying our self-supervised learning technique for panoptic segmentation tasks in multi-species fruit flower segmentation applications. Chapter 5 describes our proposed method to simultaneously segment and track multiple objects using an unsupervised multi-task learning approach. Chapter 6 describes the proposed multi-camera tracklet association algorithm for a real-time video analytic-based surveillance application. Finally, Chapter 7 concludes this dissertation by providing a summary of the findings and then discussing possible directions for future work.

CHAPTER 2

BACKGROUND

2.1 Basic Concepts of Computer Vision for Real-World Applications

There are a number of intermediate steps involving the application of computer vision algorithms for solving real-world problems. Machine vision algorithms start with image acquisition by optical sensors and lead to the solution of real-world decision-making tasks such as self-driving vehicles, robotic automation, and surveillance. Designing modern computer vision algorithms involves sensor data encoding-decoding, data augmentation and pre-processing, data splitting into train/val/test, feature extraction, machine learning or deep learning algorithm design, and then intuitively utilizing model features for predicting a task solution or combine a number of task solutions to deliver the final outcome.

2.1.1 Image Acquisition

Image acquisition refers to the process of capturing visual data where cameras, scanners, or even smartphones capture light using an imaging system and convert it into a digital image. This process involves several steps: i) Light enters the camera through the lens (illumination and reflectance). The lens focuses the light onto the image sensor, which is a light-sensitive electronic component. ii) The image sensor consists of millions of tiny light-sensitive cells, or pixels, that convert the incoming light into electrical signals (sampling). iii) The electrical signals are then processed by the camera's electronics and transformed into a digital image file (quantization), which can be stored as a single channel (gray scale) or multiple color channels (e.g., Red, Green, and Blue channels also called RGB) [29]. The quality of the image depends on various factors, including the resolution of the image sensor, the aperture

of the lens, and the exposure settings of the camera.

2.1.2 Image Data Augmentation

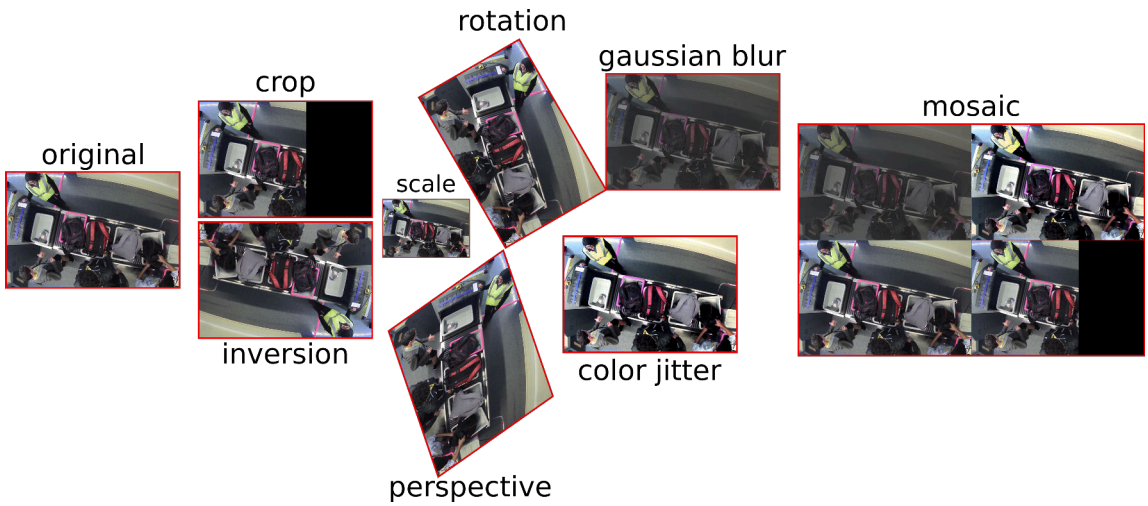


Figure 2.1: Different image augmentation techniques used in machine learning for computer vision applications.

Image data augmentation is the process of manually or automatically [30] increasing the size of a dataset by generating modified versions of existing images. Data augmentation is useful when training machine learning models, as it can prevent overfitting and improve the generalization ability of a model. Image rotations augment the images by rotating at different angles so that the model can learn to recognize objects at different orientations. In flipping, the image transform into horizontally or vertically so the model learns to recognize objects that are mirror images of each other. The image cropping technique remove a portion of the image so that the model learns to focus on the most important features of the image. Adding noise to the image is popular to learn about the variations in the image, such as variations in lighting or background clutter. The color jittering method adjust the color or contrast of the image (Fig. 2.1), so that the model learns about different lighting conditions. In the

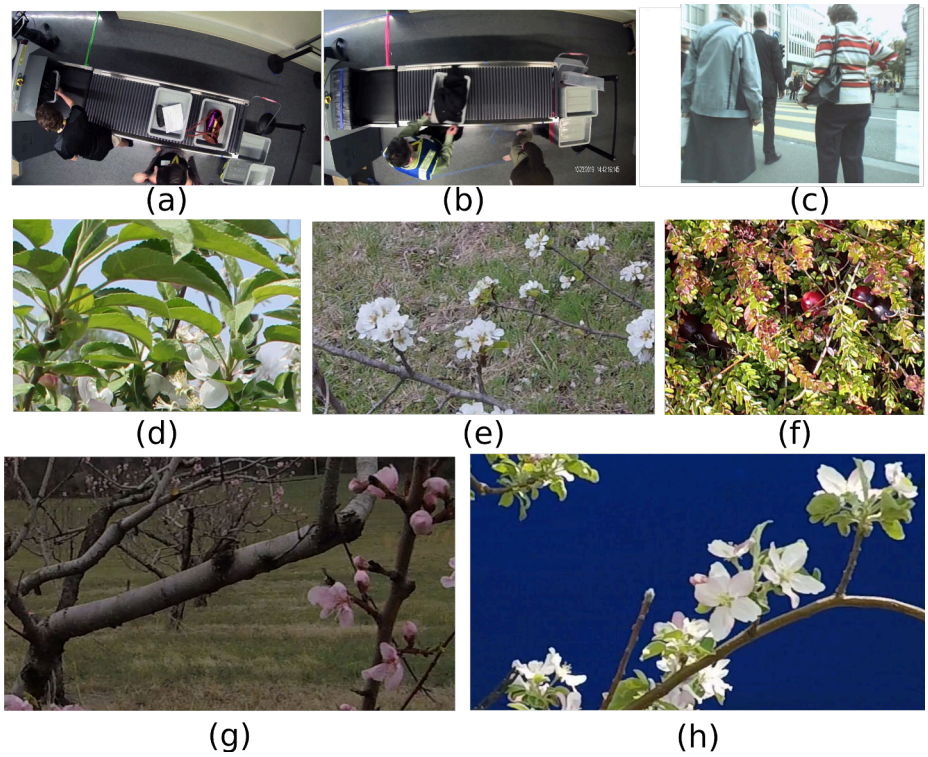


Figure 2.2: Sample frames from the datasets used in this dissertation: (a) CLASP1, (b) CLASP2, (c) MOTSChallenge, (d) AppleA, (e) Pear, (f) CRAID, (g) Peach, (h) AppleB.

motion blur augmentation technique, we apply filters or blurring effects to the image to include different scene conditions, such as when the image is blurry or distorted due to rapid motions. A generative model can generate new images that are similar to existing images in a dataset, thus allowing the model can generalize to new data. Fig. 2.1 shows some of the most popular image data augmentation techniques used to generalize a model.

2.1.3 Probabilistic Representations of Vision Data

Data distribution analysis (see Fig. 2.2) is a useful technique to understand the underlying structure of the data, i.e., how much variability there is among the samples across the training, validation, and testing sets. Probabilistic representations of vision data usually represent image information using probability distributions

rather than fixed values. This can be useful in tasks such as image classification, object detection, and segmentation, where there is often uncertainty or ambiguity in the images. Gaussian Mixture Models (GMMs) are one of the popular probabilistic representations of vision data. GMMs are a probabilistic model that can be used to represent the probability distribution of a dataset. Each GMM is composed of a set of uni-variate or multi-variate Gaussian distributions, each of which is defined by its mean vector $\hat{\mu}_k$ and covariance matrix $\widehat{\Sigma}$ and can be defined as follows

$$\mathcal{N}(\hat{\mu}_k, \widehat{\Sigma}) = \frac{1}{(2\pi)^{m/2}|\widehat{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(v_k - \hat{\mu}_k)^T \widehat{\Sigma}^{-1}(v_k - \hat{\mu}_k)\right) \tag{2.1}$$

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{i=1:y_i=k}^{N_k} h(x_i) \tag{2.2}$$

$$\widehat{\Sigma} = \frac{1}{N} \sum_{k} \sum_{i=1:y_i=k}^{N} (h(x_i) - \hat{\mu}_k)(h(x_i) - \hat{\mu}_k)^T, \tag{2.3}$$

where Eq. 2.3 represents a GMM for a learned representation $h(\cdot)$ of a CNN-based classification model trained for k classes, v_k is the number of samples belonging to class k, N is the number of samples for all classes, and x_i represents an input image. The parameters of these distributions can be learned from the data using Expectation Maximization (EM) [31] algorithm or from the convolutional feature representations of the data in a deep classifier model [32]. This can be useful in tasks such as object recognition, where the GMM can be used to represent the probability distribution of different object classes. For example, when we train a classification model using the CLASP1 dataset ((a) in Fig. 2.2) or the multi-species flower datasets ((d) and (h) in 2.2), we observe that a mixture of multiple Gaussian distributions represent the classes of interest during training. Fig. 2.3-(a-b) show the comparisons of mixtures of two Gaussian between training, validation, and open or test sets where a Wide-ResNet-42 model is trained for two categories (AppleA and AppleB in Fig. 2.3). Fig. 2.3-(c-d) also show the training and validation/open distributions comparison where the model is trained for two classes (camera 9 and 11 images, also called CL1) in

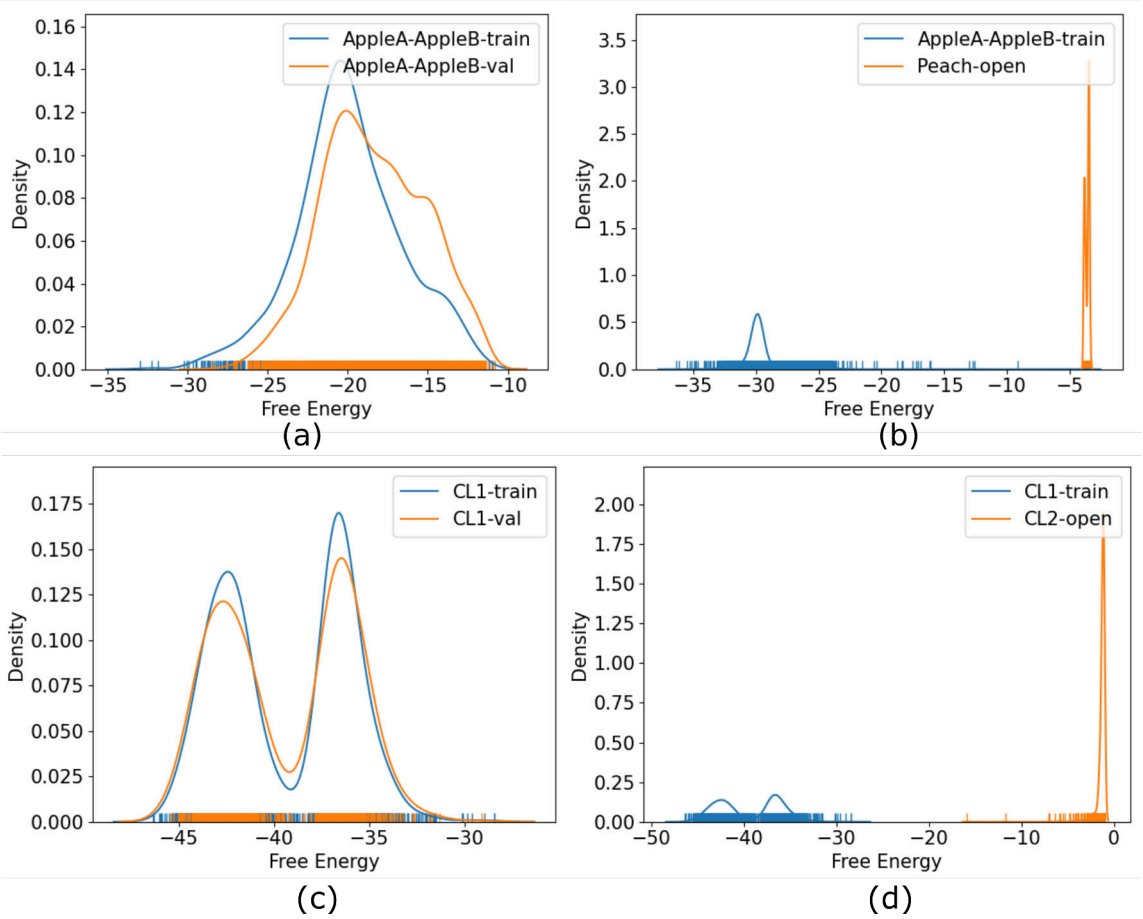


Figure 2.3: Comparisons of free energy distribution between train set and val/open sets: a) AppleA-AppleB train/val, b) AppleA-AppleB train and Peach-open, c) CLASP1-train/val, e) CLASP1-train/val and CLASP2-open.

CLASP1. Fig. 2.3 shows how different the training data distributions are compared to the validation data. Thus, the learned probabilistic representation can identify the outliers in the test data.

Another probabilistic representation is based on variational autoencoders (VAEs) [33], which is a generative model that can be used to learn a probabilistic representation of the data. VAEs consist of an encoder network that maps the input data to a latent space, and a decoder network that maps the latent space back to the input data. The encoder network is trained to learn a probabilistic representation of the data, and the decoder network can be used to generate new samples from the learned

distribution.

Common applications of the Gaussian distribution in machine learning and computer vision include: i) density estimation where the Gaussian distribution can be used to estimate the probability density function of a dataset (Fig. 2.3), which can be useful for understanding the underlying structure of the data; ii) anomaly detection where the Gaussian distribution can be used to identify outliers or anomalies in a dataset (Fig. 2.3), by comparing the data to the expected distribution based on the mean and standard deviation; iii) feature scaling, where the Gaussian distribution can be used to scale and normalize features in a dataset, which can improve the performance of machine learning algorithms.

2.1.4 Feature Extraction

Feature extraction is the process of identifying and extracting important cues or characteristics from a dataset that can be used to represent the data in a more compact and meaningful way. In computer vision, feature extraction is often used to extract relevant features from images or videos for the purpose of analysis, classification, or other tasks. There are many different techniques for feature extraction including i) edge detection, which involves identifying the edges or boundaries of objects in an image; ii) feature extraction based on image filters such as Gabor filters extract features such as texture or shape; iii) automatic feature extraction using deep learning models, such as convolutional neural networks (CNNs) (Fig. 2.4) or recurrent neural networks (RNN).

2.1.5 Clustering

Clustering methods in machine learning partition a dataset into groups, or clusters, based on the similarity of the data points within each cluster. The goal of clustering is to identify patterns or relationships within the data and group similar

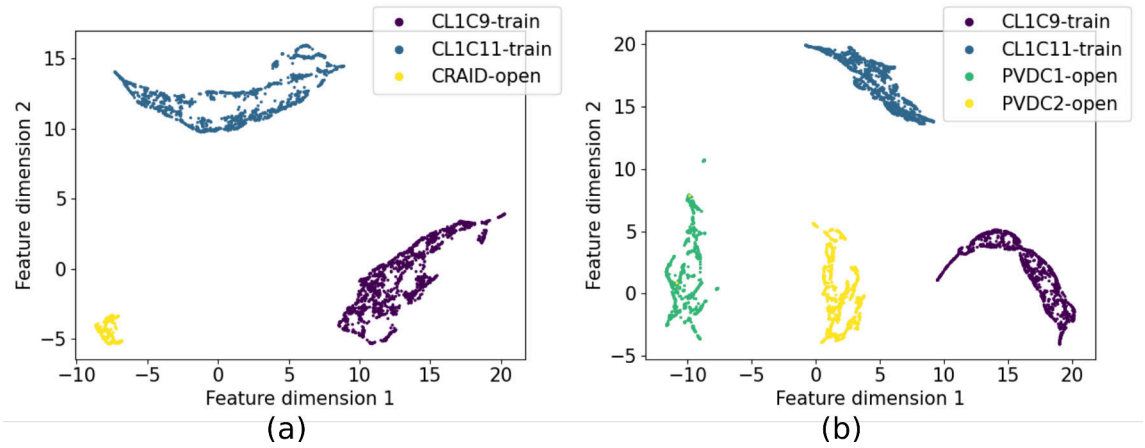


Figure 2.4: visualization of the embeddings from a classification model (WRN-42) using Uniform Manifold Approximation and Projection (UMAP).

data points together. There are many popular unsupervised clustering techniques already developed to group the data points based on their similarities. In k-means [34] clustering, the data points are partitioned into a pre-defined number of clusters by iteratively reassigning each data point to the cluster with the nearest mean, until convergence. Unlike the k-means, the constrained k-mean iteratively updates the assignment based on the nearest mean and a connectivity graph of data domain knowledge. The hierarchical clustering uses a nested approach to form cluster within one or more larger clusters. There are two main types of hierarchical clustering strategies: agglomerative, which starts with individual data points and merges clusters together, and divisive, which starts with all the data in a single cluster and splits it into smaller clusters. The density-based clustering uses the density of the data points, rather than their distance from a central point and they are often used to identify clusters with arbitrary shapes, and are robust to noise and outlier. In spectral clustering technique, the eigenvectors of the similarity matrix of the data are used to identify clusters in data that are not linearly separable, and can handle large datasets efficiently. Clustering techniques are often used in computer vision for tasks such as image segmentation, anomaly detection, and data visualization. For

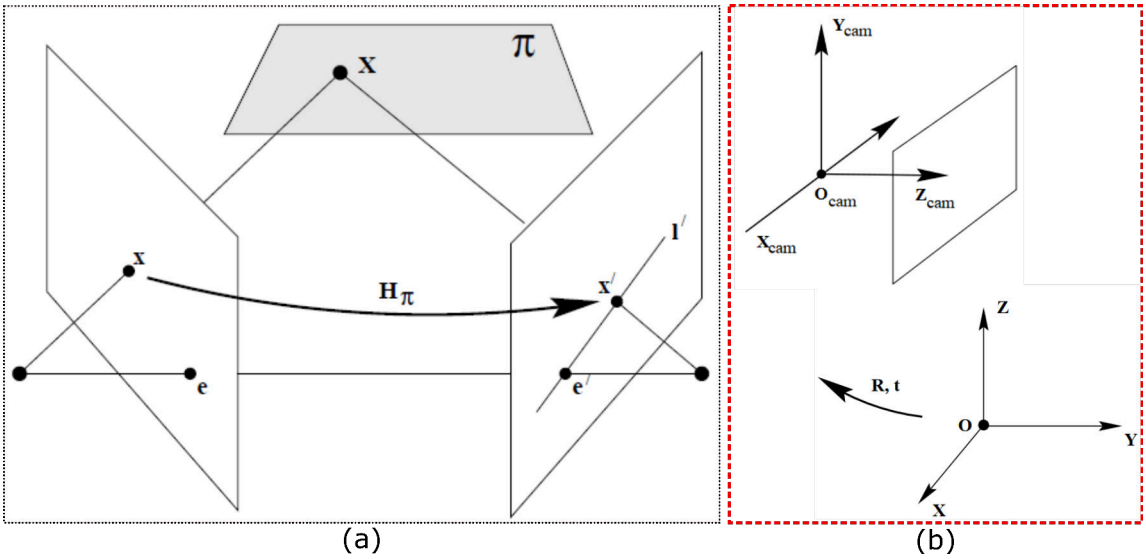


Figure 2.5: Projective transformation [2]: (a) between two image plane, (b) between world and camera coordinate system.

example, Fig. 2.4 shows the 2D visualization of the 512D feature from a deep model using UMAP [35].

2.1.6 Multi-view Geometry

Multi-view geometry is the study of the mathematical relationship among multiple images of the same scene or object, taken from different viewpoints. It is a fundamental topic in computer vision and has many practical applications, such as structure from motion, 3D reconstruction [36, 37], and object recognition. Epipolar geometry (Fig. 2.5) determines the geometric relationship between corresponding points in multiple images of the same scene, as captured by cameras with overlapping fields of view. It is characterized by the epipolar constraint, which states that the projections of a point onto the image planes of the cameras must lie on a line called the epipolar line ($I^{'}$ in Fig. 2.5). Based on the epipolar geometry, we can derive the following relationship between a point projected onto two camera planes

$$x^{'} = H_{2\pi}x_{\pi} = H_{2\pi}H_{1\pi}^{-1}x = Hx, \tag{2.4}$$

or equivalently,

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathrm{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = K\,[R|T] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & \alpha & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{2.5}$$

where X, Y, Z are the 3D world space coordinates, x, y are the 2D image plane coordinates, P is for camera perspective projection matrix which consists of camera intrinsics, rotation, and translation matrix K, R, and T respectively. The camera intrinsics matrix consists of focal length in pixels, f_x, f_y, pixel coordinates of the camera optical center, p_x, p_y, and the skew coefficient, α. The rotation matrix, R consists of nine parameters (3×3 matrix) and translation vector T consists of three parameters t_x, t_y, t_z. Thus R and T matrix determine the projective transformation between world and image coordinates. Triangulation or stereo matching techniques leverage the camera calibration to estimate the 3D structure of a scene (Fig. 2.5). Here, the same object from multiple 2D images is projected onto a global map using a transformation matrix (Eq. 2.5) to determine the 3D structure. A sequence of 2D images from a moving camera is used to estimate the 3D structure and motion using multi-view geometry. Camera calibration is a prerequisite for using multi-view geometry, where the intrinsic and extrinsic parameters of the cameras (Eq. 2.5), such as the focal length, principal point, and camera pose, relate the images to a common coordinate system. Multi-view geometry plays a key role in many practical applications of computer vision, such as augmented reality, robotics, and autonomous vehicles.

2.2 Downstream Tasks in Computer Vision Applications

Downstream tasks in computer vision refer to the solution to a particular problem after performing the initial processing and feature extraction of visual data. These solutions may directly involve a specific task, or they may involve further analysis or interpretation of the data. As shown in Fig. 2.6, common downstream tasks in computer vision applications are: object recognition techniques to localize [38, 39] and classify specific objects [40] within an image or video frame; object tracking methods [12, 41, 23] to predict the movement of specific objects within a sequence of images or video frames; image segmentation methods [15, 9] to partition an image into different regions or segments based on the characteristics of the pixels in the image; image captioning approaches [42] to transform image objects into words based on the objects and context within the image; depth estimation methods [10, 43, 44] to understand and reconstruct the scene for tracking, navigation, and augmented reality applications; pose estimation techniques [45] to locate key points in a scene for human-object interaction and human activity recognition, or in a multi-view 6D pose to recover the object shapes under severe occlusions and cluttered scenarios [46]; optical character recognition [47] and scene text detection [48] recognize characters in images or videos.

In these downstream task algorithms, a core component is effective feature extraction from the input visual data using some popular CNN backbone architectures [49, 50, 51, 52] or vision transformers [52, 53]. After extracting the features, the task-specific goal is to design a head architectures to backpropagate the task loss computed from a loss function. Based on the type of task, these heads can be a single-stage [54] or multi-stage [1] or have specialized architecture [52] to solve the task without a number of pre-processing and post-processing steps.

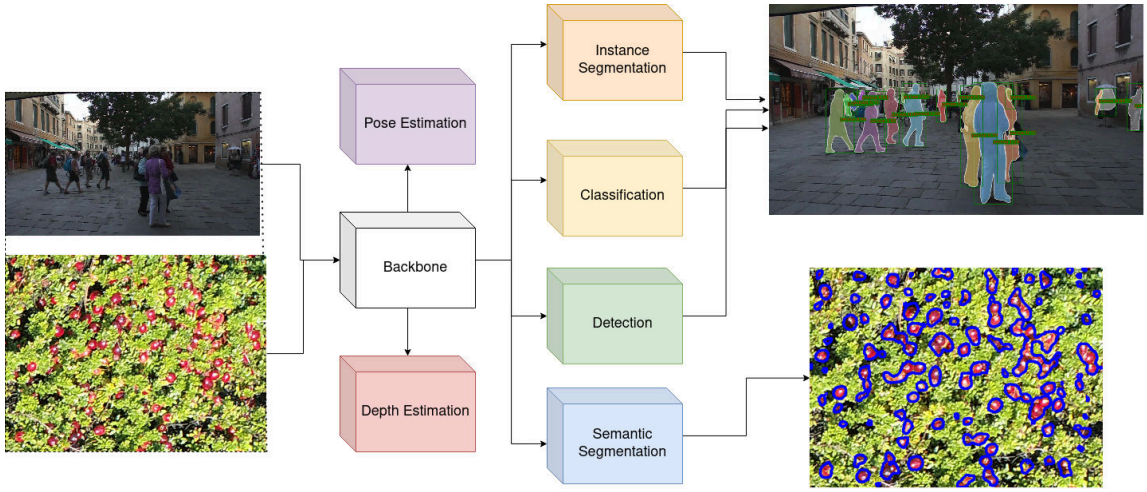


Figure 2.6: Downstream tasks in computer vision applications.

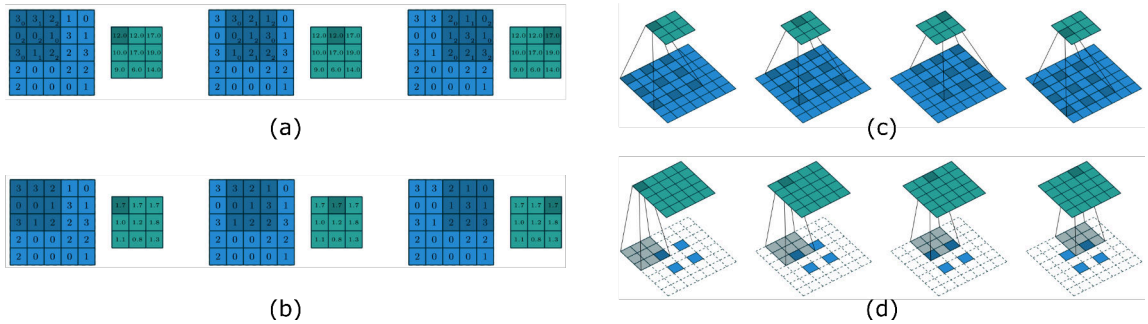


Figure 2.7: Understanding discrete convolution and pooling operations in a CNN [55]. (a) discrete convolution, (b) pooling, (C) dilated convolution, (d) transpose convolution.

2.3 Convolutional Neural Networks

A convolutional neural network (CNN) is a type of artificial neural network specifically designed for processing data with a grid-like topology, such as an image. It is composed of multiple layers of interconnected neurons, and is trained using a variant of the backpropagation algorithm. One of the key features of CNNs is the use of convolutional layers, which are designed to automatically and adaptively learn spatial hierarchies of features from the input data. These layers apply a convolution

operation to the input data, which involves sliding a small kernel or filter over the input and computing the dot product between the entries of the kernel and the input at each position. Fig. 2.7 (a) shows a 3×3 filter convolved with the input data matrix of size 5 resulting in the feature map of size 3×3 for a stride 1×1. The resulting feature map is then transformed using non-linear activation functions, such as the rectified linear unit (ReLU) function, to apply for different downstream task predictions. Another key feature of CNNs is the use of pooling layers, which are used to downsample the spatial dimensions of the input data. This can help to reduce the computational complexity of the network, as well as the overfitting of the model to the training data. Fig. 2.7 (b) shows a 3×3 average pooling operation on a 5×5 input feature map with stride 1×1, which down samples the input feature map to 3×3. The pooling operation computes the average or maximum values of the subregions in the input feature map. The other convolutional operations in deep learning are: i) dilated convolution (Fig. 2.7 (c)) to increase receptive fields without increasing the number of learnable weights; ii) transpose convolution (Fig. 2.7 (d)) to upsample the convolutional feature using learnable parameters instead of using typical interpolation techniques.

2.3.1 Convolutional Neural Network Architectures

There are many different CNN architectures that have been developed for various computer vision tasks. Some of the most well-known architectures are described in this section.

LeNet: This is an early CNN architecture developed by Yann LeCun et al. [56] in the late 1980s and early 1990s. It was originally designed for handwritten digit recognition, and consists of a series of convolutional and pooling layers, followed by fully connected layers. Fig. 2.8 (a) shows LeNet-5 [56], where the inputs are the 32×32 handwritten images and the output is a fully connected layer with 10 units

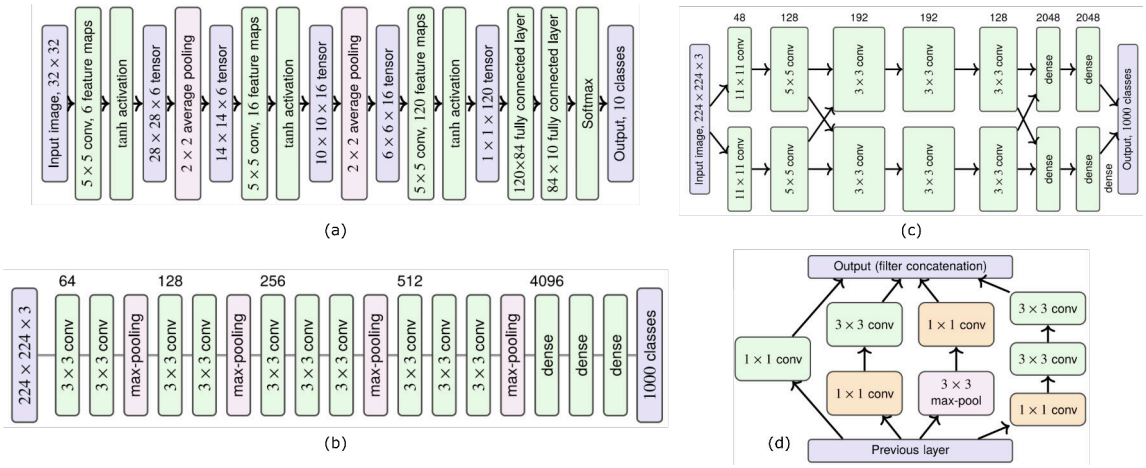


Figure 2.8: CNN architectures.

for 10 categories of the digits. This network uses gradient-based learning to extract 2D shape features from the hand written texts.

AlexNet: To tackle the challenges of high-resolution input images and extract more contextual information-based CNN features for computer vision downstream tasks, Alex Krizhevsky et al. [50] developed a modern CNN architecture that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. It is the first successful deep neural network that employs max-pooling for downsampling the features and ReLU activation as the non-linearity. It also explores the importance of data augmentation in neural network training for computer vision problems and the drop-out approach for additional regularization. The network comprises two parallel branches of 8-trainable layers for input size $224 \times 224 \times 3$ and three fully-connected layers for 1000 output categories, followed by a series of convolutional and max-pooling layers, as shown in Fig. 2.8 (c). AlexNet has been widely used as a robust CNN architecture for most downstream tasks in computer vision applications.

VGG: In 2014, the Visual Geometry Group (VGG) at Oxford University introduced a successful deep ImageNet classification model called VGG network [51]. The main feature of this network is the decomposition of large convolutions, which

allows for the use of only 3×3 convolutions rather than 7×7 or 5×5 filters for very deep models (16 to 19 layers). This approach helps in the design of very deep networks with fewer weights.

Inception and GoogleNet: This is a CNN architecture developed by Google that was introduced in 2014 [57]. It consists of a series of modules called Inception modules, which use a combination of 1×1, 3×3, and 5×5 convolutional filters to capture different scales and aspects of the input data. Fig. 2.8 (d) shows an Inception module architecture where a 1×1 convolution is used to reduce data dimensionality without hurting the training. Even though the 1×1 convolution does not collect any new features from the neighboring pixels, it is useful to change the dimensionality of the feature vector without computationally expensive transformations.

ResNet: To address the problem of error propagation in deep neural networks after the saturation of the top layers, Kaiming He et al. [49] developed a Residual Network (ResNet) architecture in 2015. This architecture won the ILSVRC 2015 in both ImageNet classification and object detection (using the Faster-RCNN [1] detection architecture) tasks. The basic residual unit shown in Fig. 2.9 computes a function $F(x)$ through a layer for an input x, and then the output of the residual unit becomes $y = F(x) + x$, which is then used as input for the next unit. This approach provides a direct way of flowing gradients around $F(x)$ without vanishing gradients for deep architectures. The first version of ResNet contains 152 layers, which are commonly used as a feature extraction backbone. Other variants, such as ResNet-101 and ResNet-50, are also popular for solving computer vision tasks.

Mobile-Net, Efficient-Net, and Squeeze-Net: To overcome the memory and runtime limitations of large Inception ResNet families in edge devices, a number of light-weight networks with excellent performance have been proposed. In the SqueezeNet [58] architecture, the 1×1 convolution-based Fire module replaces the Inception unit to reduce the number of channels and then expands back to gener-

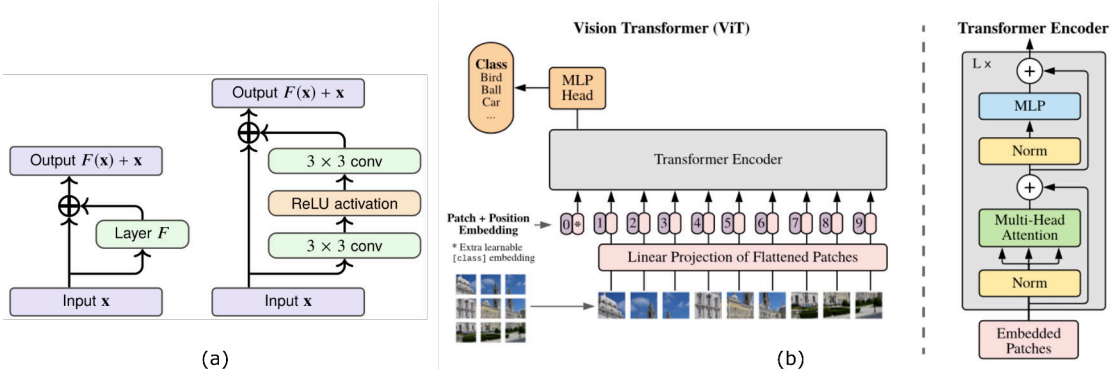


Figure 2.9: Resnet and ViT architecture.

ate the concatenated outputs. The idea of depth-wise separable convolutions [59] is the key component of the memory-efficient MobileNet family of networks. To automate the typical deep learning issues of designing building blocks (convolutional layer, max-pooling layer) and task-specific loss functions (classification, detection, and segmentation), neural architecture search techniques motivated the development of EfficientNet [60], which successfully addresses the performance/efficiency trade-off in deep learning algorithms.

Vision Transformer: Transformer networks were originally developed for natural language processing tasks to capture the long-range relationships among a sequence of words in natural language sentences. It has also been explored in computer vision tasks by combining the benefits of CNN features [38] or without leveraging CNN [52, 53]. These models operate by first encoding an input image into a feature representation (see Fig. 2.9), which is then processed by the Transformer architecture. In the encoding step, an image is divided into a grid of non-overlapping patches, and each patch is passed through a CNN to produce a feature map. The feature maps from all patches are then combined and flattened into a feature vector, which is the input to the Transformer. Then the feature vector passes through several self-attention layers, allowing the model to learn relationships between different parts of the image. The attention mechanism allows the model to focus on important

regions of the image, dynamically adjusting its focus as needed. Finally, the output of the Transformer is fed into a downstream task layer, which makes the final prediction about the image. Thus, a Vision Transformer combines the strengths of CNNs for feature extraction with the Transformer architecture for modeling long-range relationships between image elements. This allows for better handling of the spatial dependencies in images, compared to traditional CNN-based models.

2.4 Distance Metrics in Computer Vision

In deep learning and computer vision, distance metrics are often used to compare features or embeddings of images, videos, or other data. Different types of distance metrics can be used depending on the specific task and the properties of the data. Some common types of distance metrics used in deep learning and computer vision are introduced below.

Cosine Similarity Distance: The cosine similarity measures the distance between two vectors in the vector space model (VSM). The cosine similarity $S_c(\tau_a, \tau_p)$ and the corresponding cosine distance $D_c(\tau_a, \tau_p)$ between two vectors τ_a and τ_p can be defined as follows

$$S_c(\tau_a, \tau_p) = \cos\theta = \frac{\vec{\tau_a} \cdot \vec{\tau_p}}{\|\vec{\tau_a}\| \, \|\vec{\tau_p}\|}, D_c(\tau_a, \tau_p) = 1 - \cos\theta. \tag{2.6}$$

The soft-cosine similarity distance is a modified version of the traditional cosine distance [61] proposed in natural language processing that considers the similarity between two feature vectors in cosine distance.

$$sd_c = 1 - \frac{\sum\sum_{i,j}^{N} s_{ij}\tau_{a_i}\tau_{p_j}}{\sqrt{\sum\sum_{i,j}^{N} s_{ij}\tau_{a_i}\tau_{a_j}}\sqrt{\sum\sum_{i,j}^{N} s_{ij}\tau_{p_i}\tau_{p_j}}}, \tag{2.7}$$

If there is no similarity between τ_{a_i} and τ_{p_j}, $s_{ij} = 0$ for $i \neq j$ and $s_{ii} = 1$, which reduces to the traditional cosine similarity distance D_c (Eq. 2.6).

Mahalanobis Distance: The Mahalanobis distance measures the distance between a point and a distribution. Unlike the Euclidean distance, the Mahalanobis

distance takes into account the correlation between variables. In this metric, each variable contributes to the distance based on its correlation. The Mahalanobis distance $D(x, \mu)$ between $x \in R^p$ and the mean $\mu = E(X)$ of the p-variate distribution $f_x(.)$ can be defined as

$$D(X, \mu) = \sqrt{(X - \mu)^T \Sigma^{-1} (X - \mu)}. \tag{2.8}$$

For the identity covariance matrix, the Mahalanobis distance becomes the Euclidean distance.

Hausdorff Distance: The Hausdorff distance metric measures the similarity between two sets of points and is defined as the highest distance between any point in one set and the closest point in the other set. The directed Hausdorff distance [62] between two sets of points τ_a and τ_p can be defined as the maximum distance between each point $x \in \tau_a$ and its nearest neighbor $y \in \tau_p$

$$\tilde{H}(\tau_a, \tau_p) = \max_{x \in \tau_a} \{ \min_{y \in \tau_p} \|x, y\| \}, \tag{2.9}$$

where $||\dot{}||$ is the Euclidean distance function. Since $\tilde{H}(\tau_a, \tau_p) \neq \tilde{H}(\tau_p, \tau_a)$, the Hausdorff distance is also defined as the maximum of the directed Hausdorff distances in both directions, i.e.,

$$H(\tau_a, \tau_p) = \max(\tilde{H}(\tau_a, \tau_p), \tilde{H}(\tau_p, \tau_a), \tag{2.10}$$

Frechet Distance: The Fréchet distance [63, 64] measures the similarity between two curves. It is defined as the minimum over all possible alignments of the two curves of the maximum distance between corresponding points on the two curves.

$$F(\tau_a, \tau_p) = \inf_{(\alpha, \beta) \in X} \max_{t \in [0,1]} \{ \|\tau_a(\alpha_t), \tau_p(\beta_t)\| \}, \tag{2.11}$$

where the Fréchet distance $F(\tau_a, \tau_p)$ between two 2D polygonal curves τ_a and τ_p is the infimum (greatest lower bound) for all α, β of the maximum of all the Euclidean

distances between $\tau_a(\alpha_t)$ and $\tau_p(\beta_t)$. Here, $t \in [0, 1]$ indicates the time instances when we compute the distance between two curve points. To tackle the computational complexity of $O(n^2 \log(n^2))$, the discrete Frèchet distance is proposed in [65], for which the computational complexity is $O(n^2)$. The discrete Frèchet distance is often used as a similarity measure in image registration, object recognition, and computational geometry.

2.5 Cost Functions

A cost function is a measure of the error or loss in a machine learning model that is used to optimize the model during training. In deep learning and computer vision, the goal is often to minimize the cost function in order to improve the performance of the model on a given task. There are many different cost functions that can be used in deep learning and computer vision, depending on the specific task and the characteristics of the data. A few of the most common cost functions are presented below.

Mean Squared Error (MSE): This is a common cost function for regression tasks that measures the average squared difference between the predicted output and the true output. This loss is also called the L_2 norm, which measures the mean squared error between each input element x_i and target y_i, which represent the true value and the predicted value, respectively, i.e.,

$$\mathcal{L}_{\text{mse}} = \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2, \tag{2.12}$$

where n is the total number of input elements. **Cross-entropy Loss:** The Cross-entropy loss is a common cost function for binary classification tasks, which measures the difference between the predicted probability of the positive class and the true label. For logistic regression, the cost function corresponds to the maximum likelihood

estimatie (MLE). The likelihood for the given data $\mathcal{D} = \{(y_c, x)\}$ is defined as

$$P(\mathcal{D}|W) = \prod_{c=1}^{\mathcal{C}} [\hat{p}(y_c|x)]^{p(y_c|x)} [1 - \hat{p}(y_c|x)]^{1-p(y_c|x)}, \tag{2.13}$$

where $p(y_c|x)$ is the probability distribution of the training set, $\hat{p}(y_c|x)$ is the predicted distribution from a deep neural network, $\mathcal{C}$ is the total number of target classes, x is the input feature vector, and W are the model parameters. The MLE is defined as the set of parameters which maximizes the log-likelihood [66]

$$\hat{\mathcal{W}} = \arg\max_{\theta} \sum_{c=1}^{\mathcal{C}} p(y_c|x) \log \hat{p}(y_c|x). \tag{2.14}$$

Thus, the negative log-likelihood for logistic regression is also called the cross entropy and is defined as [67]

$$\mathcal{L}_{\text{ce}}(W, y) = -\sum_{c=1}^{\mathcal{C}} p(y_c|x) \log \hat{p}(y_c|x). \tag{2.15}$$

For binary classification tasks where $y_c \in \{0, 1\}$, the negative log-likelihood is defined as

$$\mathcal{L}_{\text{bce}}(W, y) = -p(y_c|x) \log \hat{p}(y_c|x) - [1 - p(y_c|x)] \log[1 - \hat{p}(y_c|x)]. \tag{2.16}$$

Thus this loss function measures the negative log probability of the true label $p(y_c|x)$ given the predicted probability distribution $\hat{p}(y_c|x)$.

Hinge Loss: The hinge embedding loss measures the similarity between input and predicted values. For the n-th sample, the hinge loss $\mathcal{L}_{hinge}$ is defined as

$$\mathcal{L}_{hinge}(x_n, y_n) = \begin{cases} x_n & \text{if} \quad y_n = 1 \\ \max(0, \Delta - x_n) & \text{if} \quad y_n = -1, \end{cases} \tag{2.17}$$

where the given input tensor is x and the corresponding label tensor y contains 1 or -1. The total loss will be the average of the losses for n-th samples. This loss is mostly useful for classification problems, learning non-linear embeddings, and semi-supervised tasks.

Kullback-Leibler Divergence Loss (KLD): The KLD loss compares the distance between two probability distributions. It is defined as

$$KLD(x, y) = y(\log(y) - \log(x)), \tag{2.18}$$

where y is the predicted probability and x is the probability of the input sample. This loss is similar to the cross-entropy loss (Eq. 2.15), but KLD compares the predicted probability distributions (mean and variance) instead of the predicted confidence.

Triplet Margin Loss: For learning image patch descriptors in computer vision models, the most popular technique is triplets-based feature learning. This loss requires three inputs: $\{\mathbf{a}, \mathbf{p}, \text{and } \mathbf{n}\}$, where $\mathbf{a}$ is the anchor, $\mathbf{p}$ is the positive sample which comes from the same class as $\mathbf{a}$, and $\mathbf{n}$ is the negative sample which comes from a different class. By optimizing the model parameters, the triplet loss brings the representations of $\mathbf{a}$ and $\mathbf{p}$ closer together in feature space and pushes the representation of $\mathbf{n}$ further away from $\mathbf{a}$ [68].

$$\mathcal{L}_t(a, p, n) = \max(0, \mu, \|f_W(\mathbf{a}) - f_W(\mathbf{p})\|_2, \|f_W(\mathbf{a}) - f_W(\mathbf{n})\|_2), \tag{2.19}$$

where μ is the margin parameter. The triplet margin loss is a convex approximation and lower bounds to 0. Another version of triplet loss is proposed in [68] and is called hard negative mining with anchor swap, where the anchor is swapped with positive-based $\min(\|f_W(\mathbf{p}) - f_W(\mathbf{n})\|_2, \|f_W(\mathbf{a}) - f_W(\mathbf{p})\|_2) = \|f_W(\mathbf{p}) - f_W(\mathbf{n})\|_2$. As a result, the hardest negative within the triplet is used to update the weights.

Contrastive Loss: The contrastive loss in deep learning trains the model with a pair of contrastive examples. There are several ways of generating contrastive samples, such as geometric transformations and multi-view samples of the same targets. The training pairs can be positive or negative based on the target similarity.

The contrastive loss is defined as [68]

$$\mathcal{L}_{cont} = \begin{cases} \|f_W(x_1) - f_W(x_2)\| & \text{if} \quad l = 1 \\ \max(0, \mu - \|f_W(x_1) - f_W(x_2)\|) & \text{if} \quad l = -1 \end{cases}, \tag{2.20}$$

where μ is an arbitrary margin and the pairs of training samples $\{x_1, x_2\}$ with their corresponding labels ($l = -1$ for negative pairs and $l = 1$ for positive pairs).

The choice of cost function depends on the specific task and the characteristics of the data, and it is often necessary to experiment with different cost functions in order to find the one that works best for a given problem.

2.6 Datasets

In the design of modern computer vision algorithms, the training, validation, and testing phases usually rely on large-scale benchmark datasets. The type of annotation label depends upon the task being solved by the model we plan to design. One can generate large amounts of training data synthetically or by applying augmentations to simulate the proposed methods, not only for machine vision algorithms but also for text, audio, and time-series recognition models. This section describes relevant benchmark datasets commonly used in several computer vision tasks.

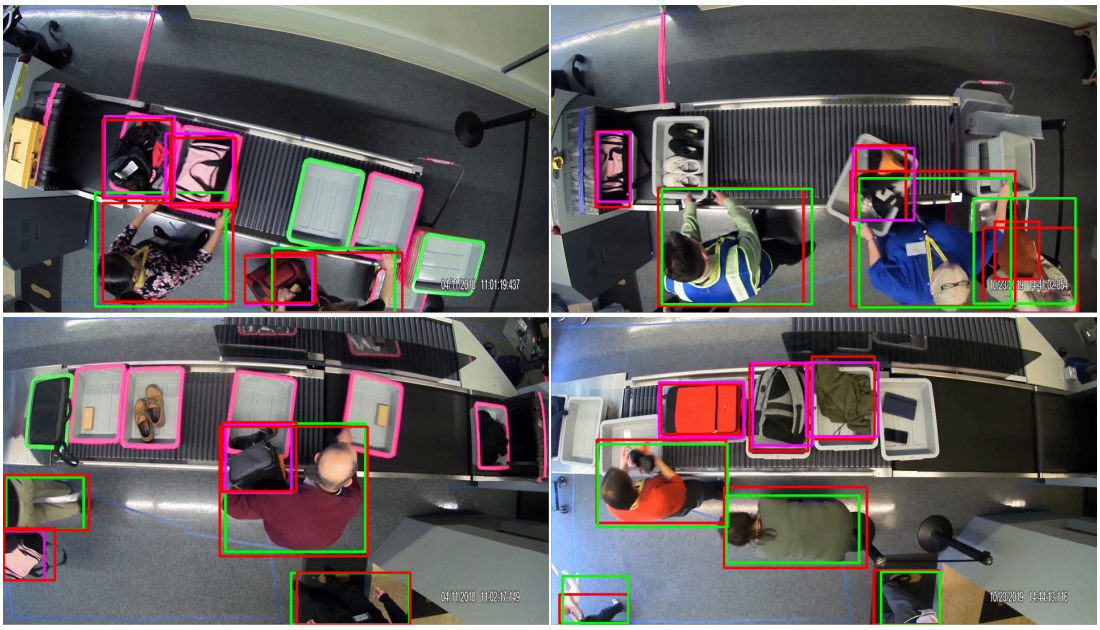
2.6.1 Multi-Object Tracking and Segmentation

In 2018, the *Multi-object Tracking and Segmentation (MOTS)* [12] datasets have been introduced to integrate detection, segmentation, and tracking tasks into a single model. The goal is to use MOTS datasets to overcome the limitations of bounding boxes in multiple object tracking, especially when they overlap significantly due to occlusions or camera perspectives. In the MOTS datasets, the existing bounding box-based MOTChallenge [69], and KITTI [70] labels were extended using manual or semi-automated techniques [6] to generate pixel-wise labels. The MOTS datasets are split into two sets: the MOTSChallenge and MOTS KITTI datasets.

Figure 2.10: Annotation examples from KITTI MOTS (car and pedestrian) datasets.

Figure 2.11: Annotation examples from MOTSChallenge (pedestrian) datasets.

In MOTS KITTI, 12 training video sequences are annotated for 99 *pedestrian* and 431 *car* tracks. There are 9 validation sequences which consist of 68 *pedestrian* and 151 *car* tracks. On the other hand, the 4 crowded MOTSChallenge scenarios contain 228 *pedestrian* tracks. Fig. 2.10 and Fig. 2.11 show sample annotations for *pedestrian* and *car* tracks. An updated MOTS dataset version has also been presented in [71, 72], which is denser in comparison with MOTS [12].



(a) CLASP1 (b) CLASP2

Figure 2.12: Examples of multi-camera CLASP datasets: a) CLASP1, b) CLASP2.

2.6.2 Correlated Luggage and Specific Passengers

Inspired by the Department of Homeland Security (DHS) goal of enhancing security at airport checkpoint screening and assist Transportation Security Officers (TSOs) in terms of cognitive load reduction and detection of theft or abandoned items, a mock airport checkpoint was constructed at the Kostas Research Institute (KRI) video analytics laboratory at Northeastern University. Two overhead video datasets were collected at the KRI laboratory in 2017 (CLASP1) and 2019 (CLASP2), where multiple human subjects participated in activities that emulate actual airport checkpoints. These datasets aim to model passenger activities as a part of the Correlated Luggage and Specific Passengers (CLASP) project. The CLASP1 and CLASP2 datasets contain 4 overhead camera scenarios observed by 14 standard IP surveil-

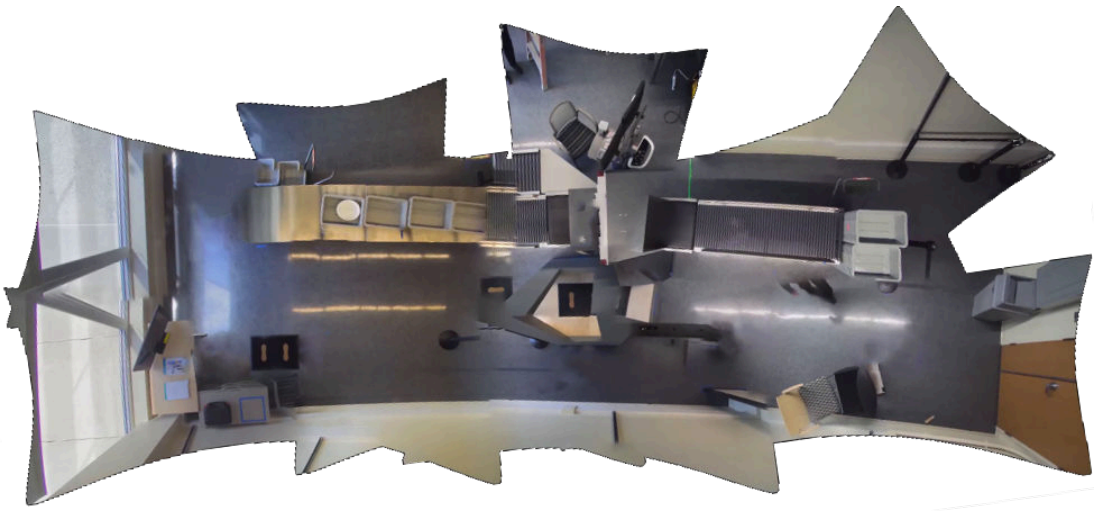


Figure 2.13: Panoramic overview of the camera views at the Kostas Research Institute simulated airport checkpoint.

lance cameras with 1920×1080 resolution and focal length between 3 mm and 9 mm. In these datasets, the passengers interact primarily with their belongings and sometimes with one another. Fig. 2.12 shows the dense camera scenarios where the passengers divest their items onto the roller conveyor (top row) and retrieve their belongings from the belt (bottom row). Additional cameras extend the fields of view (FOV) (Fig. 2.13) of the camera network so that we can track the passengers as they traverse the checkpoint. In terms of perspective and target density, CLASP2 is more challenging than CLASP1. In the CLASP datasets (see Table 3.1), a total of 146 passengers carrying 126 baggage items (*backpack*, *handbag*, and *suitcase*) leave and re-enter the FOV several times. The total number of annotated frames for classes *person* and *bag* is $5,237$, and the number of manually annotated bounding boxes is $15,482$.

2.6.3 Synthetic MNIST-MOT and Sprites-MOT

The synthetic MNIST-MOT and Sprites-MOT datasets, which include most of the common challenges observed in the multi-object tracking (MOT) task, have been

proposed in [41]. In [17], we reproduce both datasets containing 20 video sequences for each set. Each video sequence consists of 500 frames of size 128×128 (pixels) containing 28×28 (pixels) objects (8 digits or 4 sprites) having variable object density and target birth probability of 0.5. Since we choose the object's velocity and direction, the ground truth information is readily available and these datasets can be easily used for training MOT algorithms. The main benefit of using synthetic data is that we can control the target density, the total number of tracks (977 in MNIST-MOT and 983 in Sprites-MOT), occlusions, motions, and appearance, to model and train new MOT algorithms. Fig. 2.14 shows the sequence of the modified version [17] of MNIST-MOT (top row) and Sprites-MOT with an object density of 5.

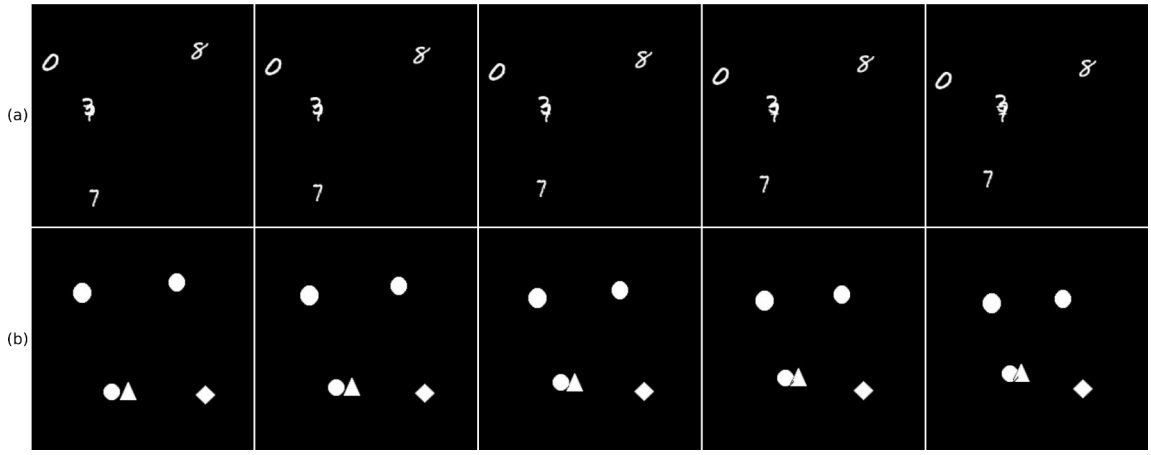


Figure 2.14: Examples of synthetic multiple object tracking datasets: a) MNIST-MOT b) Sprites-MOT.

2.6.4 Multi-species Fruit Flowers

The multiple-species flower datasets first proposed in [25] include various challenges such as variations in illumination, camera angles, brightness, and the presence of substantial occlusions (mostly by tree branches) and background clutter. These datasets aim to encourage the development of generalized models for diverse un-

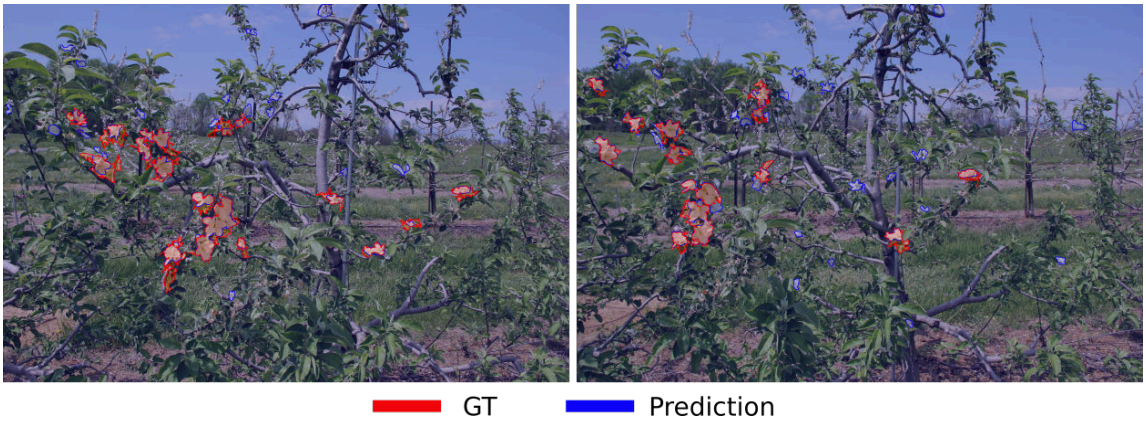


Figure 2.15: Examples of the segmentation labels and model predictions for a fruit flower datasets.

controlled environments. The publicly available flower datasets were collected in a United States Department of Agriculture (USDA) orchard and contain four sub-sets: AppleA, AppleB, Peach, and Pear. AppleA and AppleB are composed of apple trees and flowers under sunny conditions. AppleA is a collection of 147 images of size 5184×3456 (pixels) where 100 randomly selected frames are used for training and the remaining 30 images out of 47 for testing the models. The AppleB, Peach, and Pear datasets contain 18, 48, and 36 images of size 2704×1520 (pixels), primarily for testing the models. Even though these datasets are promising for testing models, the training set had pixel label inaccuracies that we address in [16]. Fig. 2.15 shows the sample manual pixel labels (red) and the corresponding model predictions (blue), which indicate several incorrect pixel labels in the original version of the training set. The techniques proposed in [16] also augment the training frames to train deep CNN models. The SSL method in [16] generates accurate panoptic pseudo-labels shown in Fig. 2.16 for unlabeled datasets. The training set is considered unlabeled images to train the previously initialized CNN models. Thus, we apply these datasets for learning both supervised/semi-supervised and self-supervised techniques.

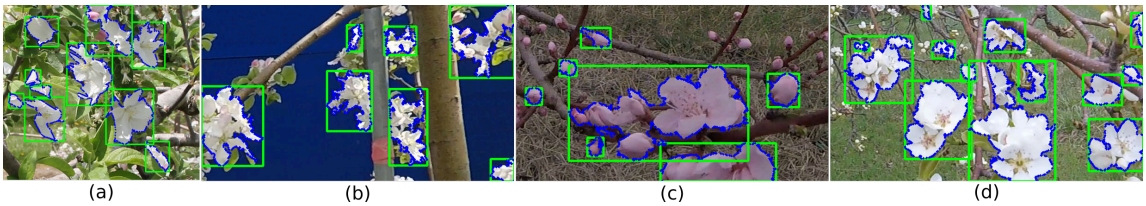


Figure 2.16: Examples of SSL pseudo-labels for multi-species unlabeled fruit flower datasets: a) AppleA test, b) AppleB, c) Peach, and d) Pear.

2.7 Evaluation Measures

Evaluation measures for different downstream computer vision tasks usually assess the quality of the model on validation or unseen test sets. The design of the evaluation measures depends on the type of downstream task model.

2.7.1 Multiple Object Tracking and Multiple Object Detection Measures

MOT and Multiple Object Detection (MOD) evaluation measures were first proposed in 2006 (CLEAR Workshop [73]) to measure the precision in determining the location of objects and the 2D or 3D tracker consistency over time. For MOT, Multiple Object Tracking Precision (MOTP) measures the tracker's ability to estimate the correct locations, and Multiple Object Tracking Accuracy (MOTA) quantifies the tracking performance on counting the number of accurate objects and the corresponding consistent trajectories, i.e.,

$$\begin{aligned} MOTP &= \frac{\sum_{i,t} d_{i,t}}{\sum_t c_t}, \\ MOTA &= 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t}, \end{aligned} \tag{2.21}$$

where c_t is the number of matches found at time t and $d_{i,t}$ is the distance between a visible object o_i and its corresponding tracking hypothesis. The number of missed detections, false positives, mismatches, and total visible objects at time t are represented by m_t, fp_t, mme_t, and g_t, respectively. The mismatch errors at t indicate wrong associations at t based on the correct mapping at $t-1$. Similarly, Multiple

Object Detection Precision (MODP) computes the spatial overlap between ground truth detections and model outputs, i.e.,

$$MODP(t) = \frac{\sum_{i=1}^{N_{\text{mapped}}^t} \frac{\left|G_i^t \bigcap D_i^t\right|}{\left|G_i^t \bigcup D_i^t\right|}}{N_{\text{mapped}}^t}, \tag{2.22}$$

where G_i^t, D_i^t, and N_{mapped}^t represent the sets of ground truth and detected object bounding boxes, and the total number of associated objects at t, respectively. Again, Multiple Object Detection Accuracy (MODA) estimates system accuracy using the number of missed detections m_t and false positives fp_t, without considering the temporal identifiers of the targets, i.e.,

$$MODA(t) = 1 - \frac{m_t + fp_t}{N_{\text{G}}^t}. \tag{2.23}$$

2.7.2 Semantic Segmentation Measures

Unlike bounding box-based association, which measures detection performance, or Region of Interest (ROI) mask-based matching, which estimates instance segmentation performance, semantic segmentation measures use pixel level matching [74] to quantify the semantic model accuracy. Like classification or detection tasks, precision (Prcn) and recall (Rcll) in semantic segmentation indicate how many returned results are relevant and how much existing relevant information is returned. The F-measure (F_1), which is given by the geometric mean of precision and recall, computes the overall performance of the method. The most widely used measure to assess the correct pixel-level prediction accuracy is the mean Intersection over Union (mIoU). In multi-class prediction problems, the intersection between predictions M and ground truth G is divided by the corresponding union and averaged over all the classes $\mathcal{C} = \{1, \ldots, n_c\}$,

$$mIoU = \frac{1}{n_c} \sum_{\mathcal{C}} \frac{|M \bigcap G|}{|M \bigcup G|} = \frac{1}{n_c} \sum_{\mathcal{C}} \frac{TP}{TP + FP + FN} \tag{2.24}$$

where *true positives (TP)* is the total number of matching pixels between M and G, *false positives (FP)* is the total number of incorrectly detected pixels in M, and *false negatives (FN)* is the total number of pixels which are missed by the detector in G.

2.7.3 Multi-Object Tracking and Segmentation Measures

Inspired by the CLEAR MOT [73] measures for multiple objects tracking problems, the popular MOTS evaluation measures assume that each pixel should be assigned to one object of interest. Evaluation measures TP, TN, FP, and FN are popular in multiple object tracking problems and hence for MOTS problems as well [12], which bounding box or instance mask IoU and defined as

$$\begin{aligned} TP &= \{h \in H | c(h) \neq \emptyset\}, \\ FP &= \{h \in H | c(h) = \emptyset\}, \\ FN &= \{m \in M | c^{-1}(m) = \emptyset\}, \end{aligned} \tag{2.25}$$

where $M = \{m_1, \dots m_N\}$ with $m_i \in \{0,1\}^{h \times w}$ is the set of N non-empty ground truth pixel masks which are also assigned to ground truth track id $id_m \in \mathbb{N}$. Similarly for the predicted hypothesis masks, $H = \{h_1, \dots h_K\}$ with $h_i \in \{0,1\}^{h \times w}$ and track id $id_h \in \mathbb{N}$. Here, $c(h)$ is the mapping from hypothesis mask to ground truth mask and is defined as

$$c(h) = \begin{cases} \arg\max\limits_{m \in M} \text{IoU}(h, m), & \text{if } \max\limits_{m \in M} \text{IoU}(h, m) > 0.5 \\ \emptyset, & \text{otherwise.} \end{cases} \tag{2.26}$$

Another mask-based evaluation measure is called soft TP and is defined as

$$\widetilde{TP} = \sum_{h \in TP} \text{IoU}(h, c(h)). \tag{2.27}$$

Thus, the mask-based multi-object tracking and segmentation accuracy (MOTSA) and multi-object tracking and segmentation precision (MOTSP) mimic the box-based

MOTA and MOTP metric,

$$\begin{aligned} \text{MOTSA} &= 1 - \frac{|FP| + |FN| + |IDS|}{|M|} = \frac{|TP| - |FP| - |IDS|}{|M|}, \\ \text{MOTSP} &= \frac{\widetilde{TP}}{|TP|}. \end{aligned} \tag{2.28}$$

where the set of id switches (IDS) is equal to the set of ground truth masks whose predecessor was tracked with a different identifier, i.e,

$$\begin{aligned} IDS = \{m \in M | c^{-1}(m) \neq \emptyset \wedge pred(m) \neq \emptyset \wedge \\ id_{c^{-1}(m)} \neq id_{c^{-1}pred(m)}\}. \end{aligned} \tag{2.29}$$

Again, to measure segmentation performance as well as detection and tracking quality, the sMOTSA measure considers the soft true positive numbers $\widetilde{TP}$ instead of account only for IoU, i.e.,

$$\text{sMOTSA} = \frac{\widetilde{TP} - |FP| - |IDS|}{|M|}. \tag{2.30}$$

2.8 Machine Learning Techniques for Computer Vision

Modern computer vision techniques mostly address real world problems by leveraging machine learning methods as a core component. Machine learning models for computer vision tasks are training using a variety of techniques, including supervised learning [7], self-supervised learning [75], reinforcement learning [76], semi-supervised learning [77], and unsupervised learning [3]. Since machine vision algorithms are data driven, data preparation before starting the training is also crucial and modern vision systems employ sophisticated techniques, such as representative sampling [78, 79, 80], formation of foundation datasets [81, 82], and effective mechanisms for using data augmentation during training [83].

2.8.1 Data Augmentation for Training/Testing

Most deep learning algorithms use data augmentation for model training. Data augmentation techniques address the typical challenges in training deep learning models, such as class imbalance [84], limited data availability [85], overfitting [86], manual labeling cost [87, 88], transfer learning challenges [89], domain adaptation [89], and generalization ability of CNN models [18, 90]. Different types of data augmentation (see Fig. 2.1) based on image operations, such as geometric transformations, image flipping, color space transformation, image cropping, rotation, translation, and noise injection are used in modern CNN model training.

2.8.1.1 Test-time Data Augmentation

In addition to the above applications of data augmentation approaches during training, test-time data augmentation has also been explored to generate more robust predictions. In [91], a knowledge distillation approach was proposed where predictions from multiple transformations of unlabeled data are used to generate new training annotations for self-training the model. Test-time data augmentation is particularly popular in challenging real-world data, as ensembles of the multiple predictions can be used as pseudo-labels to update the model without any modification to the optimization problem or model structure. Data distillation as a test-time data augmentation is also popular in student-teacher [14] (Semi-SL) learning strategies. Again cross-modal distillation addresses the problem of limited labels of a particular modality using data distillation [91], which generates pseudo-label annotations from a light-weight ensemble prediction of multiple data transformations. This strategy is widely used in semi-supervised methods as a self-training mechanism for object detection tasks [14, 91, 77]. It consists of four steps: i) train an initial model using labeled data, ii) apply the trained model on multiple transformations of unlabeled

data, iii) generate pseudo-labels from the ensemble predictions, and iv) update the model using both labeled and pseudo-labeled data.

2.8.1.2 Data Augmentation for Downstream Task Improvement

Multiple views from overlapping multi-camera networks or multiple geometric transformations of the same image sometimes provide adequate data for Semi-SL or SSL, as we can aggregate the same ROI predictions into a common coordinate frame. In [92], two unique learning algorithms use different views of data, and the supervised signals of each algorithm come from the other algorithm's predictions on the new unlabeled data. In [93], predictions of randomly transformed inputs and the original inputs are compared in terms of the transformation's stability loss to learn the consistency between the outputs from the transformed inputs. Multi-view geometry is also employed in [19] to generate keypoints pseudo-labels from multiple cameras and update the detector iteratively. The main goal of using information from multiple views is to reduce the impact of occlusions and appearance changes caused by out-of-plane rotations of the target objects on noisy detections generated by a single camera.

Test-time data augmentation can incrementally increase the prediction confidence and reduce the prediction uncertainty of a model. In [94], test-time rotation augmentation, remapping, and majority voting reduce the semantic uncertainty of geometrically distorted objects in a multi-camera network and improve the semantic prediction quality. In [20], the model retraining approach uses an ensemble of predictions from the original and flipped inputs to identify uncertain image regions as training signals. In [88], a framework for medical image segmentation uses test-time data augmentation to estimate the prediction uncertainty using spatially transformed (flipping, scaling, rotation) prediction distributions. The estimated uncertainty reduces the impact of highly confident wrong predictions, outperforming a single-prediction

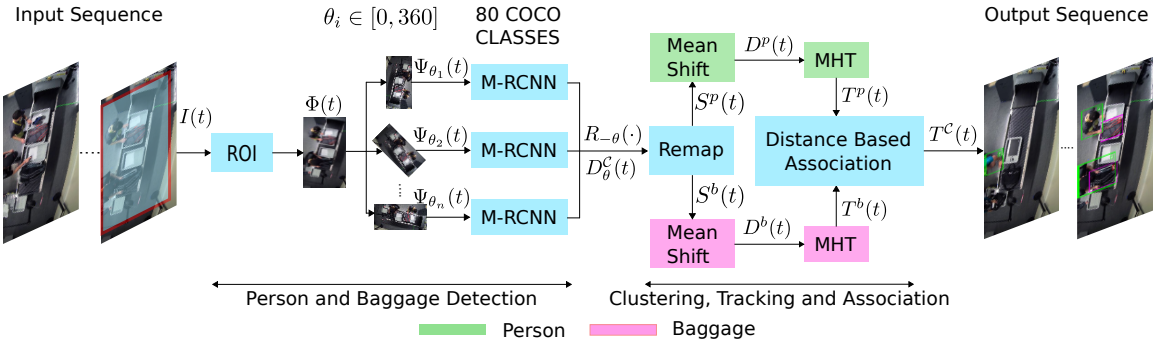


Figure 2.17: Example of test-time data augmentation-based tracking-by-detection framework. The detection stage uses multiple rotated versions of the region of interest to increase the probability of detection. The clustering, tracking, and association stage maps the rotations back to the original coordinate system, clusters nearby detections using the mean shift algorithm, and performs temporal association using MHT. The association between objects is then performed using a simple distance-based approach.

baseline model. In [18], as shown in Fig. 2.17, the detections from multiple rotated inputs are combined to improve inference performance during test-time. This test-time augmentation approach is then used to enhance the performance of the corresponding downstream tasks such as clustering, tracking, and association.

2.8.1.3 Data Augmentation for Backbone Feature Improvement

Other data augmentation methods [3, 95] use spatial properties of images to enhance the pre-trained backbone features of the input images, indirectly improving the downstream task performance. In [95], a CNN model uses visual similarity across object parts and images to identify the position of a randomly selected patch within one image from another. The representation learning method proposed in [3, 96] trains a CNN model to predict 2D rotation transformations that must be applied to an image (Fig. 2.18) for correct classification. Contrastive embeddings learning techniques [97, 98, 99] have become popular for feature learning in downstream image recognition tasks. The main goal of using contrastive predictive coding in feature

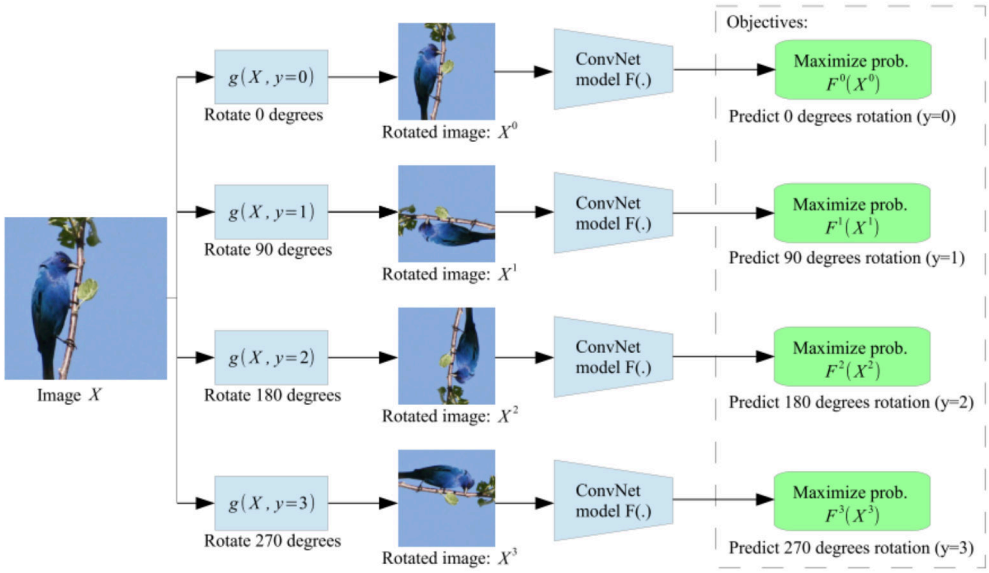


Figure 2.18: Visualization of data augmentation learning technique [3] to improve the backbone representation.

learning is to maximize the mutual information across features at different spatial locations [98] and across different views [99] of the same input image. In [40], the classification of images in the ImageNet dataset employs test-time augmentation. In that method, the final output is the average prediction on ten randomly cropped patches.

Based on the downstream vision tasks, test-time augmentation approaches need to consider some sophisticated method to aggregate predictions from transformed images. It is easy to aggregate warped image predictions for the classification task since the inverse geometric transformation is readily available. However, for segmentation and detection tasks, the aggregation of geometrically transformed image predictions is challenging since the inverse transform may distort the spatial [18, 100] (Fig. 2.17) and semantic relations [16]. For example, in [14], the test-time augmented predictions include scaling, flipping, color transformation, and bounding box jittering and regression to generate pseudo-labels for retraining.

2.8.2 Self-Supervised Learning

SSL techniques usually employ a pre-trained teacher model to automatically generate training samples from unlabeled data during test time in a multi-stage approach [18] or during training in a single-stage strategy [14]. The idea of self-supervision has been widely employed in recent works to use extensive unlabeled vision data for deep learning with the help of existing training samples [101]. Several strategies exist to train a deep CNN model using automatically generated pseudo-labels. The most common approach is representation learning for classification tasks [21] and then transferring the model to other downstream tasks [39, 13]. Contrastive Predictive Coding [102, 90] is another SSL technique where training uses multiple augmented copies (spatially or geometrically) to predict another transformed version of the same input. Overall, effective data augmentation strategies [87, 88] are essential for SSL.

2.8.2.1 Self-supervision for Detection and Segmentation

There are several ways to apply the SSL strategy for downstream tasks on unlabeled datasets. In pretext task learning methods [75], self-supervised image features are transferred to the downstream task predictions. In other methods, [14, 103, 43], the downstream task models are fine-tuned using automatically generated pseudo-labels. For the detection task [14], Mengde et al. proposed a data-dependent self-supervision technique where a student-teacher interactive learning strategy employs classification as a pretext task, and also refines pseudo-labels using box-jittering as a data augmentation method. The main limitation of applying these methods to unlabeled datasets is that the pseudo-labels may become noisy, which requires sophisticated filtering/refining mechanisms. SSL is also used to estimate instance uncertainty

and fine-tune the model using an uncertainty-aware rotation invariance learning technique [18].

In semantic segmentation applications [15, 103, 13, 43, 22], SSL approaches usually estimate the uncertain regions of the image and use them to produce pseudo-labels to update the model. In [15], a weakly supervised classification model is used in an active learning framework to generate pseudo-pixel labels for self-training. The uncertainty-aware semantic pseudo-label refinement technique proposed in [104] focuses on the low-confidence uncertain classes using a feature alignment method between a source and a target domain and an uncertainty-aware pseudo-label assignment strategy. The self-consistency method based on equivariant transformations proposed in [103] uses multiple inference uncertainties to select the most uncertain image patches for additional training. In panoptic segmentation [9], the weakly supervised technique proposed in [105] uses image-level tags and bounding boxes as weak supervision signals during training. A self-supervised approach is presented in [16] to estimate panoptic pseudo-labels with accurate contours and then update the model iteratively using unlabeled frames.

2.8.2.2 Self-supervision for Multi-Object Tracking

Recently, multi-object tracking applications employ self-supervision methods where pseudo identities are used to update the tracker embeddings or ReID model [106] as well as the detector [18] in a tracking-by-detection paradigm. Recent multi-view feature learning [4, 107, 108, 109] and data association methods [110] use SSL techniques to better address the challenges of occlusion, perspective distortion, camera synchronization, as well as scale, appearance, and viewpoint variations in camera networks. SSL is used in [4] to increase multi-view association accuracy using multi-view appearance learning by leveraging motion tracking and camera geometry. For example, Fig. 2.19 shows the sub-spaces corresponding to the same person across

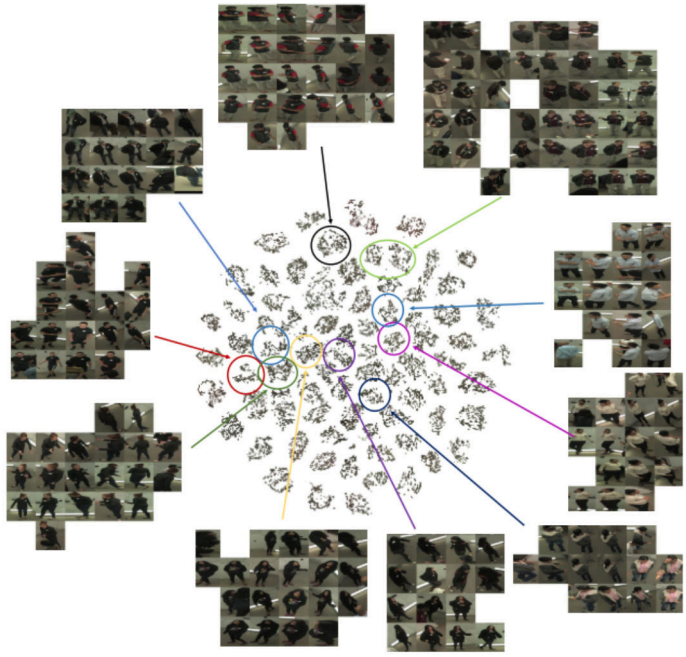


Figure 2.19: Sub-spaces from self-supervised appearance learning [4] represent the target similarity across views.

multiple views where the clusters are constructed using the embeddings from the SSL appearance model. The temporal identifiers of all the clusters represent the pseudo-labels used to train an MOT algorithm. In these methods, self-supervision automatically mines single-view and multi-view triplets to update the appearance model iteratively. In [110], a self-supervised spatio-temporal association model iteratively learns the spatial association similarity across views and the temporal association similarity in a single view. Other approaches resort to object invariant representation learning [108] across views where an instance classification model recognizes objects in one perspective by using the corresponding image embeddings as prototypes from another perspective. In [107], self-supervised latent subspace learning uses classification tasks for single-view representations and self-expressive layer-based spectral clustering for cross-view relations. In robotics applications, self-supervision is employed for 6D pose estimation [111], where a self-supervised segmentation model segments ob-

jects from multiple perspectives and merges the multi-view results into a 3D object model. Self-supervised multi-view representation learning uses temporal alignment [112] between video frames to generate temporal correspondence-aware multi-view embeddings, which improves downstream recognition.

2.8.2.3 Self-supervision for Reinforcement Learning

Reinforcement learning (RL) simultaneously solves the representation learning [113], task optimization, and reward estimation problems. Typical RL techniques [114] directly use the reward information or ground truth state to learn a successful policy. However, Shelhamer et al. [76] introduced a self-supervised technique to improve representation and policy optimization where the RL procedure is augmented using auxiliary multi-task losses. Sermanet et al. [109] proposed an SSL technique for feature learning from unlabeled videos (Fig. 2.20), where the anchors, positive, and negative targets represent the multi-camera scenarios. Dwibedi et al. [115] proposed a multi-frame representation learning where the reward function compares the learned representation with human demonstrations in a vision-based robotic system.

2.8.2.4 Self-supervision in Contrastive Learning

SSL is widely employed in different domains, such as natural language processing (NLP) [116], time-series prediction [117], and speech recognition [118]. In these contexts, self-supervised pre-training has masked words or time series segments which can be easily predicted from the remaining data points. Typical SSL algorithms use labeled and unlabeled data, similar to knowledge distillation approaches [91] or unsupervised model adaptation techniques [20], which fall into the category of semi-SL strategies. We propose an SSL framework [18] inspired by [39, 103, 20, 104, 119] which uses only unlabeled datasets. In this learning technique, the pre-trained model must be aware of the classes of interest in the new domain. Fully self-supervised methods

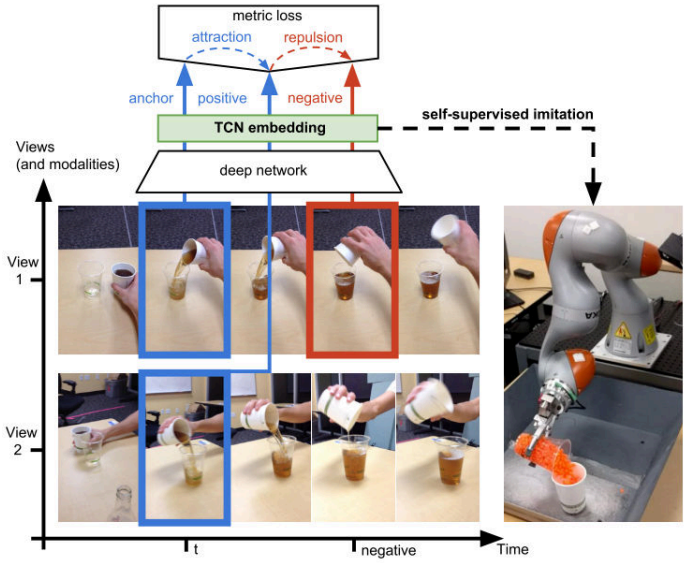


Figure 2.20: Self-supervision for robotics imitation [109] using triplet-loss where the positive and negative anchors are collected from the simultaneous viewpoints.

have no manual supervisory signals in iterative learning, so there is a chance that the resulting model is heavily biased. In these methods, the most challenging step is carefully selecting and refining the pseudo-labels to update the models. However, unlike [39], instead of resorting to multi-task strategies to guide the learning process, we employ a multi-inference approach similar to the self-consistency method based on equivariant transformations proposed in [103]. Differently from [103], rather than using multiple inference uncertainties to select image patches for additional training, our method aggregates these multiple inferences into accurate pseudo-labels to refine the model (Fig. 3.2). Our approach differs significantly from unsupervised model adaptation techniques [20], and knowledge distillation approaches, [91] in that we only use automatically generated labels and avoid human annotations altogether during model update.

2.8.3 Multi-task Learning

Multi-task learning usually improves model performance by learning shared representations across different tasks [5]. As long as the tasks are similar, the model tends to generalize better to unseen data [120]. For example, the recently introduced panoptic segmentation approach [9] jointly learns the closely related tasks of instance and semantic segmentation. Currently, it represents the state of the art in instance and semantic segmentation [121, 122, 105, 9]. Multi-task uncertainty-aware learning [120] automatically assigns low weights to a task that does not contribute to the desired representation. In Fig. 2.21, a multi-task loss function using maximum

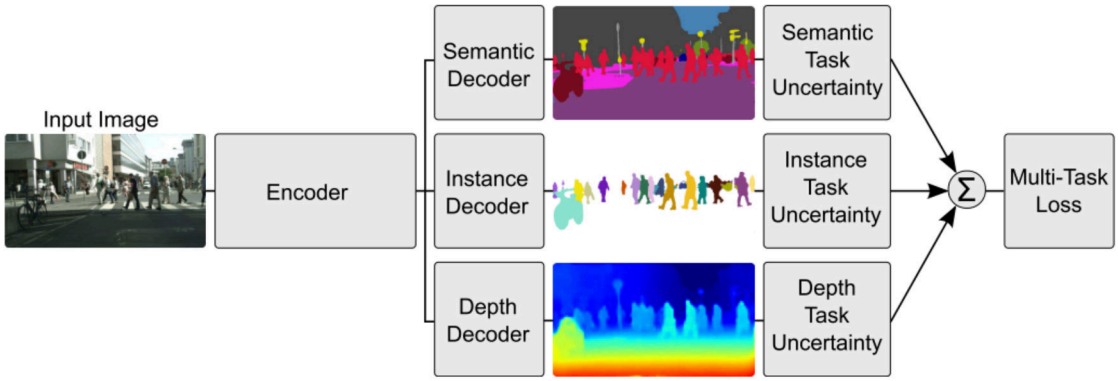


Figure 2.21: Uncertainty-aware multi-task learning for single input multiple output training [5].

log-likelihoods with task-dependent uncertainties guides a single input encoder to predict multiple tasks. In general, epistemic uncertainty captures what is unknown to the model due to lack of sufficient training data. Using data augmentation or large training datasets, the epistemic uncertainty can be reduced but the uncertainty related to the contribution of individual tasks necessitates a different type of learning strategy, which is called task-dependent uncertainty learning. Typically, the weights for multiple tasks are assigned arbitrarily based on the anticipated importance of the

task which is not optimal, i.e.,

$$\mathcal{L}_{\text{total}} = \sum_{i=0}^{n} w_i l_i, \tag{2.31}$$

where l_i is the loss corresponding to the i-th task, and w_i is its corresponding arbitrarily assigned weight. In [5], the issue of determining optimal weights for multiple tasks has been addressed for single-input multiple-output models where the output tasks y_1 and y_2 are modelled as Gaussian distributions, according to

$$\begin{aligned} p(y_1, y_2|f^{\mathcal{W}}(x)) &= p(y_1|f^{\mathcal{W}}(x)) \cdot p(y_2|f^{\mathcal{W}}(x)) \\ &= \mathcal{N}(y_1; f^{\mathcal{W}}(x), \sigma_1^2) \cdot \mathcal{N}(y_2; f^{\mathcal{W}}(x), \sigma_2^2), \end{aligned} \tag{2.32}$$

where $f^W(x)$ is the model output for input x, and W, σ_1, σ_1 are trainable parameters. Unlike traditional multi-task learning approaches [120, 10], which focus on weighing the contributions of several *homogeneous outputs*, In [17], we weighs the contribution of multiple *heterogeneous input* features. Thus, we learn the relative weights of each loss function term based on the uncertainty of distinct features. Our clustering-based MOTS approach [17] applies an instance segmentation model to extract multi-task features for uncertainty-aware embeddings learning. Thus, our proposed unsupervised technique optimizes multiple tasks from sequential data so that similar data points become closer and different points are further separated.

2.9 Multiple Object Tracking and Segmentation

Tracking-by-detection approaches for MOT rely on robust detection and data association techniques. Recent MOT approaches employ probabilistic algorithms, such as Kalman filters [123] or particle filters [124, 125], to predict the position of new observations and associate the detections using a bipartite graph matching algorithm [126]. The multiple hypothesis tracking method [123] also considers the deep appearance feature of the detections for MOT. However, tracking methods based on local data association struggle to handle occlusions and noisy detections. Some

Figure 2.22: Qualitative results from MOTChallenge validation dataset.

global or delayed optimization approaches [127, 128] have been proposed to tackle these challenges. The global optimization-based tracker introduced in [127] estimates the temporal identifiers after all the detections have been obtained. Even for delayed optimization techniques, association failures in approaches based on bounding box matching may be severe due to target localization errors. Multiple Object Tracking and Segmentation (MOTS) algorithms localize objects precisely and establish temporally consistent associations among segmentation masks of multiple objects observed in different video sequence frames (Fig. 2.22). Most state-of-the-art MOTS methods [26, 12] employ supervised learning approaches to generate discriminative embeddings of the objects and then apply feature association algorithms based on sophisticated target behavior models [41, 27, 129, 130].

2.10 Clustering-based Data Association

Subspace clustering is an unsupervised learning technique where data points are mapped to lower dimensionality subspaces and clustered based on feature similarity. Existing clustering methods employ two common strategies: i) extract low-dimensional discriminative features, ii) apply a robust clustering approach to partition the subspaces. Earlier approaches used methods based on factorization strategies [131, 132, 133] or kernels [134, 135, 136, 137] to separate data points into their re-

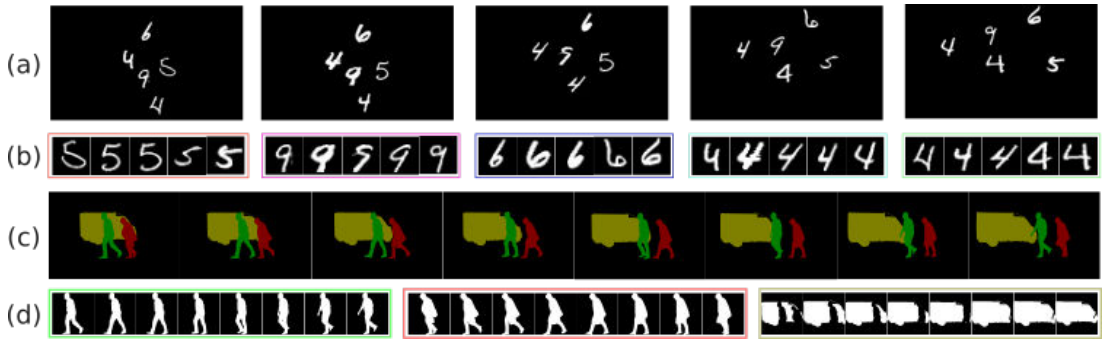


Figure 2.23: Sequence of frames for (a) MNIST-MOT with $|O| = 5$ targets per frame and (c) DAVIS (cropped for two pedestrians and one car) sequences. (b) and (d) show the corresponding subspaces generated by constrained k-means [145] in DHAE [17].

spective subspaces. More recent methods employ CNNs [138, 139, 140], or generative adversarial networks [141, 142], oftentimes in conjunction with self-expressive layers [139, 142, 143]. The autoencoder-based DSC-Net [139] uses fully connected layers to learn an affinity matrix that enhances the discriminative property of the embeddings. Some autoencoder-based techniques consider subspace reconstruction and cluster assignment errors for better sample distribution [144, 138].

Although existing approaches may produce discriminative features to cluster static data, these features must be sufficiently distinctive to identify the subspaces corresponding to multiple objects in sequential data due to significant location and appearance variations across frames. Clustering multiple objects in sequential data is an under-explored problem, particularly in real-world applications. While existing temporal clustering methods, such as ordered subspace clustering (OSC) [146] consider sequential data, they focus on clustering entire video frames without taking into consideration the spatial aspect, which must be addressed when multiple objects are present in a video segment. Few methods [147, 148] address the problem of clustering objects over video sequences, which must consider that object features may change over time [124, 125, 149]. In [17], we proposed a Deep Heterogeneous

Autoencoder (DHAE) model that generates more effective latent representations of sequential multi-object data. Since the dimensionality of our embedded feature does not depend on the number of clusters, our method does not suffer the scalability issues present in most clustering methods based on self-expressive layers [143]. We also use prior knowledge regarding the sequential data in the form of a constraints graph [145] to improve clustering robustness in sequential data. As shown in Fig. 2.23, the subspaces contain similar shapes for both synthetic (second row from top) and real (bottom row) data. Thus, our unsupervised learning approach does not require explicit cluster assignment learning [138].

2.11 Multi-View Data Association

Multi-camera tracking (MCT) systems require sophisticated trajectory association mechanisms to maintain the temporal identifiers of targets across camera views [28, 150, 151]. Even within a single camera view, temporary occlusions must be addressed using similar strategies [152, 153]. Most trajectory association approaches compute trajectory similarity scores based on a combination of dataset-specific appearance, and motion features [28, 150, 151, 152, 153]. Some methods use camera calibration information to project tracks from cameras with significantly overlapping fields of view onto a common plane and perform association using occlusion modeling [11, 37] or Re-ID techniques [154, 155, 106, 156, 157]. Learning these features requires the availability of a large number of long, continuous trajectories [11], which are difficult to obtain with typical ceiling-height overhead cameras due to their limited fields of view. Multi-camera methods may also need to project trajectories onto a global 3D coordinate frame, which requires camera calibration information. The dependency on camera calibration further limits the applicability of these methods to security systems. Obtaining such information would introduce additional barriers to adoption since calibrating multiple cameras with partially overlapping fields of view

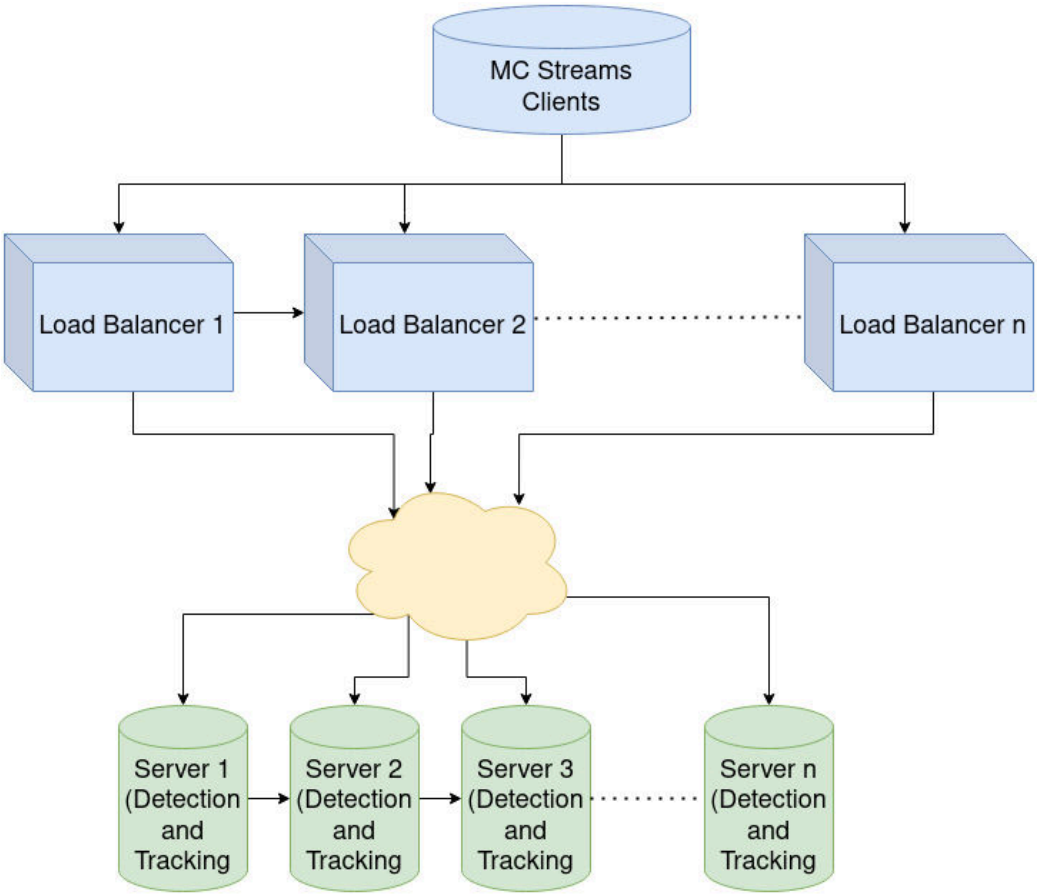


Figure 2.24: Distributed multi-camera tracking system.

is a complex task [158, 159, 160, 161].

2.12 Real-Time Multi-Camera Tracking

Tracking multiple objects either in a single camera or multi-camera networks [100, 18] is a computationally expensive task since a tracking system needs to process a significant number of high-resolution RGB frames simultaneously – e.g., a distributed multi-camera tracking system can handle multiple streams using a load balancer to distribute the incoming data frames to multiple servers. The servers process the data and perform detection and tracking. The load balancer can also collect the multi-camera tracking results to send back to the clients. This approach is scalable as we can

add or remove servers based on the network size and multi-camera streams. Real-time tracking is a challenging problem when the multiple camera views are highly dense [11]. Existing deep neural network-based multi-camera tracking approaches use the tracking-by-detection method [36, 109] where the computed 2D detections are used to generate single camera track hypotheses, and the tracklets are matched across cameras using camera geometry and deep object features (i.e., embeddings) to identify the same targets across views. Some deep learning methods also proposed mapping the 2D locations to 3D representations [11, 162] to tackle the challenges of occlusions, high target density, and background clutter. In these methods, the 2D occupancy map or the 3D occupancy volume are key features to find the matching across multiple views.

In terms of computational complexity, several methods [162, 163] address tracking performance and execution time, which is a crucial requirement for a real-time system. To achieve real-time performance in scalable camera networks, the system proposed in [163] uses low-resolution vehicle surveillance video feeds with an off-the-shelf real-time single camera detector [164, 54] and tracker [165], with each camera initialized as a GPU process to handle the computation. Since the cameras rarely have fully overlapping fields of view, they mainly focus on appearance models across views by proposing self-supervised attention-based Re-ID modules where frame-level deep features of the tracks are temporally weighted and averaged to compute a single appearance descriptor. Finally, these methods employ a hierarchical clustering approach on the computed similarity matrix to identify the tracks representing the same identity across the views. In [162], the ground points of the targets (i.e., the points where the bounding boxes of the targets touch the ground plane) are predicted using a supervised perspective-aware deep ground point network. Point detection methods, such as [166] and [167], estimate the 2D corners of the target's bounding box or its centroid. Location heat maps of the targets are projected onto a

common plane using homographies. Then a simple target classifier model is designed to detect persons in real-time assuming a fixed scale from an overhead view. These real-time multi-camera tracking methods depend on camera calibration and manual labels to train lightweight overhead detectors in camera networks with overlapping fields of view or on low video resolution to speed up the computations while maintaining tracking robustness. However, in some scenarios real-time MCT algorithms may also need to process high-resolution video feeds [18, 163]. These methods leverage real-time single camera 2D detectors [23, 54], which serve the dual purpose of detection and appearance modeling for both single camera tracker (SCT) and Re-ID using models based on homographies [100] or matching feature [168] across views. Some works [18] also extend bounding box IoU-based ReID in SCT using temporal distances and local ReID search areas around candidate target locations.

CHAPTER 3
SELF-SUPERVISED LEARNING FOR MULTIPLE OBJECT DETECTION

In this chapter, we propose a novel Self-Supervised Learning (SSL) technique to provide the model information about instance segmentation uncertainty from overhead images. Our SSL approach improves object detection by employing a test-time data augmentation and a regression-based, rotation-invariant pseudo-label refinement technique. Our pseudo-label generation method provides multiple geometrically-transformed images as inputs to a Convolutional Neural Network (CNN), regresses the augmented detections generated by the network to reduce localization errors, and then clusters them using the mean-shift algorithm. The self-supervised detector model can be used in a single-camera tracking algorithm to generate temporal identifiers for the targets.

3.1 Introduction

Automated video surveillance requires the detection, tracking, and recognition of objects of interest in a scene. Accurate and precise surveillance in crowded scenes is one of the most challenging computer vision applications. To address the problem of visual surveillance in the domain of airport checkpoint security, the Department of Homeland Security (DHS) ALERT (Awareness and Localization of Explosives-Related Threats) center of excellence at Northeastern University initiated the CLASP (Correlating Luggage and Specific Passengers) project. This initiative aims to help the Transportation Security Administration (TSA) detect security incidents, such as theft of items and abandoned bags.

Current approaches for detecting and tracking passengers and luggage in airport checkpoints divide the image area within each camera's field of view into regions of interest where certain passenger behaviors are expected (e.g., passengers divest

their items near the roller conveyor) [169, 170]. While these approaches are effective within individual regions of interest, they cannot detect and track passengers and their belongings throughout an entire checkpoint. Moreover, most recent detection algorithms [6, 171, 7, 172] are unable to detect multiple objects in realistic overhead camera scenarios due to the unavailability of large-scale datasets obtained using unconventional camera perspectives.

Fine-tuning pre-trained models using human annotated labels is a common approach in computer vision methods. However, this strategy hinders the applicability of state-of-the-art algorithms in scenarios where images are obtained from perspectives that are not commonly observed in existing publicly available datasets. The dramatic variability of video surveillance systems used in airport checkpoints would require deployment-specific fine-tuning of models, and in some scenarios, even camera-specific adjustments. To overcome this challenge, we leverage the fact that models pre-trained on large-scale datasets can build upon their initial predictions to adapt to new scenarios using SSL strategies. Our proposed SSL framework obviates the tedious and expensive human annotation procedure by automatically generating pseudo-labels to update the model.

To generate pseudo-labels, we cluster multiple detections obtained from geometrically transformed images using the mean-shift algorithm [173]. Each cluster corresponds to the detection of one object observed at different orientations on several augmented input images. The cluster modes with the corresponding bounding boxes, segmentation masks, and confidence scores are used to update the model. Thus, our model learns from rotation-invariant pseudo-labels and can be integrated with a tracking-by-detection algorithm [23] to generate accurate target tracklets from overhead perspectives.

Our SSL algorithm is inspired by the methods described in [39, 103, 20, 104, 119]. However, unlike [39], instead of resorting to multi-task strategies to guide the

learning process, we employ a multi-inference approach similar in spirit to the self-consistency method based on equivariant transformations proposed in [103]. Our method differs from [103] in that, rather than using the uncertainties from multiple model predictions to select image patches for additional training, it aggregates multiple inferences into accurate pseudo-labels that are used to refine the model. Our method departs significantly from unsupervised model adaptation [20] and knowledge distillation approaches [91] in that we only use automatically generated labels and avoid human annotations altogether during model update.

Contributions. The key contributions of this chapter are:

- A novel self-supervised object detection algorithm that generates pseudo-labels based on instance segmentation uncertainties.
- A new data augmentation and regression-based clustering mechanism that substantially improves the quality of pseudo-labels for self-supervised training.
- We provide an extensive evaluation of our methods on a dataset collected using multiple overhead cameras in a realistic airport checkpoint scenario.
- Our SSL models and the corresponding source code are available at `https://github.com/siddiquemu/SCT_MCTA`.

3.2 Related Work

Multiple target tracking using camera networks is an active research topic with several potential applications [174, 175, 176, 177]. Most works on camera networks focus on the multi-camera aspect of the problem and do not consider the challenges associated with camera perspectives. Although generic object tracking algorithms could be used in surveillance systems (e.g. [178, 124, 179]), when object categories are known, trackers based on specialized detectors are more accurate and less prone to model drift [180, 181]. This observation has led to the development of a variety of multiple target tracking algorithms that specialize in tracking humans [182, 183,

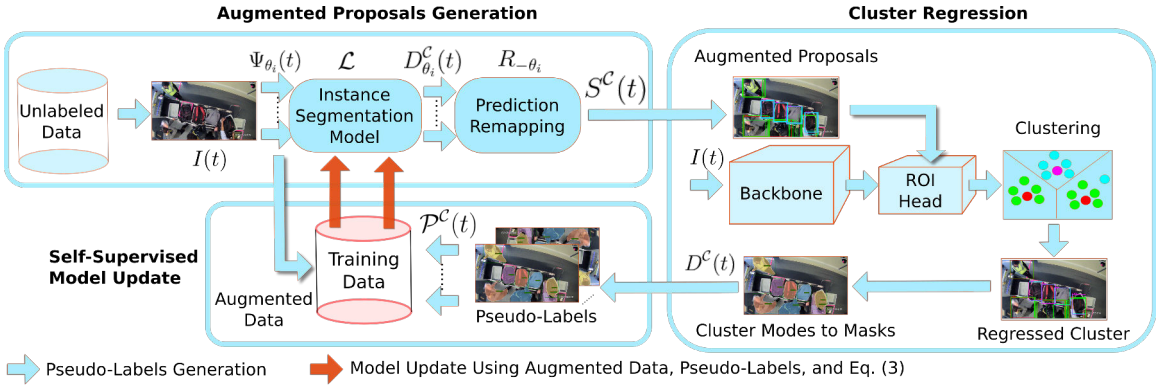


Figure 3.1: Proposed SSL framework. The augmented proposal generation stage uses multiple rotated versions of the unlabeled input images to generate augmented detections from an instance segmentation model and then remaps these predictions into their original coordinates. The clustering algorithm leverages the model's regression ability to reduce localization errors using the augmented predictions as region proposals. The regressed cluster modes are then used to generate augmented pseudo-labels to update the model.

184, 185, 186, 187, 36, 11, 37, 157, 155] or vehicles [156, 154, 106, 188]. However, in many scenarios, it is desirable to track additional objects of known categories. In these cases, more flexible detection algorithms are needed, but the effectiveness of modern object detection models is highly dependent upon the characteristics of the training datasets [171, 6, 7].

Previous works have used SSL techniques to improve visual feature learning [13, 21, 90], reducing dependency on human annotations for training backbone models. However, transferring knowledge from pre-trained backbones to downstream tasks is a far less explored topic. Unlike our proposed approach, SSL techniques for detection [39, 20] and semantic segmentation [103] rely on annotations to initialize the model before iterative learning can take place.

Data augmentation is an effective mechanism to improve the robustness of CNNs in scenarios not available during training [91, 189], but little attention has been given so far to approaches for combining the response of the network to augmented samples. In multi-target tracking applications, multiple detections mapped

to a common coordinate system can be interpreted as the probability of occupancy of the area observed by the cameras [100]. Although it is possible to use clustering techniques to map the modes of this distribution to unique target detections, bounding box alignment errors pose a challenge to the generation of high-quality pseudo-labels for SSL. Hence, we propose a test-time regression technique that leverages instance segmentation information for pseudo-label generation.

3.3 Proposed Model

Our system consists of two main components: i) a detection algorithm trained using SSL and ii) a multi-camera tracking-by-detection mechanism. A single-camera tracking algorithm uses SSL detections to generate tracklets for passengers and baggage items. We then employ a novel multi-camera target trajectory association algorithm to uniquely identify passengers throughout the checkpoint.

3.3.1 Self-Supervised Learning

We use the PANet model [7] with a ResNet-50 backbone [190, 191] as the baseline detector. Since the categories of interest are persons and their belongings, we use a model pre-trained on the COCO dataset [24], which includes object classes related to these categories (i.e., person, handbag, backpack, and suitcase). Because the COCO dataset consists mostly of images captured at roughly eye-level, detectors trained using that dataset do not perform well on overhead perspectives. To address this limitation, our SSL framework updates the baseline model using rotation-invariant pseudo-labels. As Fig. 3.1 shows, our SSL framework consists of three main steps: i) augmented region proposals generation, ii) pseudo-label generation and refinement through cluster regression, and iii) iterative model update.

Algorithm 1 Augmented Proposals Generation

1: **function** AUGMENTEDPROPOSALS($I(t)$, r)
2: $S^{\mathcal{C}}(t) = \emptyset$, $\Theta = \{i \cdot \Delta\theta\}_{i=1}^{r}$
3: **for** $\theta_i \in \Theta$ **do**
4: $\Psi_{\theta_i}(t) = R_{\theta_i}(I(t))$
5: $D^{\mathcal{C}}_{\theta_i}(t) = D_{\text{PANet}}(\Psi_{\theta_i}(t))$
6: $S^{\mathcal{C}}_{\theta_i}(t) = R_{-\theta_i}(D^{\mathcal{C}}_{\theta_i}(t))$
7: $S^{C}(t) = S^{C}(t) \cup S^{\mathcal{C}}_{\theta_i}(t)$
8: **end for**
9: **return** $S^{C}(t)$
10: **end function**

3.3.1.1 Augmented Proposals Generation

Our data augmentation method, summarized in Alg. 1, uses the PANet model to detect and segment multiple instances of objects of interest. During the first iteration of SSL training, we retain only the outputs of the pre-trained model for the *person*, *handbag*, *backpack*, and *suitcase* classes. The *person* class corresponds to passengers and detections of *handbag*, *backpack*, and *suitcase* items are treated as baggage items. In subsequent iterations of SSL training, we modify the model to generate only the object categories $\mathcal{C} \in \{pax, bag\}$, where *pax* corresponds to passengers and *bag* to baggage items. Let $D^{\mathcal{C}}(t)$ be the set of detections on image $I(t)$ at time t. That is, $D^{\mathcal{C}}(t) = \{d_1, \ldots, d_{n_t^{\mathcal{C}}}\}$, where $d_j \in \mathbb{R}^5$ is the detection of the j-th object and $n_t^{\mathcal{C}}$ is the number of objects of class $\mathcal{C}$ in frame $I(t)$. Each detection d_j consists of the coordinates and dimensions of the target's bounding box, $b_j^{\mathcal{C}} \in \mathbb{R}^4$, as well as its detection confidence score $s_j \in [0, 1]$.

We noticed that the detector performs better when objects are observed at more commonly occurring angles (e.g., upright). Therefore, to reduce the negative effect of the overhead perspective, we generate multiple rotated copies of the input image $\Psi_{\theta_i}(t) = R_{\theta_i}(I(t))$ (line 4 in Alg. 1), where $R_{\theta_i}(\cdot)$ is the rotation operator, which rotates the image by an angle θ_i. The angle of rotation θ_i varies between 0 and

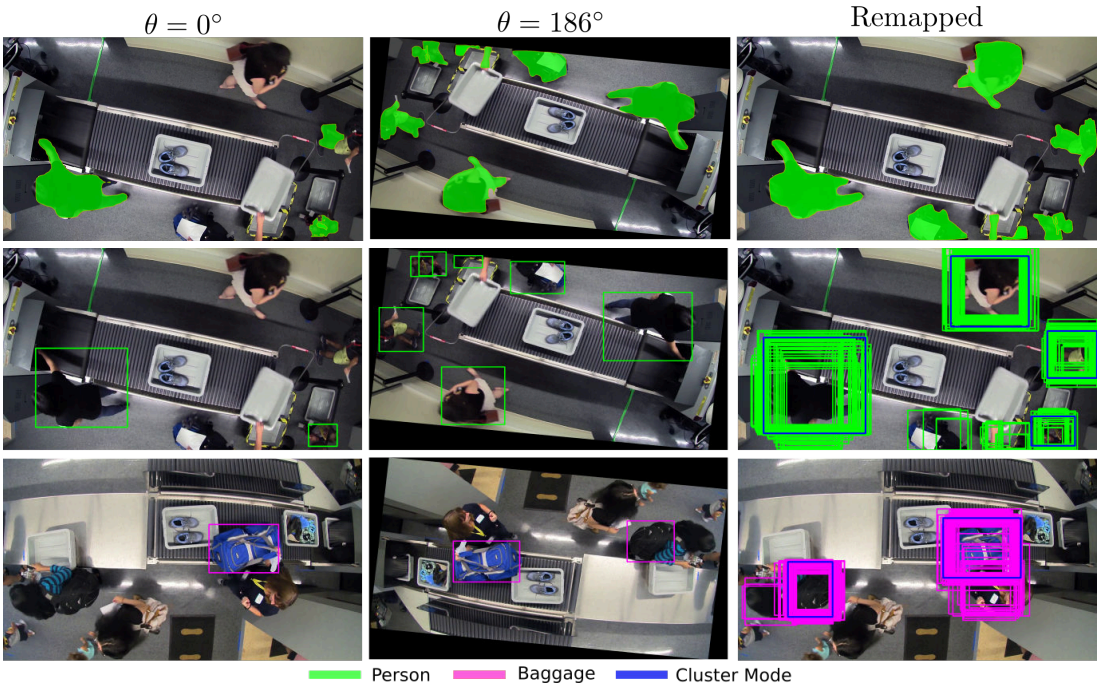


Figure 3.2: Visualization of our data augmentation approach. The first and second columns show the segmentation masks and detections at $\theta = 0°$ and $\theta = 186°$, respectively. The third column shows the remapped detections in the set $S^{\mathcal{C}}$ on the original image (using Alg. 1) with the best detections (blue) from Alg. 2.

2π at intervals of $\Delta\theta = \lfloor \frac{2\pi}{r} \rfloor$, i.e., $\theta_i = \Delta\theta, \ldots, 2\pi$, where r determines the rotation resolution. At each rotation step, we compute the detection set $D^{\mathcal{C}}_{\theta_i}(t)$ for both classes $\mathcal{C} \in \{pax, bag\}$ using a single call to the function $D_{\text{PANet}}(\cdot)$ (line 5). We then remap the resulting detections to the coordinate frame of the original image by applying the inverse rotation to each of the detections in $D^{\mathcal{C}}_{\theta_i}(t)$ (line 6). To avoid localization errors introduced by rotating axis-aligned bounding boxes, we apply the rotation operation to the binary segmentation masks produced by PANet and compute the corresponding bounding boxes using the rotated masks. At the end of Alg. 1, the set $S^{C}(t) = \cup_{i=1}^{r} S^{\mathcal{C}}_{\theta_i}(t)$ contains the detections at all the rotation angles θ_i. Fig. 3.2 illustrates the detections at two rotation angles and the result of mapping detections at 20 different orientations back to the original coordinate system.

Algorithm 2 Cluster Regression

function ClusterRegression($S^C(t)$)
  $\mathcal{D}^{\mathcal{C}}(t) = \emptyset$
  Refine the augmented detections using $S^{\mathcal{C}}(t)$ as region proposals for the D_{PANet} model
  $O^{\mathcal{C}}(t) = \texttt{mean} - \texttt{shift}(S^{\mathcal{C}}(t))$
  for $Q \in O^{\mathcal{C}}(t)$ **do**
    Compute the cluster score $\bar{\eta}_Q$ using Eq. 3.2
    if $\bar{\eta}_Q \geq \lambda$ **then**
      $d = \underset{d_i \in Q}{\text{argmax}}\,(s_i)$
      $\mathcal{D}^{\mathcal{C}}(t) = \mathcal{D}^{\mathcal{C}}(t) \cup \{d\}$
    end if
  end for
  return $\mathcal{D}^{C}(t)$
end function

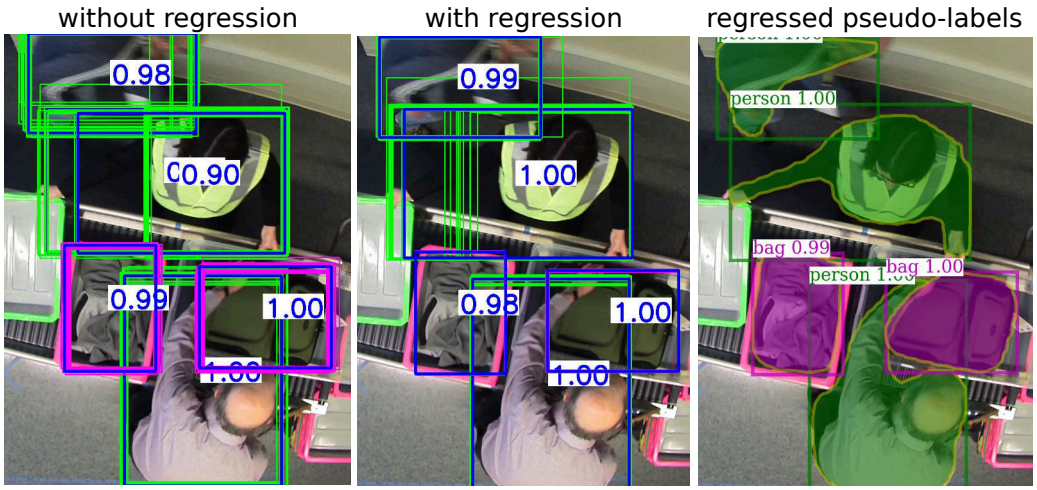


Figure 3.3: Regression on test-time augmented bounding boxes (middle) and cluster modes (right) to generate pseudo-labels for SSL training.

3.3.1.2 Cluster Regression

Alg. 2 summarizes our approach to combine the set of augmented detections $S^C(t)$ into a set of refined target detections $\mathcal{D}^C(t)$. To reduce discrepancies among bounding boxes caused by segmentation errors, we leverage the pre-trained model

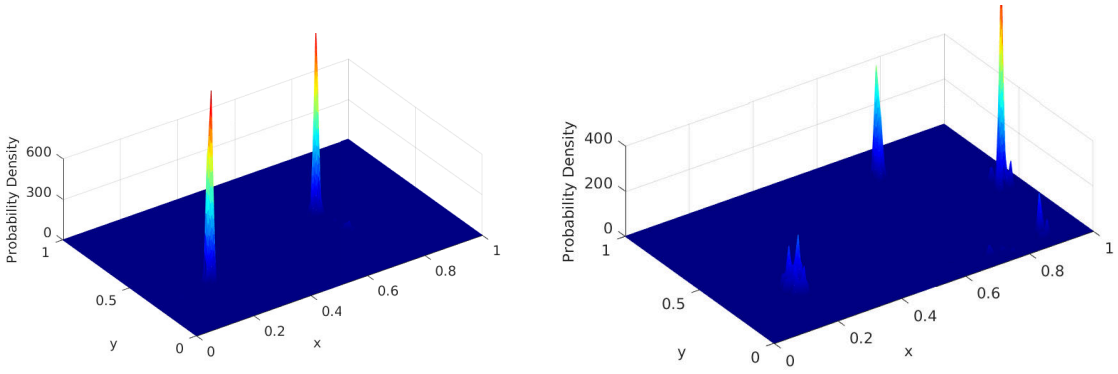


Figure 3.4: Probability of occupancy of passengers (left) and baggage (right) at one frame of our evaluation datasets. The corresponding detections are also shown in Fig. 3.2 (right column).

to regress the set of augmented detections $S^C(t)$. As shown in Fig. 3.1, our cluster regression method uses the backbone features [192] with the augmented detections $S^C(t)$ as region proposals (instead of proposals generated using the region proposal network [1]) to the downstream box and mask heads. To avoid disregarding low-confidence detections that might correspond to relevant region proposals, we do not apply non-maximum suppression to the model predictions. Fig. 3.3 shows that cluster regression significantly increases the accuracy of the bounding boxes generated using the augmented input, and the corresponding segmentation masks are consequently also more accurate.

Cluster Mode Detection. As Fig. 3.4 illustrates, detections and their corresponding confidence scores form a non-parametric distribution of the image's occupancy probability. We use the mean-shift algorithm [100] to identify the modes of that distribution and cluster detections corresponding to common targets. We cluster detections according to their bounding boxes b_j using a multivariate Gaussian kernel [100] with bandwidth $h^{\mathcal{C}}$. We use the sample variances of the object bounding boxes

Algorithm 3 Pseudo-Label Generation

1: **function** PSEUDOLABELS($\mathcal{D}^C(t)$, r)
2: $\mathcal{P}^{\mathcal{C}}(t) = \emptyset$, $\Theta = \{i \cdot \Delta\theta\}_{i=1}^{r}$
3: **for** $d_j \in D^{\mathcal{C}}(t)$ **do**
4: **for** $\theta_i \in \Theta$ **do**
5: Generate the augmented region proposals $d_{i,j} = R_{\theta_i}(d_j)$
6: **end for**
7: Generate the pseudo-label $(\hat{b}_i, \hat{m}_i, \alpha_i)$ using the region proposals $d_{i,j}$
8: $\mathcal{P}^{C}(t) = \mathcal{P}^{C}(t) \cup \left\{(\hat{b}_i, \hat{m}_i, \alpha_i)\right\}$
9: **end for**
10: **return** $\mathcal{P}^{C}(t)$
11: **end function**

at each frame to determine the kernel bandwidth, i.e.,

$$h^{\mathcal{C}} = \text{diag}\left(\sum_{j=1}^{n_t^{\mathcal{C}}} (b_j^{\mathcal{C}} - \bar{b}_j^{\mathcal{C}})(b_j^{\mathcal{C}} - \bar{b}_j^{\mathcal{C}})^T\right), \tag{3.1}$$

where $\bar{b}_j^{\mathcal{C}}$ is the sample mean of $b_j^{\mathcal{C}}$ and $\text{diag}\,(\cdot)$ is the diagonal of the covariance matrix. The correlations among the elements of b_j are negligible and can be safely ignored. Each call to the mean-shift algorithm (line 4 in Alg. 2) produces a set of clusters $O^{\mathcal{C}}(t)$ whose elements are sets of detections assigned to the same target. We consider the detections of passengers and baggage items separately. Hence, two separate invocations of the mean-shift procedure are required to produce the sets $O^{pax}(t)$ and $O^{bag}(t)$. The confidence score $\bar{\eta}_Q$ of cluster $Q \in O^{\mathcal{C}}(t)$ is defined as the ratio between the total score of detections within that cluster and the number of rotation angles considered in the augmentation process, i.e.,

$$\bar{\eta}_Q = \frac{1}{r} \sum_{d_j \in Q} s_j. \tag{3.2}$$

Lines 6-10 of Alg. 2 show that we discard clusters with scores lower than a threshold λ to remove false positive detections.

3.3.1.3 Self-Supervised Model Update

Alg. 3 shows the procedure to generate the pseudo-labels used to update the model. Since our goal is to train the model using labels generated from multiple perspectives, we rotate both the original image and the corresponding predicted modes to generate pseudo-label proposals at each orientation. That is, for each mode $d_j \in D^{\mathcal{C}}(t)$, we generate the pseudo-label mask $\hat{m}_j$ by using the rotated cluster modes $d_{i,j} = R_{\theta_i}(d_j)$, $i = 1, \ldots, r$ as region proposals for the segmentation head, using the same approach described in Section 3.3.1.2. We then find the bounding box $\hat{b}_j$ corresponding to $\hat{m}_j$. The confidence $\hat{\alpha}_j$ of the resulting pseudo-label is given by its corresponding cluster score. The set of pseudo-labels $\mathcal{P}^{\mathcal{C}}(t) = \left\{(\hat{b}_j, \hat{m}_j, \hat{\alpha}_j) \mid d_j \in D^{\mathcal{C}}(t)\right\}$ thus contains accurate annotations even for targets that the model is unable to detect at certain orientations.

Rotation-Invariant Loss. To update the model using rotation-invariant pseudo-labels in a robust and efficient manner, we propose a novel uncertainty-aware, multi-task loss function given by

$$\mathcal{L} = \sum_{\hat{c} \in \mathcal{C}} \sum_{(\hat{b}_j, \hat{m}_j, \hat{\alpha}_j) \in \mathcal{P}^{\mathcal{C}}(t)} \hat{\alpha}_j \left(\mathcal{L}^c(\hat{c}, \tilde{c}) + \mathcal{L}^b(\hat{b}_j, \tilde{b}_j) + \mathcal{L}^m(\hat{m}_j, \tilde{m}_j) \right) + \mathcal{L}_{rpn}, \tag{3.3}$$

where $\tilde{c}$, $\tilde{b}_j$, and $\tilde{m}_j$ are the object class, bounding box, and segmentation mask predicted by the network; $\mathcal{L}^c$, $\mathcal{L}^b$, and $\mathcal{L}^m$ are the classification and bounding box regression losses defined in [1] and the pixel-wise binary cross entropy mask loss described in [6]; and $\mathcal{L}_{rpn}$ is the region proposal network loss from [1]. In Eq. 3.3, the instance head losses are weighted by their corresponding cluster scores. This strategy ensures that instances with low cluster scores that might correspond to incorrect pseudo-labels have little impact on the update of the network parameters. As Alg.

4 indicates, a new set of pseudo-labels is generated at each SSL iteration using the updated model from the previous iteration.

Algorithm 4 Self-Supervised Detection Model Update

Input: Image sequence $I(t)$, $t = 1, \ldots, T$
Output: Updated detection model D_{PANet}

repeat
  for $t = 1, \ldots, T$ **do**
    $S^{\mathcal{C}}(t) = \textsc{AugmentedProposals}(I(t))$
    $\mathcal{D}^{\mathcal{C}}(t) = \textsc{ClusterRegression}(S^{\mathcal{C}}(t))$
    $\mathcal{P}^{\mathcal{C}}(t) = \textsc{PseudoLabels}(\mathcal{D}^{\mathcal{C}}(t))$
  end for
  Fine-tune the D_{PANet} model using the pseudo-labels $\left\{\mathcal{P}^{\mathcal{C}}(t)\right\}_{t=1}^{T}$ according to the loss function in Eq. 3.3
until Convergence criterion is met

3.4 Results and Discussion

In this section, we first briefly discuss the datasets that we used to evaluate our algorithms. We then present an assessment of the proposed SSL approach in terms of passenger and baggage detection. Our evaluation is based on the Multi-Object Detection (MOD) and Tracking (MOT) metrics [69, 193].

3.4.1 Datasets

The video datasets used in this work were recorded at the Kostas Research Institute (KRI) video analytics laboratory at Northeastern University, introduced in Section 2.6. As shown in Fig. 3.5, the laboratory is configured to emulate a realistic airport checkpoint. It is equipped with 14 standard IP surveillance cameras (Bosch NDN-832-V03P) with 1920×1080 resolution and focal lengths between 3 mm and 9 mm. The cameras are installed approximately three meters from the floor with partially overlapping fields of view.

Figure 3.5: Document checking station and divestiture area at the Kostas Research Institute simulated airport checkpoint.

Several actors traverse the checkpoint with baggage items while performing a variety of activities commonly observed in real airports.[1] These activities range from simple scenarios in which just a few passengers pass through the checkpoint in sequential order to crowded scenes in which multiple passengers divest and retrieve their items in a more erratic manner. We collected two separate video datasets: CLASP1, which includes relatively simple scenarios, and CLASP2, which is more complex. Fig. 3.6 shows sample frames of videos from the two datasets. As discussed

[1]The datasets are available upon request at `alert-coe@northeastern.edu`. Northeastern University's Institutional Review Board (IRB) and the Compliance Assurance Program Office (CAPO) within the DHS Science and Technology Directorate have reviewed the referenced human subjects research protocol and related research documentation. No compliance issues or concerns related to the use of human subjects in this protocol have been identified through the review, and the DHS policy requirements for human subjects research protocol review has been met.

Figure 3.6: Sample images from the datasets collected at the simulated airport checkpoint (left: CLASP2 and right: CLASP1 in Table 3.1). The images show the divestiture area (right: camera 9) and item retrieval area (left: camera 11).

Table 3.1: Datasets used to evaluate our algorithms. For each video sequence, the table shows the number of passengers, baggage items, video frames, annotated frames, and the total number of annotated bounding boxes.

Dataset	Video seq.	Passengers	Baggage	Video frames	Annotated frames/rate [fps]	Bounding boxes
CLASP1	A	12	10	6,030	288 (1)	995
	B	12	10	6,180	564 (2)	1,720
	C	8	9	6,030	491 (2)	853
	D	12	8	6,030	523 (2)	1,197
	E	9	9	4,719	1,648 (10)	4,254
CLASP2	F	20	20	12,910	179 (0.01)	737
	G	38	31	10,390	1,346 (3)	4,826
	H	35	29	11,200	198 (0.01)	900
Total	–	146	126	63,489	5,237	15,482

in Section 2.6.2, among the 14 cameras in the laboratory, most passenger interactions take place on cameras 9 and 11. Camera 9 monitors the divestiture area and camera 11 observes the baggage retrieval area. Passengers place their belongings into bins or directly on the conveyor belt in the divestiture area. Then, after passing through the metal detector, they collect their belongings in the baggage retrieval area.

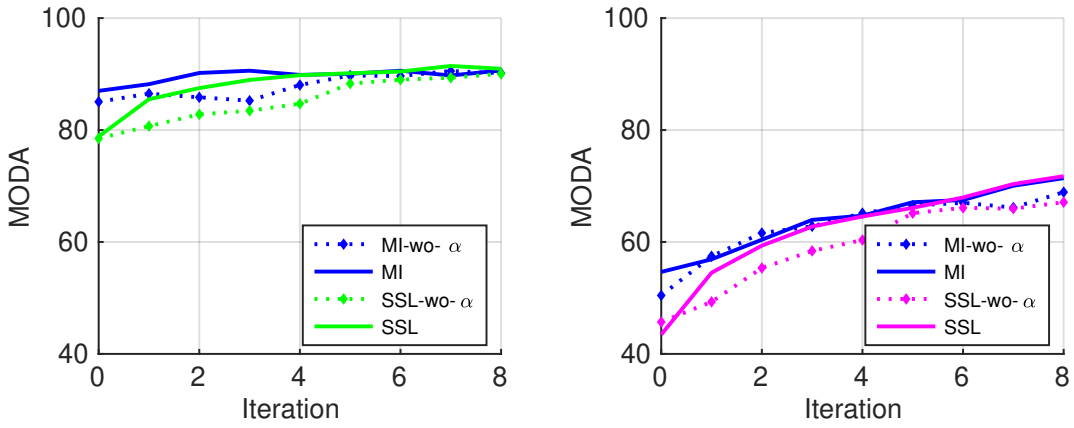


Figure 3.7: MODA measures for person (left) and baggage (right) classes during SSL training.

Table 3.1 shows the specifications of the CLASP datasets used to evaluate our SSL algorithm. We manually annotate the videos with uniquely identified axis-aligned bounding boxes. Given the large number of video frames available in the datasets, the annotation rate for the video sequences varies between 0.01 and 10 frames per second (fps). We randomly partition each dataset into a training set containing 80% of the video frames and a test set with the remaining 20%. For a fair comparison, the Supervised Learning (SL) and SSL models are trained using only the frames from the training set, but the SSL models are fully self-supervised and do not use any manual annotations.

3.4.2 Self-Supervised Learning Detection Performance

During training, we freeze the network weights up to the region proposal network layer so that the pre-trained backbone features are effectively used in the downstream task. We use an initial learning rate of 5e−3, mini-batch size per image $N = 256$, $r = 20$ different orientations, and a cluster confidence threshold $\lambda = 0.1$. Similar to the baseline model, we use stochastic gradient descent with a momentum of 0.9, weight decay of 1e−4. At each SSL iteration, we fine-tune the model for 20k

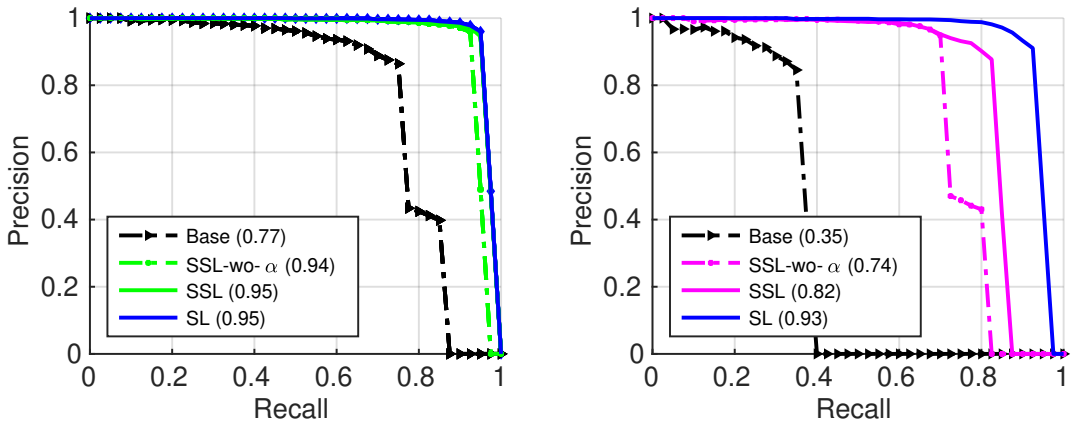


Figure 3.8: Precision-recall curves for person (left) and baggage (right) detection. The legend shows the average precision of the models.

iterations, reducing the learning rate by a factor of 10 at every 5k iterations. In our evaluation, we use an IoU threshold of 0.5, and a non-maximum suppression threshold $\eta_{\text{nms}} = 0.3$ for all the models. The detection threshold for region proposal generation is $\eta_{\text{det}} = 0.5$.

Fig. 3.7 shows the Multi-Object Detection Accuracy (MODA) of our model as a function of the number of SSL iterations. To illustrate the impact of the cluster confidence score, we also evaluate a model in which the samples are not weighed by their scores (SSL-wo-α). Instead, this model uses a hard threshold $\lambda \leq 0.4$ to discard noisy detections during training. The figure also shows the performance of the Multiple-Inference (MI) strategy used to generate the pseudo-labels, which reflects the quality of the pseudo-labels before SSL training. That is, in the MI model, the pseudo-labels themselves are used as model predictions. As the figure indicates, the SSL models gradually approach the performance of the MI strategy. The incorporation of cluster confidences not only increases the speed of convergence of the models but also leads to noticeable performance gains, particularly for baggage items.

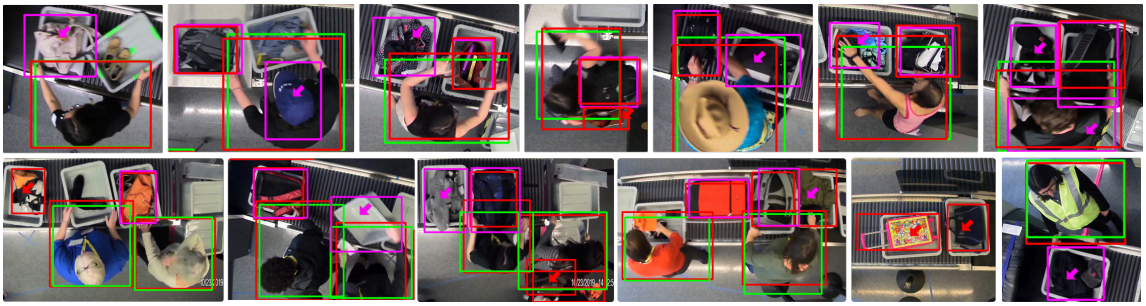


Figure 3.9: Sample results showing failure cases for baggage detection using the SSL model in CLASP1 (top row) and CLASP2 (bottom row) datasets. The magenta arrows indicate bag-like object detections that are not annotated (false positives), the red arrows indicate annotated baggage items the model fails to detect (false negative), the green bounding boxes show the passenger detections, and the red bounding boxes indicate the manual annotations for both classes.

Table 3.2: Passenger and baggage detection evaluation.

Model	Method		↑Rcll		↑Prcn		↑TP		↓FP		↓FN		↑MODA	
	α	reg.	person	bag	person	bag	person	bag	person	bag	person	bag	person	bag
Baseline	✗	✗	73.8	37.1	87.0	82.9	1560	426	228	85	552	724	62.8	29.3
SSL	✗	✗	93.8	73.8	92.3	82.5	1989	858	155	194	123	291	86.0	57.6
SSL	✓	✗	93.5	75.9	93.5	86.1	1985	863	134	144	127	286	87.1	62.9
SSL	✗	✓	93.6	73.1	96.2	92.5	1985	844	73	**71**	127	305	90.1	67.1
SSL	✓	✓	**95.7**	78.6	96.0	91.8	**2025**	903	79	83	**87**	246	91.8	71.5
SL	✗	✗	95.6	**91.4**	**96.4**	**92.8**	2022	**1048**	**70**	83	90	**101**	**92.1**	**84.4**

Fig. 3.8 shows the precision-recall curves for passenger and baggage detection using four detector models: pre-trained PANet (baseline), PANet trained using SL, SSL-wo-α, and SSL. Even though the SSL models are trained without manual annotations, they perform on par with the SL model for passengers. For baggage items, the maximum average precision for the baseline model is less than half of the performance of the SSL models. As illustrated in Fig. 3.9, the performance difference between the SL and SSL models is due to two main issues: i) appearance similarities among bags and certain garments/items placed inside security bins, and ii) baggage items that can only be partially observed before being placed on the conveyor belt.

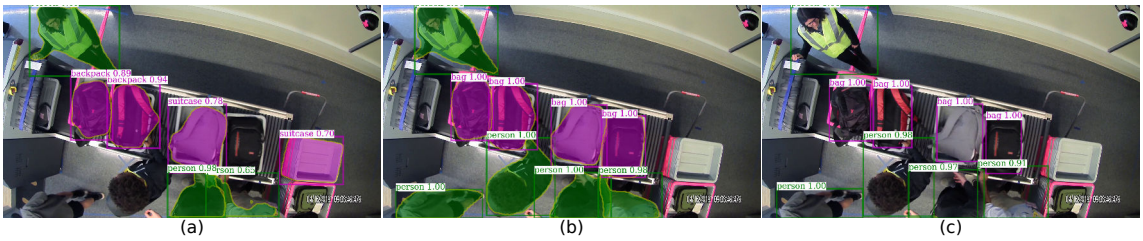


Figure 3.10: Qualitative detection results on the CLASP2 dataset using (a) Baseline, (b) SSL, and (c) SL models (the SL model only predicts bounding boxes).

Table 3.2 demonstrates the benefits of incorporating cluster uncertainties in the SSL loss function (column α) and of the proposed cluster regression technique (column *reg.*). The method that incorporates both cluster uncertainty and regression is equivalent to the approach identified as SSL in Figs. 3.7 and 3.8 whereas the method that does not include cluster confidences corresponds to SSL-wo-α. The results in the table correspond to the point that maximizes the F_1 score of the curves in Fig. 3.8 at the best performing SSL iteration. The top-performing method in Table 3.2 and in the remainder of this section is highlighted in boldface, the second-best is underlined, and ties are broken according to the MODA/MOTA results.

In comparison with the baseline model, our SSL algorithm substantially increases the recall (Rcll) and precision (Prcn) for passenger detection, which is a result of improvements in true positive (TP), false positive (FP), and false negative (FN) detections. The cluster confidence scores substantially reduce the contribution of low-confidence pseudo-labels, especially for baggage items, leading to a noticeable increase in the number of true positives. Cluster regression corrects pseudo-label errors caused by inaccurate bounding boxes generated from poor segmentation results. As a result, the reduction in false positives for both classes is even more pronounced when cluster regression is incorporated. Overall, our SSL framework shows a relative MODA score improvement of 46% for passengers and 144% for baggage items with respect to the baseline model.

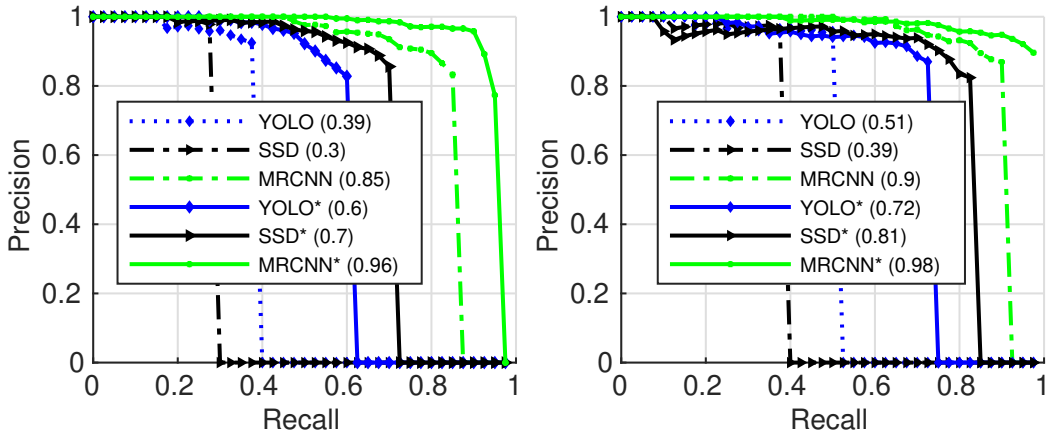


Figure 3.11: Precision-recall curves for passenger detection for camera 9 (left) and 11 (right) on dataset A. The dotted lines show the results for YOLO (blue), SSD (black), and MRCNN (green). The solid lines show the improved results obtained by incorporating the corresponding detectors into Alg. 1 and 2.

Fig. 3.10 presents qualitative results for all the models under consideration. Since the SL model is trained using manually generated bounding boxes, it cannot predict segmentation masks. Our SSL models not only improve the accuracy of the predicted bounding boxes but also generate improved segmentation masks since they are trained using automatically generated instance segmentation pseudo-labels.

3.5 Test-time Data Augmentation for Inference

This chapter's contributions are mainly related to the data augmentation-based pseudo-labels generation and iterative training of a pre-trained model using rotation invariant loss. However, test-time data augmentation is already leveraged in single-stage [54, 194], and multi-stage [6] detection model inference. In this section, the benefits of our proposed data augmentation technique (Alg. 1 and 2) during inference are compared against the performance of state-of-the-art object detectors in overhead camera scenarios.

Passenger Detection. Fig. 3.11 shows the precision-recall curves for passenger detection using three baseline detectors (dotted lines) and for the corresponding

detectors extended using our proposed multiple inference approach (solid lines). All the results are based on an IoU threshold of 0.4, which allows for the correct detection of passengers despite the substantial variability in bounding box size as passengers change their orientations or as they move their arms to interact with baggage items. In our evaluation, we use the same detection and non-maximum suppression thresholds of $\eta_{\text{det}} = 0.5$ and $\eta_{\text{nms}} = 0.1$ for the three networks. The number of rotation angles used for data augmentation is $n = 20$. In the methods augmented with our proposed approach, because of the low recall values of the detectors based on YOLO and SSD, we set the cluster confidence score threshold to $\lambda = 0.1$, whereas for the method based on MRCNN, we use $\lambda = 0.5$. We have observed that further performance improvements are possible by adjusting the value of λ according to the dataset under consideration, but we refrain from using dataset-specific values to demonstrate the generalization capability of our method. For a fair comparison, none of the methods under evaluation is fine-tuned using data from our simulated checkpoint. As previously mentioned, although fine-tuning the baseline models would lead to performance improvements across the board, it might also limit the applicability of our method in real-world scenarios. As the figure shows, our algorithm increases the area under the curve for all three detectors, leading to a maximum of 98% for MRCNN.

As shown in Table 3.3, our detection approach (marked with a $*$) substantially improves the performance of the three baseline detectors. In most of the scenarios under consideration, our algorithm substantially increases the true positive (TP) detections while reducing false positive (FP) and false negative (FN) detections, which results in noticeable improvements in recall (Rcll), precision (Prcn), and MODA results as well. Again, the results in the table correspond to the point that maximizes the F_1 score of the individual algorithms in Fig. 3.11. Hence, in some scenarios, we observe a substantial increase in recall values at the cost of some degradation in the corresponding precision values. As Fig. 3.11 indicates, it is possible to select an oper-

ating point where both metrics are higher than those obtained by the corresponding baseline methods, but that would cause an overall degradation of performance when considering the combined metrics. Overall, our algorithm shows a relative improvement of the MODA scores of 17%, 103%, and 23% with respect to YOLO, SSD, and MRCNN respectively.

Table 3.3: Passenger detection evaluation on dataset A. The $*$ indicates methods augmented with our proposed algorithm.

Cam.	Method	↑Rcll	↑Prcn	↑TP	↓FP	↓FN	↑MODA
9	YOLO	39.5	91.5	130	12	199	35.9%
	YOLO*	**43.8**	**96.6**	**144**	**5**	**185**	**42.2%**
	SSD	28.6	97.9	94	2	235	28.0%
	SSD*	**71.1**	83.6	**234**	46	**95**	**57.1%**
	MRCNN	85.7	79.7	282	42	47	63.8%
	MRCNN*	**86.6**	**96.6**	**285**	**10**	**44**	**83.6%**
11	YOLO	52.3	95.8	115	5	105	50.0%
	YOLO*	**62.7**	92.6	**138**	11	**82**	**57.7%**
	SSD	39.1	96.6	86	3	134	37.7%
	SSD*	**82.7**	82.4	**182**	39	**38**	**65.0%**
	MRCNN	88.6	86.7	195	30	25	75.0%
	MRCNN*	**90.5**	**94.8**	**199**	**11**	**21**	**85.5%**

The high recall, precision, and MODA values indicate that our approach detects most of the passengers correctly in these video sequences. Although the geometric transformations slightly increase the number of false positives caused by the detection of body parts of passengers near the edges of the scene, by retaining clusters with normalized probability score above λ, we are able to ignore most of them (see Fig. 3.2). As shown in chapter 6, Tracktor [23] or MHT [195] effectively handles the few remaining false positives.

Baggage Detection. Fig. 3.12 shows the precision-recall curves for baggage items for an IoU threshold of 0.4. In these results, we use $\eta_{\text{det}} = 0.5$ for MRCNN

and $\eta_{\text{det}} = 0.25$ for YOLO and SSD because these networks show substantially lower confidence levels in the detection of baggage items. Again, for all three networks, $\eta_{\text{nms}} = 0.1$, and the number of rotation angles is $n = 20$. As the figure indicates, the maximum average precision for MRCNN alone is 56%, which is almost 40% lower than what is obtained using our approach. The average precision of baggage detection using YOLO or SSD is less than 0.1. Although our approach substantially improves the average precision for both methods, their performance is still not satisfactory for any practical application. Therefore, we use MRCNN in the evaluation of our tracking and association algorithms discussed in the following subsections.

Table 3.4: Baggage detection evaluation on dataset A. The $*$ indicates methods augmented with our proposed algorithm.

Cam.	Method	↑Rcll	↑Prcn	↑TP	↓FP	↓FN	↑MODA
9	YOLO	3.4	63.6	7	4	199	1.5%
	YOLO*	**7.3**	**78.9**	**15**	**4**	**191**	**5.3%**
	SSD	3.9	88.9	8	1	198	3.4%
	SSD*	**12.1**	**96.2**	**25**	**1**	**181**	**11.7%**
	MRCNN	50.5	91.2	104	10	102	45.6%
	MRCNN*	**72.3**	90.3	**149**	16	**57**	**64.6%**
11	YOLO	9.8	76.9	20	6	185	6.8%
	YOLO*	**18.5**	**90.5**	**38**	**4**	**167**	**16.6%**
	SSD	2.9	100.0	6	0	199	2.9%
	SSD*	**19.0**	83.0	**39**	8	**166**	**15.1%**
	MRCNN	47.3	89.8	97	11	108	42.0%
	MRCNN*	**70.2**	**90.6**	**144**	15	**61**	**62.9%**

The additional metrics listed in Table 3.4 further highlight that, for baggage items, the baseline methods alone fail to detect most of the targets. Our proposed approach more than doubles the MODA of YOLO and SSD for both cameras. For MRCNN, we observe a relative improvement of the MODA score of 61.8% for camera 9 and 54.5% for camera 11. The increase in FP for MRCNN is mostly due to the

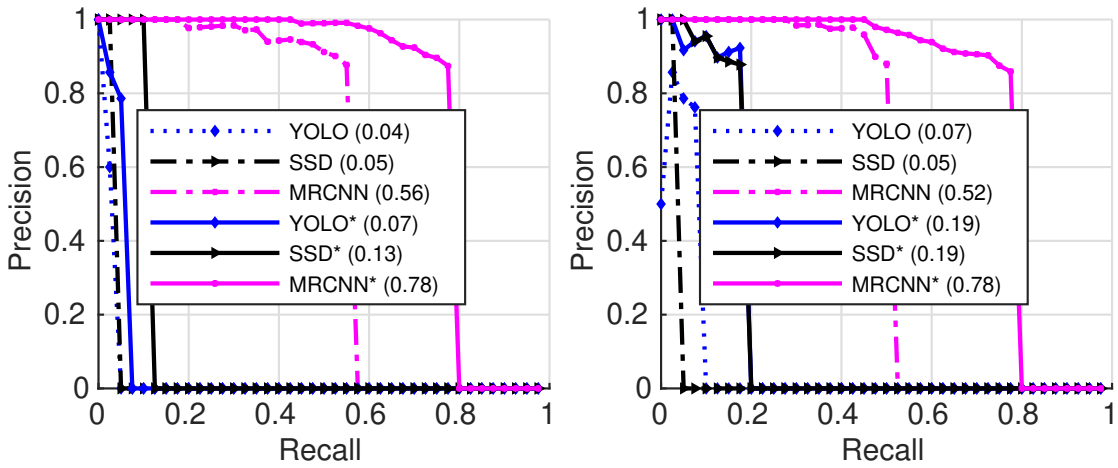


Figure 3.12: Precision-recall curves for baggage detection for camera 9 (left) and 11 (right) on dataset A. The dotted lines show the results for YOLO (blue), SSD (black), and MRCNN (purple). The solid lines show the improved results obtained by incorporating the corresponding detectors into Alg. 1 and 2.

detection of relevant partially observed items such as small purses, clothes, and bins as baggage items that have not been annotated in the ground truth dataset 3.9.

3.6 Self-supervision to Semi-supervision

This section presents a breakdown on the performance of our SSL detector for individual cameras in the CLASP1 and CLASP2 datasets. It also evaluates the performance when the labeled data are used in a semi-supervised approach.

Self-supervised Learning. Fig. 3.13 shows a detailed breakdown of the performance of our SSL detection model for individual camera views in the CLASP1 and CLASP2 datasets. The high recall, precision, and MODA values indicate that our SSL approach detects most passengers correctly in these video sequences. The average precision (AP) for passenger detection is slightly higher for camera 11 in both datasets. The main factor contributing to this performance difference is that in camera 9, passengers are only partially visible most of the time, whereas camera 11 has a better view of the region where the passengers stand next to the conveyor belt. On the other hand, this also contributes to the lower baggage detection performance

in camera 11. That is, in camera 11, partially observed baggage items being carried by passengers (see Fig. 3.9) are much more common than in camera 9. As with passenger detection, we observed similar baggage detection improvements in the camera-specific performance comparisons. This performance could be further improved by using additional unlabelled video frames available in the CLASP1 and CLASP2 datasets to train the SSL models.

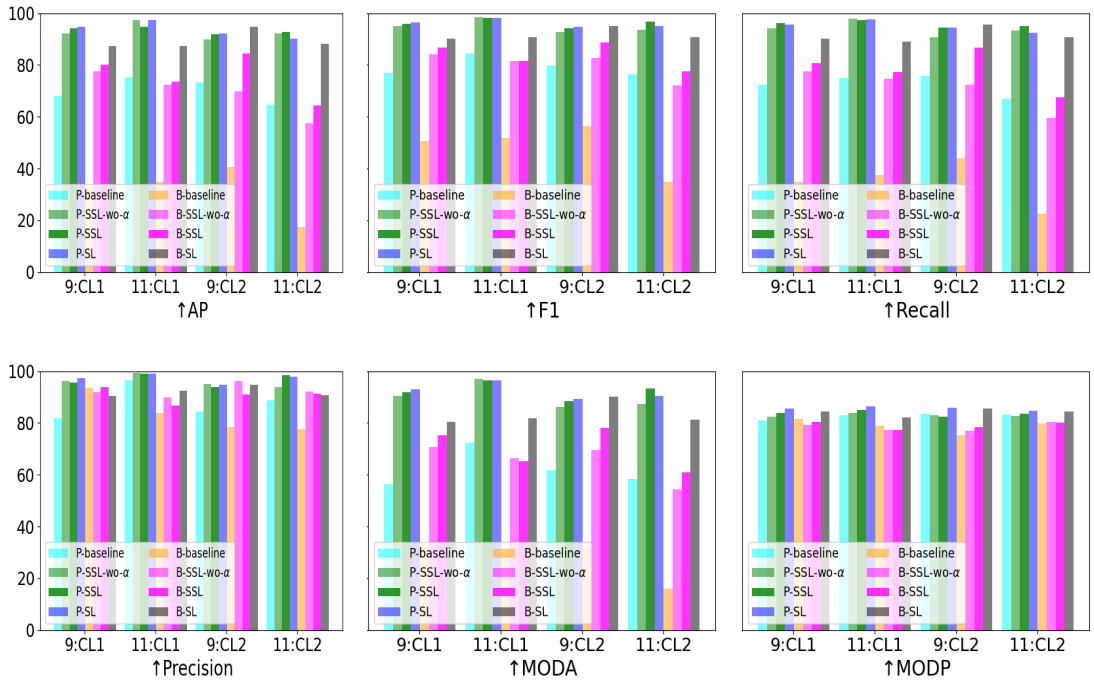


Figure 3.13: Passenger and baggage detection performance in cameras 9 and 11 for the CLASP1 (CL1) and CLASP2 (CL2) datasets. Here, P stands for passenger and B for baggage.

Semi-supervised Learning. As Table 3.2 indicates, the performance of our SSL algorithm is limited by the initial accuracy of the baseline model. Thus, we extend our method to a semi-supervised approach where we use a certain amount of manual annotations to initialize our model before initiating SSL training. For the labeled frames, we employ the same data augmentation procedure used to generate

augmented labels. Fig. 3.14 shows that training the SSL model using 10% of the manual labels leads to a performance comparable to the SL model, outperforming SoftTeacher [14], a state-of-the-art Semi-SL technique. Our method is particularly effective when small amounts of annotations are used. For example, using only 1% of the manual labels, our Semi-SL approach outperforms SoftTeacher by 104% and is only 1.6% behind the SL method (Table 3.5) for baggage items. Furthermore, we observe a 5.7% MODA improvement over the SL method when we use all the manual annotations during training.

Table 3.5: Passenger and baggage detection evaluation measures on the CLASP1 and CLASP2 test sets.

Dataset	Model	Method		↑Rcll		↑Prcn		↑TP		↓FP		↓FN		↑MODA	
		α	reg.	person	bag	person	bag	person	bag	person	bag	person	bag	person	bag
CLASP1	Baseline	✗	✗	73.8	36.2	89.0	87.9	886	233	110	**32**	314	411	64.7	31.2
	SSL	✗	✗	96.4	80.4	95.8	78.6	1157	518	51	141	43	126	92.2	58.5
	SSL	✓	✗	**96.9**	70.4	94.2	90.9	**1163**	451	71	45	**37**	193	91.0	63.0
	SSL	✗	✓	96.0	76.1	97.7	90.7	1152	490	27	50	48	154	93.8	68.3
	SSL	✓	✓	96.8	78.6	97.3	90.2	1162	506	32	55	38	138	94.2	70.0
	SL	✗	✗	96.7	**89.4**	**98.1**	**91.4**	1160	**576**	**23**	54	40	**68**	**94.8**	**81.1**
CLASP2	Baseline	✗	✗	73.9	38.0	85.1	78.0	674	192	118	53	238	313	61.0	27.5
	SSL	✗	✗	91.2	67.3	88.9	86.5	832	340	104	53	80	165	79.8	56.8
	SSL	✓	✗	90.1	81.6	92.9	81.3	822	412	63	95	90	93	83.2	62.8
	SSL	✗	✓	91.3	70.1	94.8	**94.4**	833	354	**46**	**21**	79	151	86.5	65.9
	SSL	✓	✓	**94.6**	78.6	**94.8**	93.4	**863**	397	47	28	**49**	108	**89.5**	73.1
	SL	✗	✗	94.5	**93.5**	94.8	94.2	862	**472**	47	29	50	**33**	89.4	**87.7**

3.7 Parameter Sensitivity and Computation Complexity Analysis

This section evaluate the performance impact of additional data augmentation strategies, number of rotation angles used for data augmentation, and the computation complexity of the SSL detector during both training and inference.

Additional Data Augmentation Strategies. We investigate the impact of other data augmentation strategies during SSL training, including color jittering and

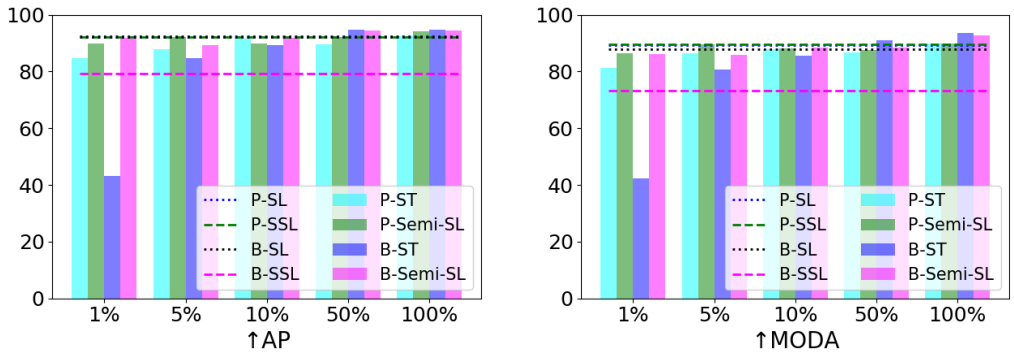


Figure 3.14: Semi-SL model performance on CLASP2 using a semi-supervised extension of our proposed SSL method versus SoftTeacher (ST) [14]. Here, P and B stand for the passenger and baggage categories. The SSL model uses no labeled data and the SL model is trained with 100% of the samples.

motion blur along with multiple rotations. For color jittering, we increase/decrease image brightness, contrast, saturation, and hue by a factor sampled uniformly from the range $[0, \max_{\text{jit}}]$, where $\max_{\text{jit}}$ is 0.4 for brightness, 0.5 for contrast, 0.2 for saturation, and 0.05 for hue. To emulate motion blur, we use Gaussian blur with kernel size uniformly sampled from the set $\{5, \ldots, 9\}$ and standard deviation sampled from the interval $[0.1, 5]$. We observe that applying color jittering and motion blur on the pseudo-label augmentation further improves MODA scores by up to 2.9% and 4.8% for passengers and baggage items, respectively. For a fair comparison, we reduced the number of rotation angles used for augmentation such that the total number of augmented images remains the same in both scenarios. Table 3.7 shows the original number of rotations ($r = 20$) would further increase performance gains.

Impact of Rotation Resolution. Table 3.7 shows the impact of rotation resolution r on the generation of pseudo-labels. One SSL iteration with $r = 20$ improves the MODA scores by up to 3.1% for passengers and 5.6% for baggage items. The inference time for a single frame increases linearly with the number of rotations, contributing to longer SSL training iterations. If training time is a concern, $r = 10$

Table 3.6: Performance impact of additional data augmentation strategies in the SSL iterations.

Dataset	Method			↑AP		↑ F_1		↑MODA	
	Rot.	C-Jit.	Mot.-Blur	person	bag	person	bag	person	bag
CLASP1	✓	✗	✗	89.2	43.4	92.0	59.7	83.9	41.6
	✓	✓	✓	**91.5**	**48.3**	**92.3**	**64.5**	**84.3**	**46.4**
CLASP2	✓	✗	✗	79.4	47.4	86.2	62.5	73.6	42.2
	✓	✓	✓	**84.2**	**47.8**	**88.0**	**63.0**	**76.5**	**43.6**

offers a reasonable speed vs. performance trade-off. We use $r = 20$ for all the SSL models to demonstrate the potential performance of our framework. As Table 3.7 indicates, further increasing the value of r would likely lead to minor additional performance gains.

Table 3.7: Performance impact of the number of rotation angles used in the SSL iterations.

Dataset	r	↓Infer. Time	↑ F_1		↑MODA	
		(secs)	person	bag	person	bag
CLASP1	1	**0.3**	94.5	69.5	89.0	51.7
	5	2.2	95.0	70.4	90.1	52.8
	10	4.5	95.3	70.8	90.6	53.6
	20	9.1	**95.8**	**70.8**	**91.5**	**53.7**
CLASP2	1	**0.3**	91.0	74.5	82.3	56.8
	5	2.6	92.1	76.4	84.6	59.6
	10	4.0	92.1	76.5	84.5	59.8
	20	11.7	**92.2**	**76.5**	**84.9**	**60.0**

Computational Complexity in Self-Supervised Learning. In this section, we analyze the theoretical computational complexity of our SSL strategy and measure the computation time and memory utilization of each step of our algorithm. All our experiments were performed on a workstation equipped with two RTX-2090Ti

GPUs and an Intel® Xeon® Silver 4112 CPU @2.6GHz. The computational complexity of our approach increases linearly with the number of rotation angles used for augmentation in the pseudo-label generation step. That is, for a baseline detection algorithm with computational complexity $\Theta(f(I(t))$, the complexity of our approach is $\Theta(r \cdot f(I(t))$, where r is the number of rotation angles. For example, for $r = 20$, the run-time is 20 times that of a single iteration without augmentation. However, these operations are parallelizable as long as the hardware resources support the simultaneous processing of multiple frames. With our unoptimized implementation, the total time to complete one SSL iteration is approximately six hours for both model training and pseudo-label generation. However, we have observed that hardware resources are severely underutilized, which indicates substantial room for reduction in overall computation time. Table 3.8 shows the computation time of the proposed SSL algorithm,

Table 3.8: Computation time of the proposed tracking-by-detection framework.

Data	Model	Infer. Time (ms)	Memory (MB)
CLASP1	Detector	333.3	1,850
	SCT	142.8	1,748
CLASP2	Detector	333.3	1,850
	SCT	166.6	1,750

employing a PANet detector with a ResNet-50 backbone. The SCT uses the detector results and a ResNet-50-based Re-Identification (Re-ID) model trained on MOT17 to re-label tracklets lost due to short-term occlusions. Hence, the computation time and memory utilization for the SCT are similar to those for the detector model. Since we are processing single images individually instead of image batches, the inference time for the detector and the SCT are far from optimal. Preliminary experiments indicate that processing batches of 10 images simultaneously leads to an approximate six-fold

reduction in detector inference time without exceeding the memory capacity of the GPUs. Reusing the backbone features from the detector in the Re-ID model should also lead to a dramatic reduction in SCT time, since feature generation is the most computationally demanding element of the tracking algorithm.

3.8 Conclusion

We propose a multistage SSL framework to overcome performance limitations of object detection in overhead camera videos for which limited training data is available. Our SSL mechanism fine-tunes object detection models to specific camera views without the need for manual annotations. Our experiments show that the proposed framework can accurately detect passengers and baggage items in multi-camera views of airport checkpoint scenarios. Our framework is flexible and scalable. It requires no training data, incurs no detection computational overhead at inference time, and is independent of the number of cameras in the network.

Our framework also allows seamless integration of additional data augmentation strategies and of manually annotated data when it is available. Our experiments show that these strategies further improve the selectivity of our detector, particularly for baggage items.

CHAPTER 4

SELF-SUPERVISED LEARNING FOR PANOPTIC SEGMENTATION

In this chapter, we extend the SSL method discussed in Chapter 3 for semantic and panoptic segmentation tasks. Convolutional neural networks trained using manually generated labels are commonly used for semantic or instance segmentation. In precision agriculture, automated flower detection methods use supervised models and post-processing techniques that may not perform consistently as the appearance of the flowers and the data acquisition conditions vary. We propose a self-supervised learning strategy to enhance the sensitivity of segmentation models to different flower species using automatically generated pseudo-labels. We employ a data augmentation and refinement approach to improve the accuracy of the model predictions. The augmented semantic predictions are then converted to panoptic pseudo-labels to iteratively train a multi-task model. The self-supervised model predictions can be refined with existing post-processing approaches to further improve their accuracy. An evaluation on a multi-species fruit tree flower dataset demonstrates that our method outperforms state-of-the-art models without computationally expensive post-processing steps, providing a new baseline for flower detection applications.

4.1 Introduction

Computer vision algorithms are becoming increasingly popular in agricultural applications. Detecting and counting flowers is an important crop management activity to optimize fruit production [196]. Automatic bloom intensity estimation methods have the potential to reduce workloads in large production fields. Many machine vision approaches have been proposed to address the challenges of estimating crop yield. Most recent flower detection and counting methods based on deep learning models require a large amount of manually labeled training data to achieve acceptable perfor-

mance [25, 197, 198]. Although weakly supervised approaches [199] can simplify the training of convolutional neural networks (CNNs), they are not particularly effective to adapt large-scale, pre-trained models to unseen object categories.

Data augmentation [88, 87] is a de facto standard technique to reduce the dependence on manual annotations when training deep neural networks. But in agricultural visual data, the appearance of objects of interest and the scene conditions vary significantly from one field to another. Besides, since agricultural production environments usually require images to be acquired from moving vehicles [25, 198, 200], the sun conditions and dense background clutter make this task challenging in terms of model generalization. Hence, we still need to generate enough manual labels for various species of crops to generalize the prediction models across species with significantly different appearance and backgrounds potentially comprised of semantically distinct elements.

Although deep CNNs can perform reasonably accurate pixel-level semantic predictions [103, 25], false alarms due to similarities between flowers, fruits at different stages of maturation, and background objects limit potential opportunities for the application of computer vision algorithms to agricultural automation tasks. Instance [7] and panoptic [9] segmentation models might be able to better identify individual flowers or clusters of flowers and thus improve detection performance.

To address the above challenges, inspired by the works presented in [9, 18, 25], we propose a novel self-supervised panoptic segmentation approach that leverages a small number of annotations for supervised learning (SL) and then adjusts the model to challenging unlabeled datasets.

Contributions. In summary, the main contributions of this chapter are:

- A robust self-supervised flower segmentation method that addresses typical agricultural visual data challenges in fruit tree orchards.

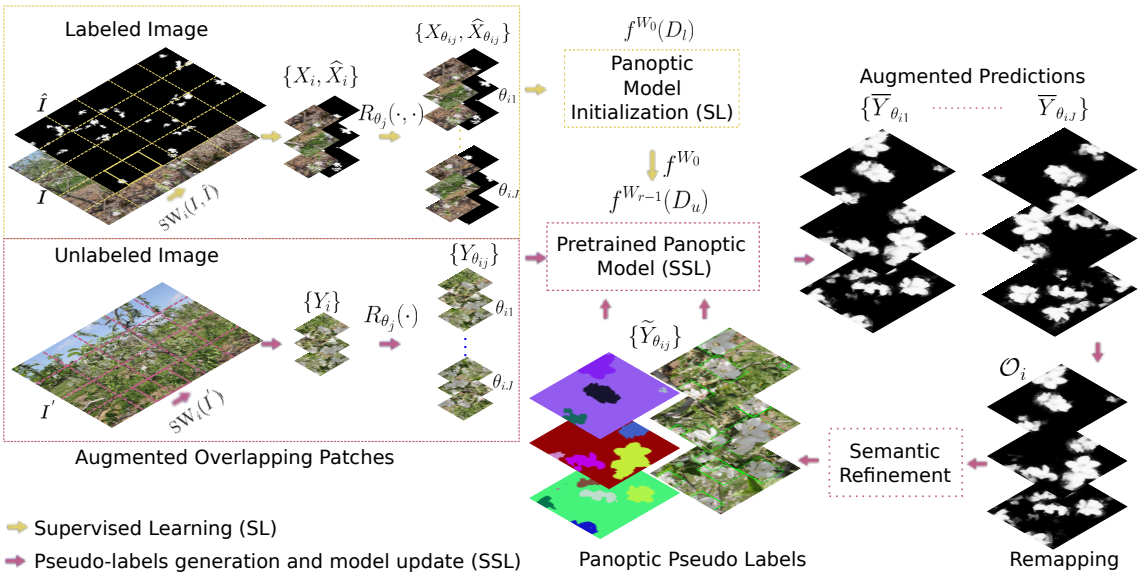


Figure 4.1: Proposed self-supervised learning framework for multi-species flower segmentation. Labeled images are used to initialize the model for flower segmentation. The overlapping sliding window patches of the unlabeled input images are rotated multiple times to generate augmented semantic predictions from a previously initialized panoptic segmentation model. The remapping step transforms the score maps to the input coordinate system and then the normalized predictions are used to generate the panoptic pseudo-labels using a semantic refinement procedure to update the pre-trained model.

- A novel panoptic pseudo-label generation technique for automatically updating the model for unlabeled datasets that contain severe clutter and illumination challenges.
- A robust sliding-window-based training and testing approach that does not require additional post processing to refine the network predictions.
- Extensive evaluations on multiple-species datasets, which demonstrate superior generalized performance over state-of-the-art techniques.
- Our source code and pre-trained models are available at `https://github.com/siddiquemu/ssl_flower_semantic`.

4.2 Related Work

In agricultural automation, several supervised [201, 202, 6] and weakly supervised [203] deep learning models have been employed to address the challenges of detecting flowers [25, 197, 204, 205], fruits [206, 207, 198], or entire plants [208]. Applications of these methods range from robotic harvesting to estimating fruit load and optimizing fruit production by counting flowers in the early blooming season. Although some of these approaches leverage data augmentation techniques to generate automatic labels [209, 210, 18], none of these methods addresses model generalization ability for significantly different test datasets. In the context of object detection and segmentation, recent methods attempt to accommodate data distribution shifts through the following techniques: a) supervised learning, b) semi-supervised learning, c) self-supervised learning, and d) multi-task panoptic segmentation models.

Supervised Methods. These methods usually employ basic image transformations [7, 201] or sophisticated data augmentation techniques [30, 211] to improve model generalization. In addition to data augmentation during training, some methods incorporate post-processing algorithms at test time [212, 213] or include specialized input/output units that are easier to fine-tune to new datasets [214, 215]. While these techniques reduce the dependency on annotations for different datasets, they do not eliminate it. Model performance is still largely dependent on the amount of training data available.

Semi-supervised Methods. Using labeled data to bootstrap a model whose predictions are then employed to fine-tune the initial model (or to train a student model) is a popular approach to develop methods for multiple object detection [14], as well as instance [7, 18] and semantic [103] segmentation. This strategy is effective when labeled and unlabeled data have similar appearance and sufficient labeled data is

available to bootstrap a deep model. When the characteristics of the labeled and unlabeled data differ significantly, as is the case among different flower species, more sophisticated supervision mechanisms are needed [216, 217].

Self-supervised Methods. When no labeled data is available, self-supervision strategies can be used to automatically generate pseudo-labels from the unlabeled data [39, 75]. In these scenarios, the initial model is trained to solve a surrogate task that presumably has a similar representation structure as the target task [218]. Unsupervised learning techniques are widely used to align latent feature representations [216]. Self-supervision strategies that use model prediction uncertainties to guide the learning process, while arguably more interpretable and predictable, are less commonly explored. Our approach uses a multi-inference data augmentation mechanism in conjunction with the region growing refinement (RGR) algorithm [212] to generate robust and accurate pseudo-labels in an iterative manner. These pseudo-labels allow our model to continuously improve its performance on previously unseen datasets.

Panoptic Methods. Multi-task learning is commonly used to improve model performance across different tasks [5]. As long as the tasks are similar, the model tends to generalize better to unseen data [120]. The recently introduced panoptic segmentation approach jointly learns the closely related tasks of instance and semantic segmentation and currently represents the state of the art in instance and semantic segmentation [121, 219]. However, training such models requires a significant number of manual labels containing instance and semantic information. Our approach makes it possible to apply a panoptic model to significantly different datasets without resorting to manual labels. To our knowledge no self-supervised panoptic segmentation method has been proposed so far.

4.3 Self-supervised Panoptic Segmentation

Our proposed self-supervised learning (SSL) technique for panoptic segmentation shown in Fig. 4.1 comprises three main components: i) labeled and unlabeled data augmentation, ii) panoptic model initialization using the labeled dataset, and iii) panoptic pseudo-label generation from unlabeled data to update the model. As shown in Alg. 5, we use images from the training set and their corresponding labels to train our initial model using an SL strategy. Our SSL approach then updates the initial model iteratively in a fully self-supervised manner using the pseudo-labels generated by the model at a previous iteration.

4.3.1 Data Augmentation

Our method is based on the panoptic segmentation model proposed in [9] pre-trained on the COCO [24] and COCO-stuff [220] datasets. To fine-tune the

Algorithm 5 Self-supervised Learning Algorithm

Input: Set of high resolution labeled images I, their corresponding segmentation labels $\hat{I}$, and the set of unlabeled images $I^{'}$

Output: Self-supervised model f^{W_r} for unlabeled data $I^{'}$

1: Generate the augmented training set D_l using I and $\hat{I}$ according to Eq. 4.1
2: Train the initial model $f^{W_0}(D_l)$ using D_l
3: Generate the augmented unlabeled image patches $Y_{\theta_{ij}}$
4: **for** $r \leftarrow 1$ to maxIter **do**
5: Generate the augmented predictions $\overline{Y}_{\theta_{ij}}$ using Eq. 4.3
6: Compute the normalized score map $\mathcal{O}_i$ using Eq. 4.4
7: Compute the binary semantic mask S_i from $\mathcal{O}_i$ using RGR
8: Generate the augmented binary semantic masks $\hat{S}_{\theta_{ij}}$
9: Apply connected component analysis to $\hat{S}_{\theta_{ij}}$ to find the instance masks $\hat{m}^{(l)}_{\theta_{ij}}$ and bounding boxes $\hat{b}^{(l)}_{\theta_{ij}}$
10: Construct the set of pseudo-labels $\widehat{Y}_{\theta_{ij}}$ using Eq. 4.5
11: Construct the set $D_u = \{Y_{\theta_{ij}}, \widehat{Y}_{\theta_{ij}}\}$
12: Update the self-supervised model $f^{W_{r-1}}(D_u)$ using D_u
13: **end for**

model for flower segmentation, we augment the training set introduced in [25] using a sliding window (SW) technique. That is, we extract from the input image I and its corresponding semantic label $\hat{I}$, both of size $M \times N$ pixels, overlapping patches of size $m \times n = \lfloor M/K \rfloor \times \lfloor N/K \rfloor$ pixels with a stride of $p \times q = \lceil m/2 \rceil \times \lceil n/2 \rceil$, where K is the window size factor. Let $(X_i, \widehat{X}_i) = \mathrm{SW}_i(I, \hat{I})$ be the i-th image patch and its corresponding semantic label. We augment X_i and $\hat{X}_i$ by applying J different rotations at randomly selected angles $\{\theta_j\}_{j=1}^{J}$. For the sake of sampling efficiency, rather than directly sampling from the interval $[0, 2\pi]$, we employ a stratified sampling strategy. That is, we partition the circle into five sectors centered at $(\pi/2) \cdot k$, $k = 0, 1, \ldots, 4$ and sample each sector uniformly. This strategy increases sample diversity, ultimately reducing the variance of the pseudo-labels generated using our method. Thus, the set of labeled image patches and corresponding manual labels used to train the supervised model is given by

$$D_l = \left\{ \left(X_{\theta_{ij}}, \widehat{X}_{\theta_{ij}} \right) \right\} = \left\{ R_{\theta_j}(\mathrm{SW}_i(I, \hat{I})) \right\}, \tag{4.1}$$

where $R_{\theta_j}(\cdot, \cdot)$ rotates its two arguments by an angle θ_j.

We employ the same data augmentation procedure for each unlabeled image of the test sets to generate the unlabeled augmented samples $Y_{\theta_{ij}}$ from the corresponding image patches Y_i. In the SSL approach, we use the SL model to predict the initial augmented pseudo-labels $\widehat{Y}_{\theta_{ij}}$ used to fine-tune the model for unseen datasets. The procedure for pseudo-label generation is described in detail in Section 4.3.2. Thus the unlabeled dataset for each flower species is

$$D_u = \left\{ \left(Y_{\theta_{ij}}, \widehat{Y}_{\theta_{ij}} \right) \right\}. \tag{4.2}$$

At test time, we simply apply the sliding window operation to generate the normalized semantic score maps and combine the predictions corresponding to the overlapping portions of each window using majority voting. We observed that the

(a) $\bar{Y}_{\theta_{ij}}$ (b) $\mathcal{O}_i$ (c) $b_{\theta_{ij}}$ (d) $m_{\theta_{ij}}, S_{\theta_{ij}}$

Figure 4.2: Illustration of the steps of our panoptic pseudo-label generation method. a) semantic prediction for a single augmented patch, b) normalized average score map obtained using Eq. 4.4, c) instance bounding boxes, and d) instance segmentation masks and semantic labels generated during SSL iterations.

benefit of test-time data augmentation is negligible after a few SSL training iterations. Hence, we do not perform rotation augmentation at inference time, which ensures that the computational time of the model remains unchanged.

4.3.2 Pseudo-label Generation

Data distribution shifts degrade the accuracy of segmentation models. Strong data augmentation is an effective strategy to mitigate this problem [221]. Thus, to improve the sensitivity of our model to different flower species, we apply the data augmentation procedure described above to Y_i and use the previously computed network weights $W_{(r-1)}$ to generate the augmented predictions at the r-th SSL iteration according to

$$\overline{Y}_{\theta_{ij}} = f^{W(r-1)}(Y_{\theta_{ij}}). \tag{4.3}$$

To remap the semantic predictions back to the original image coordinate frame, we apply the reverse rotation operator $R_{-\theta_j}(\cdot)$ with bi-linear interpolation to the augmented predictions $\overline{Y}_{\theta_{ij}}$. We then normalize the scores using a softmax function and use the average normalized score map $\mathcal{O}_i$ as our final prediction, i.e.,

$$\mathcal{O}_i = \frac{1}{J} \sum_j \sigma \left(R_{-\theta_j}(\overline{Y}_{\theta_{ij}}) \right), \tag{4.4}$$

where $\sigma(\cdot)$ represents the softmax function applied element-wise to the individual logits for the classes $\mathcal{C} \in \{\text{background}, \text{flower}\}$. As Figs. 4.2 (a) and (b) illustrate,

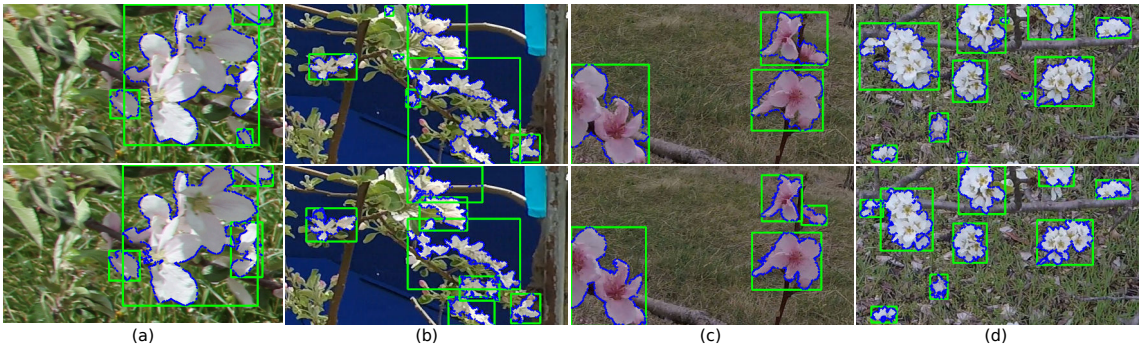


Figure 4.3: Comparisons between the pseudo-labels generated using a fixed threshold τ_{seg} (top row) and the RGR-based semantic refinement (bottom row). a) AppleA, b) AppleB, c) Peach, d) Pear. The segmentation masks in the images at the bottom row better reflect flower boundaries and the corresponding bounding boxes better distinguish nearby flower instances.

$\mathcal{O}_i$ contains a significantly higher number of flowers segmented with high confidence than a single augmented patch $\overline{Y}_{\theta_{ij}}$.

4.3.3 Semantic Prediction Refinement

Instead of applying a fixed threshold τ_{seg} to generate panoptic pseudo-labels from $\mathcal{O}_i$, we employ RGR, a robust segmentation refinement method [212]. RGR uses a Monte Carlo strategy to perform an appearance-based refinement of low-confidence regions in $\mathcal{O}_i$ using the corresponding image patch Y_i, which allows it to generate an improved binary segmentation mask. RGR uses three key elements to determine the boundaries of an object of interest: 1) the confidence of the model predictions, 2) appearance similarities among pixels, and 3) distances among pixels. That is, every pixel in an image is associated with a nearby pixel of similar appearance whose semantic class has been predicted with high confidence. As Fig. 4.3 illustrates, RGR improves the boundary adherence of the pseudo-labels and better distinguishes flower instances.

Let S_i be the semantic binary mask obtained from $\mathcal{O}_i$ using RGR. As in the pseudo-label generation step, we apply J rotations to S_i to generate augmented

semantic binary masks, $\hat{S}_{\theta_{ij}} = R_{\theta_j}(S_i)$. We then perform connected component analysis to obtain the corresponding instance masks $\hat{m}^{(l)}_{\theta_{ij}}$ and bounding boxes $\hat{b}^{(l)}_{\theta_{ij}}$ for the $l = 1, \ldots, L$ distinct elements of $\hat{S}_{\theta_{ij}}$. The augmented panoptic pseudo-labels are given by

$$\widehat{Y}_{\theta_{ij}} = \left\{ (\hat{b}^{(l)}_{\theta_{ij}}, \hat{m}^{(l)}_{\theta_{ij}}), \hat{S}_{\theta_{ij}} \right\}_{l=1}^{L}. \tag{4.5}$$

Figs. 4.2 (c) and (d) show that this approach generates high-quality bounding boxes and instance masks.

4.3.4 Multi-task Loss

In both the SL and SSL models, the instance bounding boxes $\hat{b}^{(l)}_{\theta_{ij}}$ and segmentation masks $\hat{m}^{(l)}_{\theta_{ij}}$ from the augmented labels are used to train the ROI-heads for the flower class. The augmented semantic masks $\hat{S}_{\theta_{ij}}$ are used to train the semantic segmentation head for the background and flower classes. For panoptic segmentation learning, we consider background as a stuff class and flower as a thing class [222] to jointly update the model using the following multi-task loss function

$$\mathcal{L} = \sum_{\hat{c} \in \mathcal{C}} \sum_{(\hat{b}_{\theta_{ij}}, \hat{m}_{\theta_{ij}}, \hat{S}_{\theta_{ij}}) \in \widehat{Y}_{\theta_{ij}}} (1 - \lambda) \left(\mathcal{L}^c(\hat{c}, \tilde{c}) + \mathcal{L}^b(\hat{b}_{\theta_{ij}}, \tilde{b}_{\theta_{ij}}) + \mathcal{L}^m(\hat{m}_{\theta_{ij}}, \tilde{m}_{\theta_{ij}}) \right) + \lambda \mathcal{L}^s(\hat{S}_{\theta_{ij}}, \tilde{S}_{\theta_{ij}}), \tag{4.6}$$

where $\mathcal{L}^c$ is the classification loss, $\mathcal{L}^b$ is the bounding-box loss, $\mathcal{L}^m$ is the mask loss, and $\mathcal{L}^s$ is the segmentation loss, as defined in [9]. By further training the initial SL model on the unlabeled datasets using the proposed SSL approach where the augmented panoptic labels are robust to prediction uncertainty and intrinsically incorporate rotation invariance, it is possible to iteratively improve the performance of the model.

4.4 Experiments

We compare the performance of our method against the state-of-the-art algorithms presented in [25, 197] using the evaluation metrics and procedures described in [25]. To quantify the benefit of employing RGR as part of our pseudo-label generation strategy, we evaluate two different techniques to generate the pseudo-labels. First, we evaluate an approach in which we apply a fixed threshold τ_{seg} to the predicted score maps. For a fair comparison, we determine τ_{seg} based on the maximum F_1 score obtained by the model on the training set at a previous iteration (see Fig. 4.5). We call this model `SSL`. The model in which we employ RGR to refine the score maps without hard thresholding is deemed `SSL+RGR`. We also assess the performance improvements obtained by applying RGR as a post-processing mechanism in conjunction with our SSL model. We refer to that approach as `SSL+RGR (pp)`, where *pp* stands for post-processing. As a baseline, we also assess the performance of the `SL` model trained only on the AppleA dataset applied to the other datasets.

4.4.1 Datasets

We evaluate our method on the multi-species flower dataset first introduced in [25], which comprises four subsets: i) AppleA (train/test), ii) AppleB, iii) Peach, and iv) Pear. The AppleA and AppleB datasets contain images of the same apple orchard, but collected on different dates and under distinct conditions. While AppleA was collected using a hand-held camera, AppleB images were captured by a camera mounted to a mobile platform. For additional details regarding the datasets, we refer the reader to Section 2.6.4 and to [25].

We train our SL model using the AppleA training set, which consists of 100 images with a resolution of $M \times N = 5184 \times 3456$ pixels [25]. After applying J rotation augmentation steps, the number of training patches $X_{\theta_{ij}}$ for each input image

is $J \times (2K-1)^2$ since $i = 1, 2, \ldots, (2K-1) \times (2K-1)$ and $j = 1, 2, \ldots, J$. Hence, for $K = 4$ and $J = 20$, there are $98,000$ training patches in the AppleA dataset. These patches are used to train our initial panoptic flower segmentation model.

We consider a randomly selected subset comprising 70% of the 30 images from the AppleA test set as unlabeled images $I^{'}$ to fine-tune the SL model using the automatically generated panoptic pseudo-labels. Similarly, 70% of the images from the AppleB, Peach, and Pear datasets (18, 24, and 18 images, respectively), all of which have a resolution of 2704×1520 pixels, are considered unlabeled images used to update the SL model iteratively. The remaining images in each dataset are used solely for performance evaluation. Given the relatively small size of the test sets, we evaluate our methods using five-fold cross-validation.

Figure 4.4: Examples of improved annotations in the AppleA training set. The cropped sections show (a) incorrect contours containing background pixels, and (b) improved labels.

The datasets introduced in [25] provide pixel-level, high-resolution annotations of individual flowers. However, as Fig. 4.4 shows, the annotations have imperfections that can only be observed when closely inspected. Despite being small, these inaccuracies comprise a non-negligible portion of the image pixels, especially considering

that only a fraction of the pixels correspond to flowers. To resolve this issue, we use the MATLAB® image labeler tool to manually correct inaccurate labels and to label additional smaller but clearly visible unannotated flowers. Fig. 4.4 shows some examples of the annotations before and after the corrections.

4.4.2 Training Details

The vast majority of image pixels in the datasets correspond to background pixels. Hence, to provide the model sufficient samples containing flower pixels, we train the network for $20,000$ iterations using stochastic gradient descent with a batch size of 512 samples and a base learning rate of 25e$-$4, which is divided by 10 at 10%, 25%, and 50% of the training period. We freeze the ResNet-101 backbone [49] during training. To emphasize semantic learning, we use $\lambda = 0.8$ in Eq. 4.6. We have empirically observed that setting RGR's average spacing between samples to 100 pixels provides an adequate balance between the accuracy of the refined score map and the computation time required to produce it. We use the values reported in [25] for the remaining parameters, namely, the number of iterations is 10, the score map threshold is 0.5, the high-confidence foreground threshold is 0.8, and the high-confidence background threshold is 0.01.

4.5 Results and Discussion

Table 4.1 compares the performance of the SL and SSL models against the algorithms presented in [25, 197]. Although the SL model trained using our proposed data augmentation strategy segments flowers using a fixed threshold τ_{seg}, it performs either on par with or better than the state-of-the-art models on test sets that are similar to the training set, even without applying our proposed SSL strategy. However, for datasets with significantly different characteristics, the SL model does not perform satisfactorily. The SSL approach using a fixed threshold outperforms the

Table 4.1: Evaluation of flower segmentation performance using our SSL panoptic model. The best results are shown in boldface and the second-best are underlined. We report the average value of the evaluation measures and their standard deviations across five runs.

Dataset	Method	IoU	F1	Rcll	Prcn
AppleA	DeepLab+RGR [25]	71.4	83.3	87.7	79.4
	DeepLab+SCL [197]	**81.1**	**89.6**	**91.9**	87.3
	SL	77.1±0.9	87.0±0.5	86.7±0.6	87.3±0.8
	SSL	76.2±0.6	86.1±0.7	88.2±0.9	84.8±0.9
	SSL+RGR	77.9±0.6	87.5±0.3	87.8±0.6	87.3±0.6
	SSL+RGR (pp)	79.6±0.6	88.6±0.3	89.2±0.6	**88.1±0.7**
AppleB	DeepLab+RGR [25]	63.0	77.3	**91.2**	67.1
	DeepLab+SCL [197]	65.3	79.6	72.7	87.4
	SL	75.8±0.8	86.2±0.5	85.4±1.1	87.1±0.5
	SSL	76.8±0.7	86.8±0.4	87.0±0.7	86.7±0.8
	SSL+RGR	78.7±0.4	88.1±0.2	87.9±0.3	88.2±0.7
	SSL+RGR (pp)	**79.9**±0.8	**88.9**±0.5	86.7±1.0	**92.2**±0.3
Peach	DeepLab+RGR [25]	59.0	74.2	64.8	86.8
	DeepLab+SCL [197]	64.3	77.7	70.3	88.2
	SL	48.9±3.5	65.6±3.2	62.6±4.2	68.9±2.6
	SSL	67.8±4.1	80.7±2.9	**85.3**±2.1	76.7±3.6
	SSL+RGR	75.2±3.2	85.8±2.1	84.6±1.9	86.9±2.4
	SSL+RGR (pp)	**78.3**±3.2	**87.8**±1.7	84.9±2.1	**91.1**±3.0
Pear	DeepLab+RGR [25]	75.4	86.0	79.2	94.1
	DeepLab+SCL [197]	74.5	85.4	75.4	**97.3**
	SL	77.3±1.9	87.2±1.3	85.1±2.4	89.4±0.7
	SSL	78.6±1.7	87.9±1.0	87.9±1.6	88.1±0.8
	SSL+RGR	82.4±1.9	90.4±1.2	**89.4**±1.8	91.3±1.4
	SSL+RGR (pp)	**84.2**±2.1	**91.4**±1.2	87.4±1.9	95.8±0.9

baseline methods on the AppleB, Peach, and Pear datasets by significant margins (11.5%, 3.5%, and 4.1% absolute IoU improvement with respect to [197]). For the AppleA dataset, the SSL method alone outperforms [25] but is slightly worse than [197]. This is largely due to the fact that the baseline methods perform dramatically better on the training set, whereas the performance of our model remains relatively stable across datasets. As discussed in more detail below, background flowers also

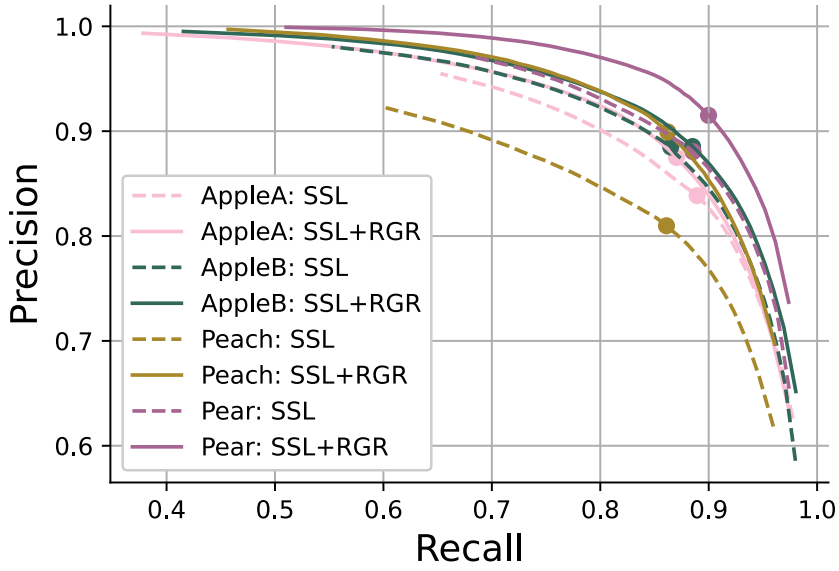


Figure 4.5: Precision-recall curves for the SSL models with and without RGR pseudo-label refinement. Solid circles represent points that maximize F_1 scores.

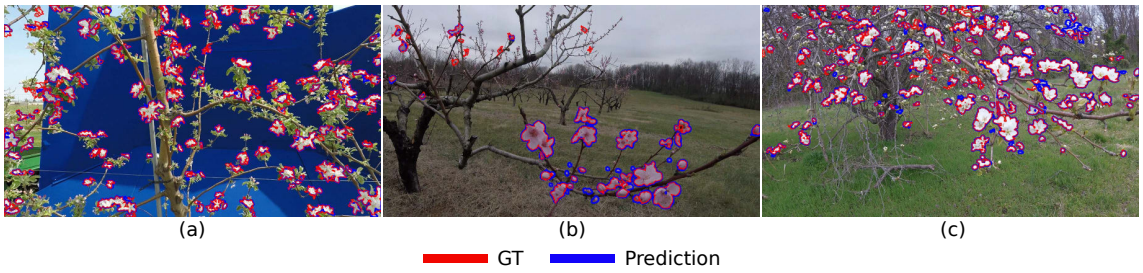


Figure 4.6: Qualitative assessment of our proposed SSL approach on test datasets (a) AppleB, (b) Peach, (c) Pear. Most false positives correspond to small, unlabeled flowers.

contribute to the performance degradation. When we use RGR to refine the pseudo-labels, we observe IoU improvements with respect to the fixed threshold SSL method of 1.9%, 7.4%, and 3.8% for the AppleB, Peach, and Pear datasets, respectively. The performance improvements obtained with RGR are proportional to the appearance dissimilarities between the AppleA dataset used for model pre-training and the corresponding target dataset. The average hue, saturation, and value difference between

the AppleA dataset and the AppleB dataset is 30.3, whereas for the Peach and Pear datasets it is 76.9 and 28.9, respectively. Finally, performing an additional RGR step at test time leads to an additional average IoU improvement of approximately 1.9% but at the cost of substantially higher inference times, as discussed in the next section. Fig. 4.5 shows the precision-recall curves for the proposed SSL methods with and without pseudo-label refinement using RGR.

The qualitative results in Fig. 4.6 show that the SSL models are highly sensitive to flowers in complex regions. For some datasets, the SSL methods show slightly lower precision than [197]. The main reason for the lower precision is the presence of small, unannotated flowers in the datasets that our model can detect. This can be observed in Fig. 4.6 (c) where several small flowers are present, especially on branches farther from the camera. Determining which flowers should be annotated is an application-specific problem that requires further investigation.

4.5.1 Parameter Sensitivity and Computation Time Analysis

Table 4.2 shows the impact of the sliding window size factor K and the number of rotation angles J on model performance and average inference time per input image. This evaluation is performed on the first SSL iteration of a model initialized with $K = 4$ and $J = 20$. That is, the evaluation reflects the impact of model parameters on the accuracy of the resulting pseudo-labels. The top two rows show the test-time impact of varying K without employing test-time rotations (i.e., $J = 1$) for the AppleA and AppleB datasets, respectively. The last row of the table shows that the IoU and F_1 measures on the Peach dataset gradually increase with J when rotation augmentation is employed at inference time, but so does the computation time. Inference times were obtained using one NVIDIA® GeForce® RTX 2080 Ti GPU without any multi-processing technique. Post-processing times using RGR are approximately 16× higher than those presented in Table 4.2 on our Intel® Xeon®

Table 4.2: Performance impact of sliding window size and number of rotation angles.

Dataset	$M \times N$	K	J	IoU	F_1	Inf. Time (sec.)
AppleA	5184×3456	4	1	73.6	84.8	7.2
		8		75.4	86.0	15.4
		16		53.3	69.5	90.0
AppleB	2704×1520	2	1	71.6	83.0	1.4
		4		76.7	86.8	5.3
		8		57.1	72.6	22.1
Peach	2704×1520	4	1	51.3	67.8	5.5
			5	58.2	73.6	34.7
			10	60.3	75.2	71.5
			20	61.3	76.0	147.8

Silver 4112 CPU @2.6GHz. Results for the remaining datasets are similar and are omitted for brevity. Fig. 4.7 shows the impact of λ in the multi-task loss (Eq. 4.6) for different flower species. Although the performance of our approach remains relatively stable as we vary λ, for most datasets, the best results are obtained with $0.7 \leq \lambda \leq 0.9$, especially in cross-species scenarios, where appearance variation is more prominent.

4.6 Conclusion

We introduced a self-supervised learning technique to accurately segment multiple tree flower species without significant manual labeling efforts. To automatically generate instance and semantic labels for unlabeled datasets, we propose a data augmentation technique associated with a semantic segmentation refinement strategy that produces accurate pseudo-labels for self-supervised model training. The proposed SSL technique makes it possible to train a deep multi-task model effectively on unlabeled fruit flower datasets. Self-supervised learning substantially reduces model dependency on computationally expensive post-processing steps to further refine the

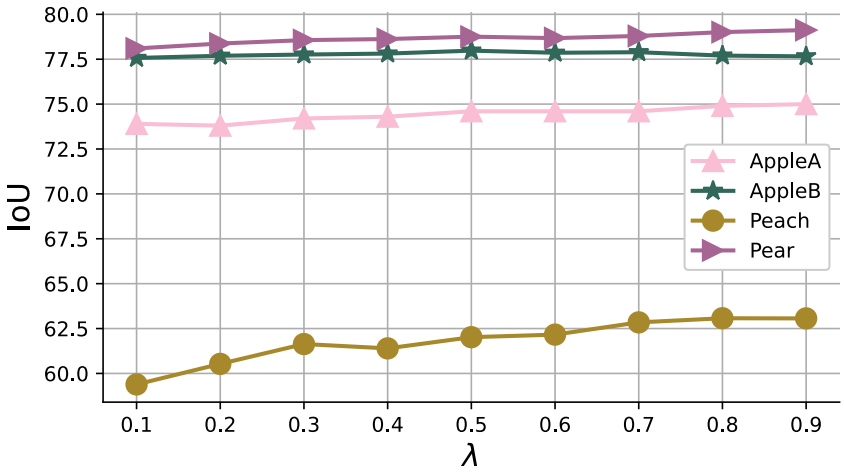


Figure 4.7: Impact of the loss weight λ (Eq. 4.6) on flower segmentation performance at the first SSL iteration with $J = 20$ and $K = 4$.

model predictions at inference time. That being said, employing a post-processing approach with our SSL model can further improve its prediction accuracy. Our novel SSL method creates a new baseline for the multi-species flower segmentation task.

A robust and accurate multi-species flower detection method is the first step toward the development of autonomous robotic thinning systems [223]. In the future, the proposed panoptic flower segmentation algorithm can be further improved in a number of ways. First, our proposed framework resorts primarily to a data augmentation strategy based on image rotations. Given the characteristics of the problem under consideration, it stands to reason that additional data augmentation strategies such as color jittering and image blurring would further contribute to the generation of accurate pseudo-labels. In addition, instead of using empirically defined weights for the instance and semantic segmentation tasks, task-dependent uncertainty learning strategies [17] may better capture appearance variations to optimize the task weights. Finally, pseudo-label pixels or sometimes entire instances may have low prediction scores. The uncertainty of the pseudo-labels may be used to weigh the contributions of individual samples. As shown in Chapter 3, uncertainty-weighed

loss functions [18] are a promising technique to accomplish that goal.

CHAPTER 5
UNSUPERVISED SPATIO-TEMPORAL EMBEDDING LEARNING FOR MULTIPLE OBJECT TRACKING AND SEGMENTATION

This chapter addresses the challenges of joint segmentation and tracking problems as a single-camera video understanding task that can utilize the previously discussed image understanding tasks, such as detection and instance segmentation. Assigning consistent temporal identifiers to multiple moving objects in a video sequence is a challenging problem. A solution to that problem would have immediate ramifications in multiple object tracking and segmentation problems. In this chapter, we propose a strategy that treats the temporal identification task as a spatio-temporal clustering problem. We propose an unsupervised learning approach using a convolutional and fully connected autoencoder, which we call deep heterogeneous autoencoder, to learn discriminative features from segmentation masks and detection bounding boxes. We extract masks and their corresponding bounding boxes from a pre-trained instance segmentation network and train the autoencoders jointly using task-dependent uncertainty weights to generate common latent features. We then construct constraints graphs that encourage associations among objects that satisfy a set of known temporal conditions. The feature vectors and the constraints graphs are then provided to the k-means clustering algorithm to separate the corresponding data points in the latent space. We evaluate the performance of our method using challenging synthetic and real-world multiple-object video datasets. Our results show that our technique outperforms several state-of-the-art methods.

5.1 Introduction

The goal of MOTS algorithms is to establish temporally consistent associations among segmentation masks of multiple objects observed at different frames of a video sequence. To accomplish that goal, most state-of-the-art MOTS methods [26, 12] em-

ploy supervised learning approaches to generate discriminative embeddings and then apply feature association algorithms based on sophisticated target behavior models [41, 27, 129]. This chapter proposes a novel perspective on the problem of temporal association of segmentation masks based on spatio-temporal clustering strategies.

Subspace clustering algorithms applied to sequential data can separate sequences of similar data points into disjoint groups. State-of-the-art subspace clustering methods have shown promising performance on single object patches [132], video sequences [148], and face tracking datasets [147]. For video sequences containing multiple objects, subspace clustering can be used as a data association strategy to assign a unique temporal identifier to each object. However, due to variations in the data distribution caused by changes in the appearance of the objects and by misdetections, occlusions, and fast motions, subspace clustering in video segments using only location [224, 225], shape [139, 138, 142, 226], or appearance [147, 140] features might not produce satisfactory results.

The goal of this work is to increase the discriminative capability of spatio-temporal latent representations. Traditional subspace clustering techniques are trained based either on appearance [139, 226, 142] or location information, generally in the form of bounding boxes [224]. Instead, in this work, we propose a novel approach that learns location and shape information jointly using a convolutional and fully connected autoencoder, which we call Deep Heterogeneous Autoencoder (DHAE). To learn a latent representation that leverages motion and appearance information in an unsupervised manner, we employ a multi-task loss function with task-dependent uncertainties [10]. The use of multi-task uncertainty-aware learning automatically assigns low weights to samples that do not contribute to the desired representation. A point with high uncertainty would correspond to a point which the self-expressive layer cannot represent well, i.e., a point whose corresponding subspace is not yet known.

Contributions. The major contributions of this chapter are as follows:

- We propose a novel unsupervised mechanism based on task-dependent uncertainties that learns to generate spatially and temporally distinctive latent features based on heterogeneous inputs.
- We propose a new data partitioning algorithm that uses constrained clustering strategies to associate object detections over multiple frames with their corresponding temporal identifiers.
- We evaluate our model on two synthetic and two real-world datasets that include most of the challenges commonly observed in MOTS problems, such as pose and appearance variations.

5.2 Related Work

Subspace clustering is an unsupervised learning technique in which data points are mapped to lower dimensionality subspaces where it is easier to make inferences about the relationships among different data points. Existing clustering methods employ two common strategies: i) extract low-dimensional discriminative features, ii) apply a robust clustering approach to partition the subspaces. Earlier approaches employed methods based on factorization strategies [131, 132, 133] or kernels [134, 135, 136, 137] to separate the data points into their respective subspaces. More recent methods employ convolutional neural networks [138, 139, 140], or generative adversarial networks [141, 142], oftentimes in conjunction with self-expressive layers [139, 142, 143]. The autoencoder-based DSC-Net [139] uses fully connected layers to learn an affinity matrix that enhances the discriminative property of the embeddings. Some autoencoder-based techniques consider both subspace reconstruction error and cluster assignment error for better sample distribution [144, 138]. Although existing approaches may be able to find discriminative features to cluster static data, these

features are not sufficiently distinctive to identify the subspaces corresponding to multiple objects in sequential data.

Clustering multi-object sequential data is an under-explored problem, particularly in real-world applications. While existing temporal clustering methods, such as ordered subspace clustering (OSC) [146] consider sequential data, they focus on clustering entire video frames, without taking into consideration the spatial aspect of the problem, which must be addressed when it is necessary to distinguish multiple objects in a video segment. Few methods [147, 148] address the problem of clustering objects over video sequences, which must take into account the fact that object features may change over time [124, 125, 149]. Our approach addresses this challenge using a simple yet effective unsupervised learning framework.

5.3 Subspace Clustering for Sequential Data

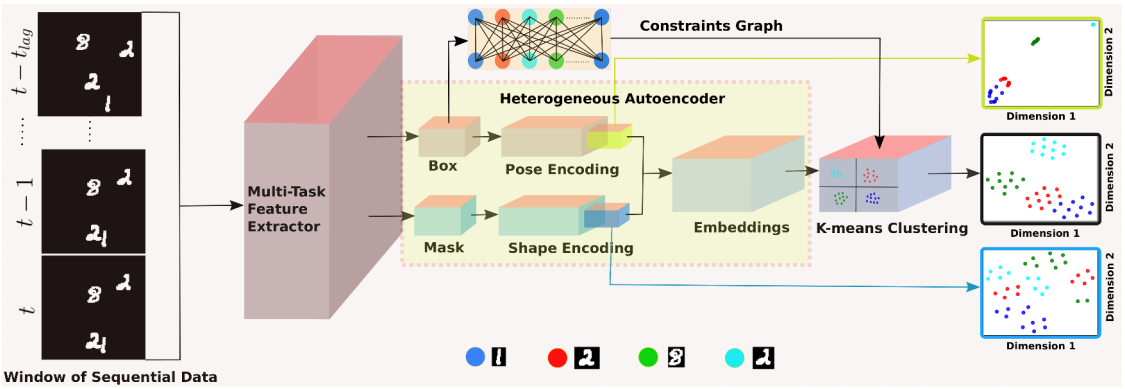


Figure 5.1: Proposed subspace clustering framework. The Multi-task Feature Extractor detects the bounding boxes and segmentation masks of multiple objects within a window of sequential data. The Deep Heterogeneous Autoencoder then uses these features to generate joint embedded representations of the objects. These embeddings are then clustered into target trajectories using constrained k-means.

As Fig. 5.1 illustrates, our spatio-temporal clustering framework extracts features of interest from the objects in each video frame using a multi-task feature

extractor module. Then, our proposed DHAE generates a discriminative latent representation of the pose and appearance of each object based on these features. To establish temporal coherence among targets, we adopt a graph-based method [227] to preclude the association of points that violate a set of constraints that are known to hold and enforce the association of points having common temporal identifiers within a temporal window. Finally, we use the constrained k-means algorithm [145] to determine the labels of the targets by minimizing the dissimilarity of their latent representations while satisfying the association constraints. Alg. 6 summarizes the steps of the proposed method, which are described in detail below.

Algorithm 6 Subspace clustering

Input: Set of video frames $\{I^t\}_{t=1}^{T}$
Output: Subspace clusters $\mathcal{C}_K$

repeat
  $\mathcal{W}^t = \texttt{MTFE}(\{I^{t'} \,|t' \in \mathcal{T}^t\})$
  $\mathcal{Z}^t = \texttt{DHAE}(\mathcal{W}^t)$
  Compute $\mathcal{G}^t$ using Eqs. (5.4)-(5.6)
  $\mathcal{C}_K = \emptyset$
  $\mathcal{C}_t = \texttt{kmeans}(\mathcal{Z}^t, \mathcal{G}^t)$
  for $Q \in \mathcal{C}_t$ **do**
    $\bar{\tau} = {}^{1}/_{|Q|} \sum_{d_i \in Q} (s_i)$
    if $\bar{\tau} > \lambda$ **then**
      $\mathcal{C}_K = \mathcal{C}_K \cup \{Q\}$
    end if
  end for
until end of the video sequence

5.3.1 Multi-Task Feature Extractor

The multi-task feature extractor (MTFE) module is responsible for generating segmentation masks and bounding boxes of objects of interest in each video frame (see Fig. 5.2). This task is independent of the proposed temporal clustering mechanism and can be performed by any supervised or unsupervised segmentation method such as [6, 228]. More specifically, let $d_{b,i}^t \in \mathbb{R}^N$ be the detected bounding box of the i-th

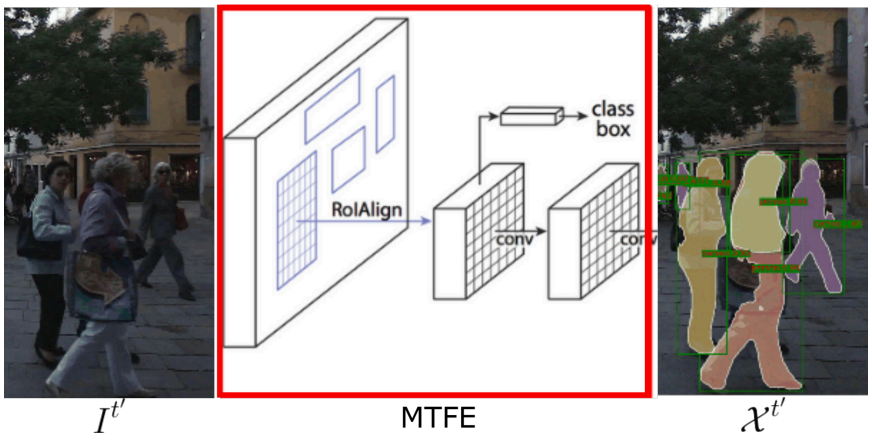


Figure 5.2: Multi-task feature extraction using MTFE [6].

target observed at time t, $d^t_{m,i} \in \mathbb{R}^{M \times M \times D}$ be a segmentation mask representing the appearance of that object, and $s^t_i \in [0, 1]$ be the corresponding detection confidence. The MTFE takes as input a video frame I^t and generates the set

$$\mathcal{X}^t = \left\{ [d^t_{m,i}, d^t_{b,i}, s^t_i] \right\}_{i=1}^{\mathcal{O}^t}, \tag{5.1}$$

where $\mathcal{O}^t$ is the number of unique objects at time t. The bounding box $d^t_{m,i}$ is represented by the coordinates of its centroid and its dimensions, hence $N = 4$. Regarding appearance representation, we propose two closely related models. In the *shape* model, the number of channels of the mask $D = 1$ and $d^t_{m,i}$ corresponds to the binary segmentation mask of the object. In the *appearance* model, $D = 3$ and $d^t_{m,i}$ is given by the binary segmentation mask multiplied by the corresponding RGB contents of the image.

5.3.2 Deep Heterogeneous Autoencoder

Clustering methods that resort only to object appearance information do not perform well when multiple objects are observed simultaneously in a sequence of video frames. As the number of objects of a certain category (e.g., pedestrians) observed in a given frame increases, the average appearance difference among them becomes

increasingly lower. At the same time, as the duration of the temporal segment increases, so does the variability in the appearance of any given target. Hence, to allow for sufficient temporal appearance variability while preventing incorrect associations among temporally proximal observations, we incorporate location information into the latent feature representation.

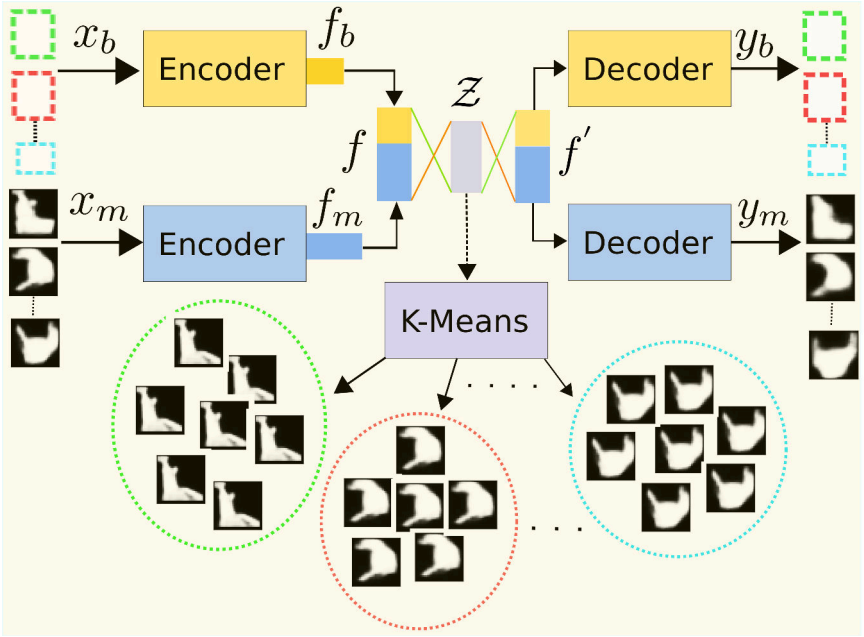


Figure 5.3: Proposed Deep Heterogeneous Autoencoder (DHAE) model architecture, which jointly learns shape and location information for sequential data clustering.

Fig. 5.3 shows our proposed DHAE architecture. To combine shape and location information, we design a network consisting of three parts: i) a pair of encoders that take as input the N-dimensional location vector[1] d_b and the $M \times M \times D$ mask d_m, ii) an uncertainty-aware module based on self-expressive layers [139] to reconstruct the concatenated feature f' and learn the latent feature $\mathcal{Z}$, iii) a pair of decoders to reconstruct the bounding box y_b and the mask y_m. The DHAE takes the extracted set of shapes and locations $\mathcal{X}^t$, which are generated by the MTFE, and reconstructs

[1]To simplify the notation, we henceforth drop the subscript i and the superscript t.

them by minimizing the combined reconstruction loss. To incorporate the location features d_b into our model, we employ a fully connected auto-encoder (AE) with N inputs, which is represented by the yellow boxes in Fig. 5.3. The corresponding encoded feature vector is $f_b = h_b(d_b)$, where $h_b : \mathbb{R}^N \rightarrow \mathbb{R}^F$ is the encoding function. The shape information d_m is encoded by a convolutional auto-encoder (CAE) with an input size of $M \times M \times D$, which is represented by the blue boxes in Fig. 5.3. Let $f_m = h_m(d_m)$ be the latent feature of the CAE, where $h_m : \mathbb{R}^{M \times M \times D} \rightarrow \mathbb{R}^F$ is the encoding function. A function $h_a : \mathbb{R}^{2F} \rightarrow \mathbb{R}^F$ takes the concatenated feature vector $f = [f_m, f_b]$ and converts it into the latent representation $\mathcal{Z} \in \mathbb{R}^F$. A function $h'_a : \mathbb{R}^F \rightarrow \mathbb{R}^{2F}$ then takes the latent representation $\mathcal{Z}$ and produces a new feature $f^{'} = [f^{'}_m, f^{'}_b]$, which combines both shape and location information. We then process the two components of the feature vector separately using the corresponding decoders. That is, the outputs produced by the decoding function $h^{'}_m : \mathbb{R}^F \rightarrow \mathbb{R}^{M \times M \times D}$ and $h^{'}_b : \mathbb{R}^F \rightarrow \mathbb{R}^N$are $y_m \in \mathbb{R}^{M \times M \times D}$ and $y_b \in \mathbb{R}^N$.

5.3.3 Multi-task Likelihood

We train our DHAE using a multi-task loss function using maximum log-likelihoods with task-dependent uncertainties. Let $f^W(x) = [f^W_m(x), f^W_b(x)]$ be the output of the DHAE with weights W, input vector $x = [d_m, d_b]$, and predicted output $y = [y_m, y_b]$. The likelihood for the regression task is given by

$$p(y_m, y_b | f^W(x), \sigma_m, \sigma_b) = p_m(y_m | f^W_m(x), \sigma_m) \cdot p_b(y_b | f^W_b(x), \sigma_b), \tag{5.2}$$

where y_b and y_m are normally distributed with means $f^W_b(x)$, $f^W_m(x)$, and variances σ_b, σ_m, respectively. Thus, the cost function is given by

$$\begin{aligned} \mathcal{L}(W, \sigma_m, \sigma_b) &= -\log p(y_m, y_b | f^W(x), \sigma_b, \sigma_m) \\ &\propto \frac{1}{2\sigma_b^2} \left\| y_b - f^W_b(x) \right\|^2 + \frac{1}{2\sigma_m^2} \left\| y_m - f^W_m(x) \right\|^2 + \log \sigma_m + \log \sigma_b, \end{aligned} \tag{5.3}$$

where W, σ_m, and σ_b are trainable parameters. Unlike traditional multi-task learning approaches [120, 10], which focus on weighing the contributions of several *homogeneous outputs*, our method weighs the contribution of multiple *heterogeneous input* features. Thus, we learn the relative weights of each loss function term based on the uncertainty of distinct features.

5.3.4 Network Implementation and Training Details

The AE branch of our DHAE has an input size of $N = 4$ and one fully connected layer of size 128. The CAE branch uses 5 convolutional layers with kernel size 3×3, ReLU activations, and a stride of 2×2 for downsampling and upsampling [55]. The size of the input layer is $M = 128$ and the subsequent layers have half the size of the previous layer. The number of convolutional channels in each layer is 16, 16, 32, 32, and 64 (the decoder mirrors the structure of the encoder). The functions $h_a(\cdot)$ and $h'_a(\cdot)$ are implemented using fully connected layers of size $F = 128$. We train the network using stochastic gradient descent with ADADELTA [229] learning rate adaptation.To account for the dimensionality of the feature vectors, we initialize $\log \sigma_b^2 = {}^1\!/_N$ and $\log \sigma_m^2 = {}^1\!/_{(M^2)}$. The network weights W are initialized using the Glorot method [230]. At inference time, the segmentation masks are isotropically scaled such that the largest dimension of the mask is M. The image is then centered along the smallest dimension and zero-padded.

5.3.5 Sequential Data Constraints

In spatio-temporal subspace clustering, constraints based on prior knowledge regarding the sequential data may reduce association mistakes by imposing penalties on unlikely pairwise matching. We use an undirected graph $\mathcal{G}^t$ to encode constraints among pairs of detections in the temporal window $\mathcal{T}^t$. This graph determines which pairs of detections cannot belong to the same cluster. It also enforces the association

of detections that were assigned the same temporal identifier in previous temporal windows.

Constraints Graph Formulation. To incorporate prior knowledge regarding object correspondences in a temporal window, we construct the graph $\mathcal{G}^t = (V^t, E^t)$ using *cannot link* and *must link* constraints. The vertices of the graph correspond to the set of segmentation masks in all the frames in the window $\mathcal{T}^t$, i.e.,

$$V^t = \left\{ d^t_{m,i} \,\middle|\, t \in \mathcal{T}^t, i \in \left\{1, \ldots, \mathcal{O}^t\right\} \right\}. \tag{5.4}$$

The set of edges consists of pairs of nodes v_i and v_j that meet the spatial and temporal restrictions imposed by the *cannot link function* $f_{cl}(\cdot)$ and *must link function* $f_{ml}(\cdot)$, i.e.,

$$\begin{aligned} E^t_{cl} &= \left\{ (v_i, v_j) | v_i \in V^t, v_j \in V^t, f_{cl} = 1 \right\}, \\ E^t_{ml} &= \left\{ (v_i, v_j) | v_i \in V^t, v_j \in V^t, f_{ml} = 1 \right\}. \end{aligned} \tag{5.5}$$

The function f_{cl} prevents the association of vertices from the same frame or vertices in close temporal proximity whose segmentation masks do not overlap. Conversely, f_{ml} enforces the association of detections with common temporal identifiers. That is, if a detection has been assigned an identifier at a previous temporal window, it is only allowed to be associated with detections that have the same identifier or that have not been assigned one, i.e.,

$$\begin{aligned} f_{cl}(v_i, v_j) &= \begin{cases} 1 & \text{if } t_i = t_j \text{ or } \gamma_i \neq \gamma_j \\ 1 & \text{if } \texttt{iou}(v_i, v_j) = 0,\ t_i - t_j \leqslant \tau \\ 0 & \text{otherwise} \end{cases}, \\ f_{ml}(v_i, v_j) &= \begin{cases} 1 & \text{if } l_i = l_j,\ \gamma_i = \gamma_j, \\ & t_i \neq t_j,\ l \neq \emptyset \\ 0 & \text{otherwise} \end{cases}, \end{aligned} \tag{5.6}$$

where t_i, t_j are the timestamps for the detections corresponding to v_i and v_j, the function $\texttt{iou}(\cdot)$ computes their mask intersection over union, γ_i, γ_j are the corresponding object classes, and l_i, l_j are the initialized cluster identities corresponding to nodes v_i and v_j.

5.3.6 Modified Constrained K-means

In the constrained k-means algorithm [145], a node v_i may be assigned to the same cluster as v_j only if the edge $(v_i, v_j) \notin E^t_{cl}$ and the initialized detections maintain the same subspace clustering when $(v_i, v_j) \in E^t_{ml}$. We cluster objects that do not satisfy the *cannot link* constraints and whose tracklet identities are not yet initialized by minimizing the distance between their corresponding latent features z^t_i and the cluster centroids z_k.

The dimensionality of the subspace corresponding to window $\mathcal{T}^t$ is given by the total number of objects observed during that period, which is unknown. We propose a novel mechanism to determine the number of clusters $|\mathcal{K}|$ that leverages our temporal clustering constraints. Our approach consists of initially setting the number of clusters to the maximum number of targets observed in a single frame within the window $\mathcal{T}^t$, i.e., $|\mathcal{K}| = \max_{t' \in \mathcal{T}^t} \mathcal{O}^{t'}$. The centroids of the clusters are then initialized using the detections in $\mathcal{O}^{t'}$. The objective of our algorithm is to associate each embedded feature $z^t_o \in \mathcal{Z}^t$ to a unique cluster with centroid z_k while minimizing the cost

$$L(\mathcal{K}) = \sum_{z_k \in \mathcal{K}} \sum_{z^t_o \in \mathcal{Z}^t} \left\| z_k - z^t_o \right\|^2 . \tag{5.7}$$

Thus, our approach groups the nodes into $|\mathcal{K}|$ disjoint clusters $C_1, \ldots, C_{|\mathcal{K}|}$ while assuring that two feature vectors z_m, z_n can only belong to the same cluster C_i if the corresponding edge $(v_i, v_j) \notin E^t_{cl}$. That is, the nodes are grouped into subsets which contain only nodes from distinct frames and such that their corresponding bounding

boxes have an intersection over union greater than zero when the temporal difference between them is less than τ. Again, if the clustering constraints cannot be satisfied, the number of clusters is adjusted accordingly. That is, if a detection does not satisfy the constraints in Eq. 5.6, it is considered a tentative new target, a new cluster is created, and as new detections join that cluster in subsequent frames, a new tracklet is generated. If no subsequent detections join the cluster within a t_{lag} interval, the new detection is considered a mistake and the corresponding cluster is discarded.

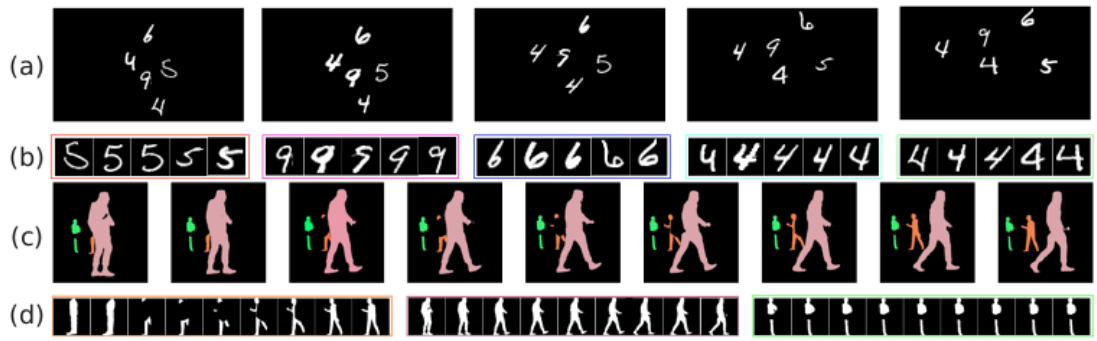


Figure 5.4: Sequence of frames for (a) MNIST-MOT with $|O| = 5$ targets per frame and (c) MOTSChallenge (cropped for three pedestrians) sequences. (b) and (d) show the corresponding subspaces generated by our method.

5.3.7 Spatio-temporal Clustering

To assign unique temporal identifiers to multiple objects in a video sequence, at each time instant t, we cluster the observations present in the frames within the window $\mathcal{T}^t = \{t - t_{lag}, t - t_{lag+1}, \ldots, t\}$ (Fig. 5.4). Since the window is computed at each frame, there is an overlap of $t_{lag} - 1$ frames between subsequent windows, which ensures that most of the data points used to form the subspace cluster in each window of a video sequence are shared. As shown in Alg. 6, the input to our spatio-temporal clustering algorithm is the set of frames $\{I^t\}_{t=1}^{T}$ where T is the number of frames in the video. Our algorithm applies the MTFE to each video frame in the window $\mathcal{T}^t$

Table 5.1: Specifications of the evaluation datasets.

Dataset	#Seq.	#Obj.	Instances	Shape	Challenge: randomly changed cues
MNIST-MOT	20	977	22232	28×28	pose, shape, motion
Sprites-MOT	20	983	22104	28×28	entry/exit, pose, scale, shape, motion
KITTI MOTS	9	210	10738	128×128	entry/exit, pose, scale, camera motion
MOTS-Challenge	4	228	26890	128×128	entry/exit, pose, scale, shape, motion

to construct the set $\mathcal{W}^t = \left\{\mathcal{X}^{t'} | t' \in \mathcal{T}^t\right\}$, where $\mathcal{X}^{t'}$ is the output of the MTFE for frame $I^{t'}$. The algorithm then clusters the detections within each window using the embeddings $\mathcal{Z}^t = \left\{ z_i^{t'} \,\middle|\, t' \in \mathcal{T}^t, i \in \{1, \ldots, \mathcal{O}^{t'}\} \right\}$ generated by the DHAE. Each call to the `kmeans`$(\cdot)$ algorithm produces a set of clusters $\mathcal{C}_t$ whose elements are detections assigned to the same object. For each cluster $Q \in \mathcal{C}_t$, we compute its normalized score $\bar{\tau}$ as the ratio $1/|Q| \sum_{d_i \in Q}(s_i)$, where s_i is the confidence score of detection d_i. Clusters with a score higher than a threshold λ are included in the cluster set $\mathcal{C}_K$.

5.4 Datasets and Experiments

We evaluate our algorithm on two synthetic and two real-world datasets. The MNIST-MOT and Sprites-MOT [41] synthetic datasets allow us to simulate challenging MOTS scenarios involving pose, scale, and shape variations. In addition, their bounding box and segmentation masks are readily available. To evaluate clustering performance in real videos using detections generated by a semantic segmentation algorithm, we employ the MOD [193] metrics. Then, we use the recently published MOTS [12] benchmark, which includes video sequences from the MOTChallenge [69] and the KITTI [70] datasets annotated with segmentation masks, to evaluate our model in real-world videos.

Table 5.2: Comparison of clustering performance on *MNIST-MOT* ($t_{lag} = 5$, $k = |O| = 3$) and *MNIST-MOT** sequences ($t_{lag} = 5$, $|O| = 5$, k varies in a range, $1 \ldots |O|$).

Configuration	MNIST-MOT				MNIST-MOT*			
	ACC	PUR	NMI	ARI	ACC	PUR	NMI	ARI
DEC [138]	0.87	0.89	0.88	0.81	0.84	0.85	0.89	0.78
ClusterGAN [141]	0.81	0.78	0.82	0.80	0.71	0.75	0.58	0.80
Shape Embed	0.90	0.90	0.82	0.75	0.85	0.87	0.84	0.72
Loc Embed	0.96	0.95	0.94	0.91	0.91	0.92	0.92	0.85
Loc+Shape	0.97	0.96	0.95	0.92	0.92	0.93	0.93	0.86
Loc+Shape+$\mathcal{G}^t$	**0.99**	**0.99**	**0.99**	**0.98**	**0.96**	**0.97**	**0.97**	**0.95**

To evaluate clustering performance in synthetic data, we adopt the popular clustering performance asssessment metrics: accuracy (ACC), purity (PUR), normalized mutual information (NMI) [142, 231], and adjusted random index (ARI) [232]. Since traditional clustering evaluation measures [142, 232, 231] require each observation to be mapped to exactly one cluster, they are not suitable for real-world scenarios where incorrect detections may occur. Hence, we adopt the popular CLEAR-MOT [193] tracking performance assessment measures: MOTA, Fragmentation (Frag), Identity Switches (IDs), Mostly Tracked (MT), and Mostly Lost (ML) targets [233]. Finally, we employ the MOTS [12] evaluation measures to quantify the effectiveness of our algorithm in maintaining the temporal consistency of target identities in real-world video sequences. Table 5.1 summarizes the characteristics of the datasets we used in our evaluation.

5.4.1 Synthetic Datasets

We generate synthetic MNIST-MOT and Sprites-MOT sequences using the procedure described in [41], which we discussed in Chapter 2, section 2.6.3.

Synthetic Digit Clustering. We evaluate the proposed approach on the MNIST-MOT dataset for clustering multiple moving digits in the temporal segments.

We apply the extracted cues to DHAE for clustering them in the jointly learned latent space using constrained k-means. Since we update the temporal segment at each t with stride 1, the final evaluated clustering metrics are the average over all the possible segments in the entire dataset. Table 5.2 summarizes the clustering results for different multi-object challenges on MNIST-MOT according to the evaluation procedure described in [141, 142]. Although location features play a critical role in clustering multiple moving targets, shape features also contribute significantly to the performance of our approach. As the t-SNE visualization in Fig. 5.1 illustrates, as the digits 2 and 1 approach each other, their corresponding embeddings remain separable. In Table 5.2, the shape-only model performs worse in comparison with location since both versions of the MNIST dataset periodically change the target shape, which reduces the benefits of shape features (Table 5.4 shows that the benefits of shape features in real videos are more pronounced). Due to the lack of availability of methods that perform clustering based on location and shape features, we select two state-of-the-art clustering methods for performance comparison [141, 138]. Although those baselines show comparable performance with our model when only shape features are used (maximum 9% improvement in terms of ACC), Table 5.2 shows how the different components of our method increase clustering robustness where the best results are shown in boldface and the second-best are underlined. More concretely, our joint DHAE model (Loc+Shape) improves over the baselines by 10% and 7%, 7% and 11%, 16% and 18%, 13% and 12% in terms of ACC, PUR, NMI, and ARI (*MNIST-MOT* results in Table 5.2). If we increase the object density and incorporate other multi-object challenges observed in real surveillance videos, our method also shows improvements in terms of most metrics (*MNIST-MOT** results in Table 5.2). Moreover, our jointly learned embeddings achieve much better results than the baseline when we use $\mathcal{G}^t$. By using constraints we obtain nearly perfect results with only 1% wrong assignments.

Synthetic Object Clustering. We also evaluate the clustering performance on the Sprites-MOT sequential data [41] where randomly selected objects ($|O| = 5$) move in random directions while showing most of the multi-object video challenges. Table 5.1 shows that we obtain highly satisfactory clustering metrics (ACC $> 90\%$) when all the components of the DHAE-based spatio-temporal clustering technique are employed.

Synthetic Multi-object Tracking. Due to the lack of availability of methods that perform clustering based on location and shape features, we select one state-of-the-art MOT method for performance comparison [41]. We relax the IoU constraint by imposing a limit on the Euclidean distance between embeddings since the target displacement among consecutive frames may be relatively large with respect to the size of the targets due to the low resolution of the frames.

Table 5.3: Performance evaluation on MNIST-MOT and Sprites-MOT with $t_{lag} = 3$, $|O| = 3$.

Method	MNIST-MOT							Sprites-MOT						
	↑IDF1	↑MT	↓ML	↓FN	↓IDs	↓Frag	↑MOTA	↑IDF1	↑MT	↓ML	↓FN	↓IDs	↓Frag	↑MOTA
shape embed	89.1	944	0	304	5	132	98.6	86.7	906	4	853	13	275	96.1
loc embed	87.7	905	0	708	61	331	96.5	88.2	920	0	689	56	347	96.6
loc+$\mathcal{G}^t$	86.3	977	0	0	72	0	99.7	85.6	983	0	4	94	0	99.6
loc+shape	89.6	934	0	444	3	189	98.0	88.9	912	0	734	**8**	314	96.6
loc+shape+$\mathcal{G}^t$	**100.0**	**978**	**0**	**0**	**0**	**0**	**100.0**	**99.5**	983	**0**	**45**	17	**0**	**99.7**
TBA [41]	99.6	978	0	49	22	7	99.5	99.2	**985**	1	80	30	22	99.2

Table 5.3 summarizes the performance of our method on the MNIST-MOT and Sprites-MOT datasets according to the evaluation procedure described in [193, 41, 233]. Although location features play a critical role in clustering multiple moving targets, shape features also contribute significantly to the performance of our approach. The shape-only model (*shape embed*) outperforms the location-only model (*loc embed*) on some of the evaluation criteria because the shapes remain unchanged until

Table 5.4: Evaluation of clustering quality using MOD metrics for the noisy detections of MOTSChallenge training set where ResNext-101-based Mask-RCNN [6] is used as MTFE.

Model	F1	Rcll	Prcn	TP	FP	FN	MODA
loc+shape+$\mathcal{G}_t$	82.7	82.7	82.6	16749	3526	3497	65.3
shape embed	**74.8**	**70.1**	**80.1**	**14192**	**3532**	**6054**	**52.7**
DEC[138]	71.6	65.6	78.8	13277	3568	6969	48.0
CGAN[141]	67.9	61.3	76.1	12415	3909	7831	42.0

they leave the scene. This effect is more pronounced on the MNIST-MOT dataset because the appearance of the characters is more distinctive than the shapes in Sprites-MOT. Although the incorporation of the constraints graph into the location-only model (*loc+*$\mathcal{G}^t$) leads to slightly better performance on some evaluation measures than the joint embedding (*loc+shape*), the overall method (*loc+shape+*$\mathcal{G}^t$) achieves near-perfect results.

5.4.2 Clustering in Real Videos

To evaluate the performance of our proposed clustering algorithm using real noisy detections, we use detections (bounding box and segmentation mask) generated by a Mask-RCNN model with a ResNext-101 backbone [6] on MOTSChallenge video sequences and compute the corresponding MOD evaluation metrics [69, 193]. We compare the performance of our shape-only model with the DEC [138] and ClusterGAN [141] models trained using shape features from the datasets. In our evaluation, we cluster the multi-task embedded features from the DHAE and retain only the clusters that satisfy the minimum cluster score λ. Again, we estimate the number of clusters $|\mathcal{K}|$ and utilize the bounding boxes for MOD evaluation. For a fair comparison, we use the same values of $\lambda = 0.62$, $t_{lag} = 5$, and minimum IoU = 0.4 for all the models under consideration.

Table 5.5: Evaluation of person and car tracking on the KITTI MOTS validation set.

Method	↑sMOTSA		↑MOTSA		↑MOTSP	
	car	ped	car	ped	car	ped
loc+shape+$\mathcal{G}_t$	<u>80.4</u>	55.5	<u>89.5</u>	<u>69.7</u>	<u>90.3</u>	<u>82.8</u>
loc+app+$\mathcal{G}_t$	**80.8**	**58.3**	**89.9**	**72.5**	**90.3**	**82.8**
EagerMOT [234]	74.5	<u>58.1</u>	-	-	-	-
GMPHD [129]	76.9	48.8	-	-	87.1	76.4
MOTSFusion [27]	77.5	49.9	89.2	66.6	-	-
MOTSNet [26]	78.1	54.6	87.2	69.3	89.6	79.7
TR-CNN [12]	76.2	46.8	87.8	65.1	87.2	75.7

Table 5.6: Evaluation of person tracking on the MOTSChallenge training set.

Method	↑sMOTSA	↑MOTSA	↑MOTSP	↑MT
loc+shape	51.3	60.1	86.3	21.5
loc+shape+$\mathcal{G}_t$	65.3	<u>76.3</u>	86.3	<u>53.1</u>
loc+app	56.1	65.5	**86.4**	26.3
loc+app+$\mathcal{G}_t$	<u>65.5</u>	76.5	<u>86.3</u>	**53.1**
GMPHD [129]	**65.8**	**77.1**	86.1	-
PointTrack [235]	58.1	70.6	-	-
MOTSNet [26]	56.8	69.4	82.7	-
TR-CNN [12]	52.7	66.9	80.2	-

5.4.3 MOTS Dataset

The MOTSChallenge dataset consists of four fully annotated videos of crowded scenes and the KITTI MOTS dataset consists of 21 videos acquired from a moving vehicle (see Table 5.1). We discuss these datasets in more detail in Chapter 2, section 2.6.1. Both datasets contain objects that show substantial scale and shape variations over time. We use the segmentation masks and the corresponding RGB content from 12 KITTI MOTS sequences to train the DHAE and use the remaining sequences for testing.

In our evaluation, we use the publicly available instance segmentation masks and bounding boxes [12] from the benchmark validation set. We compare the performance of our method against the state-of-the-art approaches presented in [234, 27, 12,

Figure 5.5: Qualitative results on the (a) KITTI MOTS and (b) MOTSChallenge dataset.

26, 129, 235]. Table 5.5 shows that, without resorting to sophisticated mechanisms for target re-identification, trajectory interpolation, or entry/exit detection, our method outperforms all the baseline methods in the KITTI MOTS dataset even if only the binary segmentation masks are used in the joint embeddings (loc+$shape$+$\mathcal{G}_t$). Including the RGB information (loc+app+$\mathcal{G}_t$) leads to further performance gains, particularly for the pedestrian class. As Table 5.6 indicates, in the MOTSChallenge sequences, our joint embeddings alone (loc+app) perform on par with [26, 12, 235], and the incorporation of the constraints graph leads to results comparable to [129] using the same set of detections (whereas [235] and [26] use privately refined detections). Figure 5.5 illustrates some of the results generated by our method. Both datasets are comprised of crowded scenes with significant amounts of temporary partial and full occlusions, particularly among pedestrians. Unlike [129], our method does not incorporate occlusion reasoning or motion modelling techniques, which contribute significantly to the performance of that method and could be easily incorporated into our framework.

To assess the impact of detection noise on the performance of our method, we evaluate the sMOTSA measure as a function of the minimum confidence score for a detection to be considered valid. Fig. 5.6 shows that performance increases until the detection threshold reaches approximately 0.65 and after 0.80 it starts to decrease again. In the experiments discussed above, we use a detection threshold of 0.70 for all the datasets. Further performance improvements would be achieved with

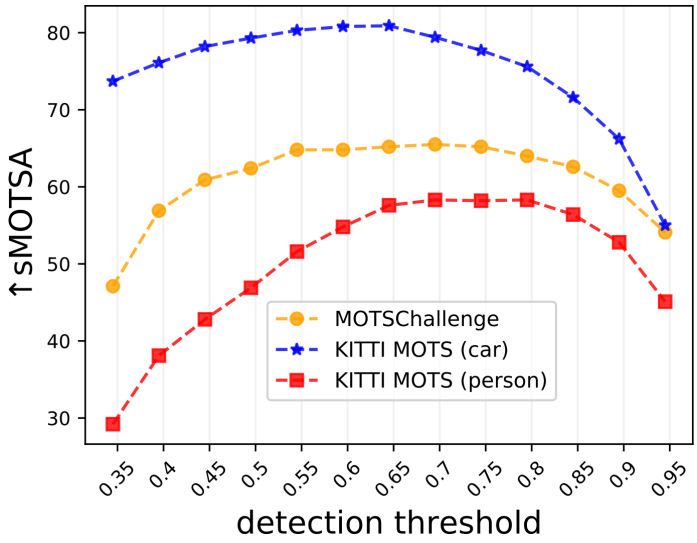


Figure 5.6: MOTS performance as a function of MTFE detection score threshold.

dataset-specific thresholds. For our ablation studies, we use ground truth annotations instead to evaluate each step of the method independently from the performance of the underlying detector.

5.4.4 Ablation Study

Table 5.7 shows the impact of the uncertainty-aware multitask learning loss of Eq. 5.3, of the constraints graph $\mathcal{G}^t$, and of estimating the number of clusters $|\mathcal{K}|$ using the method described in Section 5.3.6 for three different window sizes t_{lag}. The table demonstrates the positive impact of multi-task learning using task uncertainties instead of assigning equal weights in Eq. 5.3. We observe that the constraints graph leads to consistent and substantial improvements in all the evaluation criteria. We also see that higher values of t_{lag} lead to an increase in the MT measure but also to an increased number of fragmentations. Since we do not model target motion, the overlap among detections reduces as t_{lag} increases, leading to violations of the must-link constraints. As a result, the optimal sMOTSA score is obtained with $t_{lag} = 3$. Finally, estimating $|\mathcal{K}|$ from the video segments does not degrade the performance of

our algorithm. In summary, Table 5.7 shows that MTL and $\mathcal{G}^t$ lead to improvements in the sMOTSA measure of 10.8% and 15.5% in the MOTSChallenge, 8.2% and 14.1% for the car category in the KITTI MOTS, and 11.1% and 26.2% for the person class, even when $|\mathcal{K}|$ is unknown.

Table 5.7: Ablation study to evaluate the impact on tracking based on MOTS [12] oracle performance of: window size t_{lag}, constraints graph $\mathcal{G}^t$, multi-task learning (MTL), and number of subspaces $|\mathcal{K}|$ as a prior (✗) or estimated (✓).

				MOTSChallenge				KITTI MOTS (car)				KITTI MOTS (person)			
t_{lag}	$\mathcal{G}^t$	MTL	$\lvert\mathcal{K}\rvert$	↑sMOTSA	↑MT	↓IDs	↓Frag	↑sMOTSA	↑MT	↓IDs	↓Frag	↑sMOTSA	↑MT	↓IDs	↓Frag
	✗	✗	✓	74.0	43.9	962	1835	77.5	63.6	217	479	62.7	57.4	255	318
	✗	✓	✓	82.6	62.3	660	1416	84.4	68.2	115	348	70.5	54.4	138	258
3	✗	✓	✗	84.0	68.9	653	1367	84.7	70.9	113	343	71.7	52.9	138	248
	✓	✓	✗	97.5	**100**	636	639	98.1	98.7	127	127	95.6	98.7	119	**115**
	✓	✓	✓	**97.8**	99.6	548	**548**	**98.3**	**98.7**	119	**121**	**95.6**	98.5	124	124
	✗	✗	✓	66.4	31.1	939	2000	75.5	58.9	191	474	53.0	52.9	226	282
	✗	✓	✓	78.9	50.9	606	1476	76.7	56.3	97	408	63.0	32.4	100	238
5	✗	✓	✗	80.9	58.3	610	1422	77.4	55.6	94	409	63.2	36.8	92	254
	✓	✓	✗	97.2	100	692	733	97.7	98.7	160	171	94.1	100	186	199
	✓	✓	✓	97.7	100	581	589	97.8	98.7	162	162	94.6	**100**	180	181
	✗	✗	✓	56.9	19.7	897	1800	71.4	50.3	218	493	46.0	38.2	141	227
	✗	✓	✓	71.8	41.7	**447**	1499	73.0	47.7	79	376	58.2	30.9	83	246
8	✗	✓	✗	73.6	46.9	528	1440	74.8	47.0	**77**	384	57.5	35.3	**82**	250
	✓	✓	✗	97.0	99.1	765	800	96.8	98.0	223	236	93.5	100	215	217
	✓	✓	✓	97.5	100	657	662	97.2	98.0	210	210	94.1	100	194	196

5.5 Conclusions

Our proposed method uses task-dependent uncertainties to simultaneously learn the contribution of shape/appearance and location features from multi-object video datasets and further improves clustering performance by imposing simple constraints on acceptable sequential data patterns. Our experimental results show that our approach can accurately cluster multiple objects using embeddings generated by the DHAE. This method can be extended without significant modifications to include additional tasks of interest in similar scenarios such as object motion prediction. In

the future, we intend to extend our method with target motion models and more robust entry/exit/occlusion detection techniques so that it can be employed as a robust, standalone data association mechanism to the problems of multiple object tracking [236] and video instance segmentation [237].

CHAPTER 6

UNSUPERVISED MULTI-VIEW DATA ASSOCIATION

In the previous chapters, multiple self-supervised and unsupervised strategies were designed to address the multiple object detection and segmentation challenges in unseen data. Although we successfully apply learned detection models for SCT in overhead scenarios, a systematic approach needs to be designed to track multiple objects across cameras for multi-camera surveillance applications. To accomplish that goal, we developed an unsupervised multi-camera association technique that leverages the proposed self-supervised detector models in a single-camera tracking algorithm to generate temporal identifiers for the targets. Then we incorporate a multi-view trajectory association mechanism to maintain consistent temporal identifiers as objects travel across camera views. An evaluation of detection, tracking, and association performances on videos obtained from multiple overhead/lateral cameras in a realistic airport checkpoint and on a campus environment demonstrates the effectiveness of the proposed approach. Our results show that the self-supervised model improves SCT accuracy by up to 8% in overhead views and 3.9% in lateral views, compared with the corresponding baselines, without increasing the inference time of the model. Our multi-camera association method achieves up to 89% multi-object tracking accuracy in overhead views with an average computation time of less than 15 ms.

6.1 Introduction

Multi-camera association algorithms [99, 4, 110] track multiple objects across cameras as they traveled throughout the camera views. In most existing multi-camera tracking approaches, the initial step involves feeding the single-camera video streams into an multi-object detector [7, 6] to localize the targets. The detected targets are

temporally connected with unique identities using several feature extraction techniques such as optical flow [27], motion and appearance [195, 165], and track regression [23]. Some approaches also include Re-ID [157] technique to enhance association performance. These approaches leverage supervised models to generate target ROIs and the Re-ID features. However, the training of camera-specific models or single generalized models is challenging owing to the necessity of significant manual annotations. To address the camera-specific model deployment issue, we have proposed a self-supervised model in Chapter 3, which we employ in our SCT algorithm. After generating the SCT tracks, an automated surveillance system [112, 100, 11] requires a consistent multi-object temporal association across cameras. Most existing multi-camera tracking works address cross-camera association using a calibrated overlapping camera networks [11, 37] that employ both appearance and motion models to track targets in common coordinates, or non-overlapping cameras using multi-view Re-ID [154, 155] models. Some recent methods address the multi-view feature learning challenge using self-supervised methods [4, 108, 110], which can be used in both overlapping and non-overlapping cameras. However, real-time cross-camera association is challenging when cameras overlap partially and there is significant perspective distortion. To address these challenges, we propose a multi-Camera tracklet association (MCTA) algorithm that maintains the temporal identifiers of targets across cameras. Furthermore, we leverage the fact that our system is comprised of overhead cameras with partially overlapping fields of view to employ a simple but effective geometry-based trajectory association method. Our algorithm compares the projected centroids of target detections on neighboring cameras using the homographies between their image planes. We also track persons and bags across multiple views and generate global tracks by combining pairwise associations from partially overlapping camera views.

We evaluate the tracking performance of our algorithms on videos from a simulated airport checkpoint and campus environment to demonstrate that our self-supervised model-based SCT approach performs on par with a model trained in an entirely supervised manner, and substantially outperforms the pre-trained detection model-based SCT. Our multi-camera evaluation shows that the MCTA method effectively handles the problem of person identity hand-off across cameras.

Contributions. In summary, the key contributions of this work include:

- A new recursive tracklet association algorithm to address the identity hand-off issue during transitions between crowded overhead camera views.
- A distance-based association algorithm to keep track of the ownership of each baggage item across an airport checkpoint.
- A real-time end-to-end multi-camera tracking system.
- We provide an extensive evaluation of our methods on a dataset collected using multiple overhead and lateral cameras in realistic airport and campus scenarios.
- Our SSL models and the corresponding MCTA source code are available at `https://github.com/siddiquemu/SCT_MCTA`.

To our knowledge, this is the first approach to solve the overhead or lateral multi-view association problem in a network of cameras with partially or fully overlapping fields of view using a self-supervised detection strategy.

6.2 Related Work

In multi-target tracking applications with fully overlapping camera networks, multiple single-camera tracks mapped to a common coordinate system can be interpreted as the probability of occupancy of the area observed by the cameras [100]. Although it is possible to use clustering techniques to map the modes of this distribution to unique target locations, bounding box alignment errors due to perspective

challenges generate spurious associations. Hence, we propose a simple stereo camera association using 2D homographies that can be extended for multiple-camera networks.

A systematic solution to the data association problem is another important component of multi-target tracking-by-detection methods [238, 239, 236, 240, 241, 242, 243, 244, 245, 12, 123, 128, 246, 17]. Single-camera trackers [23, 247] utilize detectors trained on multiple datasets [69] to generate bounding boxes and form track hypotheses for all the targets in each frame. In this chapter, we employ a state-of-the-art single-camera tracker [23] using a detector based on our self-supervised models, which achieves unprecedented tracking performance in previously unseen airport surveillance videos.

Finally, multi-camera tracking systems require sophisticated trajectory association mechanisms to maintain target identities across cameras [28, 150, 151]. Even within a single-camera, occlusions must be handled using similar strategies [152, 153]. Most association approaches compute trajectory similarity scores based on a combination of appearance and motion features [28, 150, 151, 152, 153]. These features are learned using a large number of continuous trajectories, which are difficult to obtain with typical ceiling-height overhead cameras due to their limited fields of view. Multi-camera methods also need to project trajectories onto a global 3D coordinate [11] frame that requires camera calibration information. Some methods utilize camera calibration information to project tracks onto a common plane and perform association using occlusion modeling [37] or re-identification techniques [154, 155, 106, 156, 157]. However, dependency on camera calibration further limits the applicability of these methods to security systems, since calibrating multiple cameras with partially overlapping fields of view is a complex task [158, 159, 160, 161].

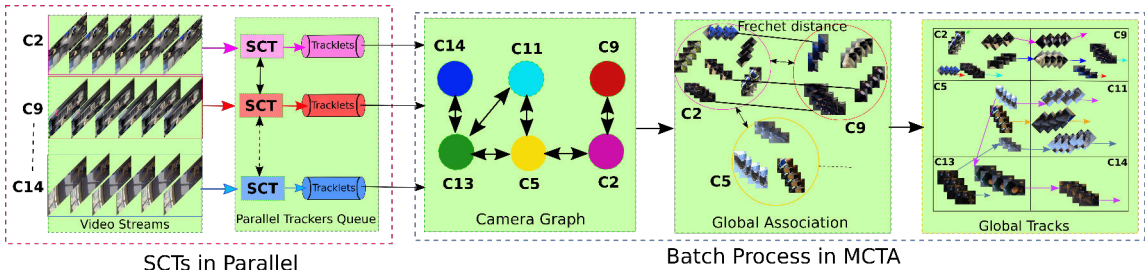


Figure 6.1: Real-time multi-camera tracklet association.

6.2.1 Proposed Approach

As Fig. 6.1 shows, our multi-camera tracking framework comprises two main steps: i) a single-camera, multiple-target tracking-by-detection algorithm, and ii) a multi-camera trajectory association mechanism. Our single-camera tracker uses the detections generated by our SSL framework and a single-camera trajectory association (SCA) method to keep track of the identities of individual passengers and baggage items within the field of view of each camera. Our MCTA strategy then projects the trajectories of passengers observed in cameras with overlapping fields of view onto a common image plane. These trajectories are then compared using the Fréchet distance and associated using a recursive graph-based mechanism.

6.2.1.1 Single-camera Tracking

We use the Tracktor algorithm [23] as our baseline single-camera tracker. The output of the algorithm at each image frame is a set $T^{\mathcal{C}}(t) = \{\omega_1, \ldots, \omega_{n_t^{\mathcal{C}}}\}$, where $\omega_j = [b_j, l_j]$, with l_j corresponding to a unique identifier label for each person or baggage item in the frame. These labels remain the same throughout the video sequence and hence perform temporal association among detections. The tracklet for the k-th object is thus given by the set of detections over the entire video sequence whose temporal identifier is $l_j = k$, i.e., $\tau_k = \cup_{t=1}^{T} \{\omega_j \mid \omega_j \in T^{\mathcal{C}}(t), l_j = k\}$. Temporary occlusions between passengers may lead to the fragmentation of trajectories

within the field of view of a camera. Tracktor's simple re-identification strategy is unable to accommodate the longer occlusions, appearance variations, and somewhat erratic motion patterns commonly observed in airport checkpoints or crowded campus scenes. Thus, we incorporate an SCA mechanism to resolve this issue. Our method associates new tracklets with recently terminated tracklets such that the Euclidean distance between the centroids of the last detection of the previous tracklet and the first detection of the new tracklet is minimized. That is, let τ_m and τ_n be two distinct tracklets, and $b_m^{t^i}$, $b_n^{t^f}$ be the first detection of τ_m and the last detection of τ_n, respectively. Defining $\delta_e = ||b_m^{t^i} - b_n^{t^f}||^2$, the association cost between τ_m and τ_n is given by

$$\mathcal{C}_{sc}(\tau_m, \tau_n) = \begin{cases} \delta_e & \text{if } 0 < t^i - t^f \leqslant t_{th},\ \delta_e < \delta_{max} \\ \infty & \text{otherwise,} \end{cases} \tag{6.1}$$

where t_{th}, δ_{max} are the maximum temporal offset and maximum distance to consider two tracklets for association. We then compute the optimal tracklet assignment using the Hungarian algorithm based on the costs $\mathcal{C}_{sc}(\tau_m, \tau_n)$.

6.2.1.2 Multi-Camera Tracklet Association

Since passengers may temporarily leave and later re-enter the fields of view of individual cameras, their corresponding trajectories may be fragmented into multiple segments. Therefore, to associate tracklets across camera views, we consider the fact that two tracklets corresponding to the same target include temporally overlapping detections. Let the camera whose partial tracklets we wish to complete be our *primary* camera, and let the *auxiliary* camera be the one whose tracklets will be used to complement the tracklets observed by the primary camera. Further, let $\mathcal{T}_p$ and $\mathcal{T}_a$ be the sets of tracklets in the primary and auxiliary cameras, respectively. As Alg. 7 shows, we use the homography $H_{p,a}$ to project detections from the auxiliary camera onto the primary camera. However, due to projective distortions, the corresponding

Algorithm 7 Multi-Camera Tracklet Association Algorithm

Input: Set of tracklets from the primary camera $\mathcal{T}_p$ and the auxiliary camera $\mathcal{T}_a$, homography $H_{p,a}$ mapping the auxiliary camera image plane to that of the primary camera

Output: Updated set of primary tracklet labels

1: Project the detections of tracklets in $\mathcal{T}_a$ onto the image plane of the primary camera using $H_{p,a}$
2: Compute the association costs $\mathcal{C}_{mc}(\tau_a, \tau_p)$ $\forall \tau_p \in \mathcal{T}_p$, $\forall \tau_a \in \mathcal{T}_a$ according to Eq. 6.2
3: Initialize the graph $\mathcal{G}_{mc} = (V, E)$, $E = \emptyset$, $V = \{\tau | \tau \in \mathcal{T}_p \cup \mathcal{T}_a\}$
4: **while** $\min_{\tau_p \in \mathcal{T}_p, \tau_a \in \mathcal{T}_a} (\mathcal{C}_{mc}(\tau_a, \tau_p)) < \infty$ **do**
5: Associate tracklet segments using the Hungarian algorithm based on the costs $\mathcal{C}_{mc}$
6: Update the costs of the tracklets $\tau \in \mathcal{T}_a$ and $\tau' \in \mathcal{T}_p$ for which $\tau \cap \tau_a \notin \emptyset$ and $\tau' \cap \tau_p \notin \emptyset$ to $\mathcal{C}_{mc}(\tau, \tau_p) = \mathcal{C}_{mc}(\tau_a, \tau') = \infty$
7: $E = E \cup (\tau_a, \tau_p)$
8: **end while**
9: **for** each $\tau_p \in \mathcal{T}_p$ **do**
10: $\mathcal{N}_p = \text{DFS}(\tau_p, \mathcal{G}_{mc})$
11: Update the labels of tracklets in $\mathcal{N}_p$ using Eq. 6.3
12: $E = E - \{(\tau_i, \tau_j) | (\tau_i, \tau_j) \in \mathcal{N}_p\}$
13: **end for**

bounding boxes in the two cameras may not necessarily overlap. Hence, we compute the optimal association cost using the Fréchet distance [64] between the centroids of the detections in each tracklet as follows:

$$\mathcal{C}_{mc}(\tau_a, \tau_p) = \begin{cases} \mathtt{f}(\tilde{\tau}_p, \tilde{\tau}_a) & \text{if } \tilde{\tau}_a \neq \emptyset,\ \tilde{\tau}_p \neq \emptyset,\ \mathtt{f} < f_{max} \\ \infty & \text{otherwise,} \end{cases} \tag{6.2}$$

where $\tilde{\tau}_p$ and $\tilde{\tau}_a$ are the temporally overlapping segments of tracklets $\tau_p \in \mathcal{T}_p$ and $\tau_a \in \mathcal{T}_a$, $\mathtt{f}(\tilde{\tau}_p, \tilde{\tau}_a)$ is the Fréchet distance of the centroids of the corresponding detections, and f_{max} is the maximum distance threshold that allows tracklet pairs to be considered for association.

We use the Hungarian algorithm again to determine optimal tracklet associations according to the costs $\mathcal{C}_{mc}(\tau_a, \tau_p)$. However, since the trajectory of a passenger that re-enters the field of view of a camera multiple times consists of a sequence of

tracklets, we iteratively update the association costs until no further associations are possible. Furthermore, we keep track of indirectly associated tracklets by constructing the reachability graph $\mathcal{G}_{mc} = (V, E)$, which contains one edge for each pair of associated tracklets. We then set the temporal identifiers of all the tracklets in $\mathcal{T}_p$ associated with a common tracklet τ_a to the first identifier among them. That is, the temporal label of a tracklet τ is given by

$$l_\tau = \min_{(\tau_i, \tau_j) \in \mathcal{N}_p} (l_{\tau_i}), \tag{6.3}$$

where l_{τ_i} is the temporal label of tracklet τ_i, and $\mathcal{N}_p$ is the set of tracklets that can be reached from tracklet τ_p on $\mathcal{G}_{mc}$, which we obtain through Depth-First Search (DFS).

6.3 Results and Discussions

In this section, we first discuss the datasets and then the evaluation of the monocular tracking and the multi-view tracklet association algorithms. Our evaluation is based on the Multi-Object Tracking (MOT) metrics [69, 193].

6.3.1 Multi-camera Datasets

To evaluate the MCTA algorithm, we use partially overlapping CLASP1 and CLASP2 datasets described in Chapter 3, Table 3.1. Since tracking evaluation depends upon frame-by-frame temporal information, rather than substantially limiting the number of video sequences used in the assessment of our tracking algorithms, we consider all the annotations listed in Table 3.1 in our experiments. The only method that uses the training set annotations is the SL approach. Hence, this evaluation strategy also more accurately reflects the generalization performance of the SSL approaches although it favors the SL method. We also use the recently published WILDTRACK multi-camera datasets [11] to evaluate the cross-camera association performance in fully camera networks with fully overlapping fields of view. This

Figure 6.2: Sample images from the WILDTRACK dataset.

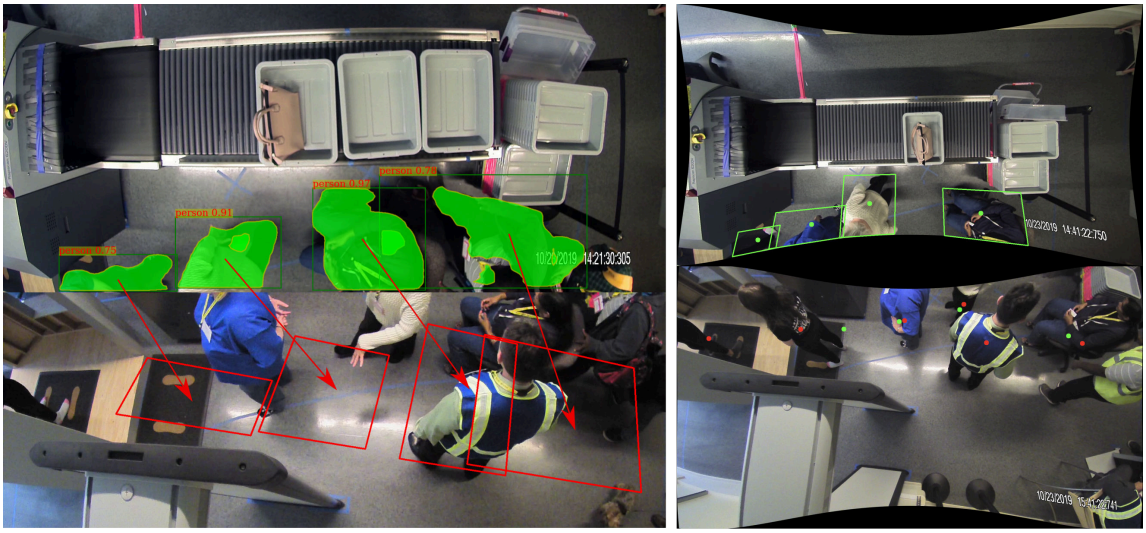


Figure 6.3: Homography projection from primary to auxiliary camera in CLASP2 datasets: (a) before distortion correction, (b) after distortion correction.

datasets consist of seven significantly overlapped lateral camera views with a resolution of 1980×1080 and a frame rate 60 FPS. The total number of bounding box annotations is $9,518$, which corresponds to 313 persons annotated at 2 fps for each videos. Fig. 6.2 shows sample images from the WILDTRACK datasets.

6.3.2 Association in Overhead Camera Networks

In the proposed SSL model-based tracking-by-detection method, we consider different association mechanisms. Firstly, we associate the detections and their corresponding appearance features across single-camera frames using Tracktor [23] to generate the tracklets. Secondly, we only consider tracklets in the primary camera for association when there are temporally and geometrically overlapping tracklets in neighboring auxiliary cameras. Fig. 6.3 shows the 2D homography-based geometrical relationship between the primary and auxiliary cameras.

Table 6.1: Single-camera tracking evaluation for person and baggage classes.

Class	Model	α	SCA	GT	↑IDF1	↑IDR	↑IDP	↑Rcll	↑Prcn	↓FP	↓FN	↑MT	↓ML	↓IDs	↓FM	↑MOTA	↑MOTP
Person	Base	✗	✗	391	84.5	83.3	86.1	91.2	94.6	750	753	283	42	93	152	84.0	85.5
	SSL	✗	✗	391	87.8	87.2	88.3	95.1	96.4	**350**	554	319	31	80	123	90.1	85.2
	SSL	✓	✗	391	88.4	87.9	88.9	95.6	96.6	354	438	326	27	86	122	90.7	85.2
	SSL	✓	✓	391	**88.5**	**88.1**	**88.9**	**95.6**	96.5	358	**435**	326	26	**76**	123	**90.8**	**85.2**
	SL	✗	✗	391	87.0	86.4	87.7	95.2	**96.9**	357	457	**332**	**24**	86	**121**	90.5	85.2
Bag	Base	✗	✗	255	67.5	57.0	86.4	61.1	92.3	431	1800	108	73	**31**	**89**	54.3	**81.0**
	SSL	✗	✗	255	78.9	74.2	85.4	81.0	**93.7**	**308**	1014	159	38	71	105	72.9	80.4
	SSL	✓	✗	255	81.3	78.2	85.5	84.4	92.3	401	822	169	28	68	97	75.1	80.4
	SSL	✓	✓	255	84.3	81.1	**88.5**	84.4	92.3	401	822	169	28	48	97	75.6	80.4
	SL	✗	✗	255	**85.3**	**86.6**	84.1	**94.7**	91.7	387	**339**	**226**	**10**	103	69	**83.2**	80.0

6.3.2.1 Single-Camera Tracking

We compare the performance of our single-camera tracking algorithm using the proposed SSL detectors with the pre-trained baseline detector and the SL detector. We also evaluate the impact of our SCA algorithm, described in Section 6.2.1.1, where we use $t_{th} = 3$ seconds and $\delta_{max} = 200$ in Eq. 6.1. To preserve the entirely self-supervised nature of our pipeline, we refrain from fine-tuning the re-identification module of the baseline tracker, which is pre-trained on the MOT17 [69] dataset. To

dissociate the evaluation of the tracking method from our MCTA approach, we use a modified version of the annotations in Table 3.1, where a passenger that re-enters the field of view of a camera receives a new identifier. Thus, the number of unique ground truth passenger identifiers (column GT in Table 6.1) is much higher than those listed in Table 3.1. We evaluate our system's ability to maintain consistent passenger identifiers across multiple perspectives in Section 6.3.2.2. Fig. 6.4 shows the SCT

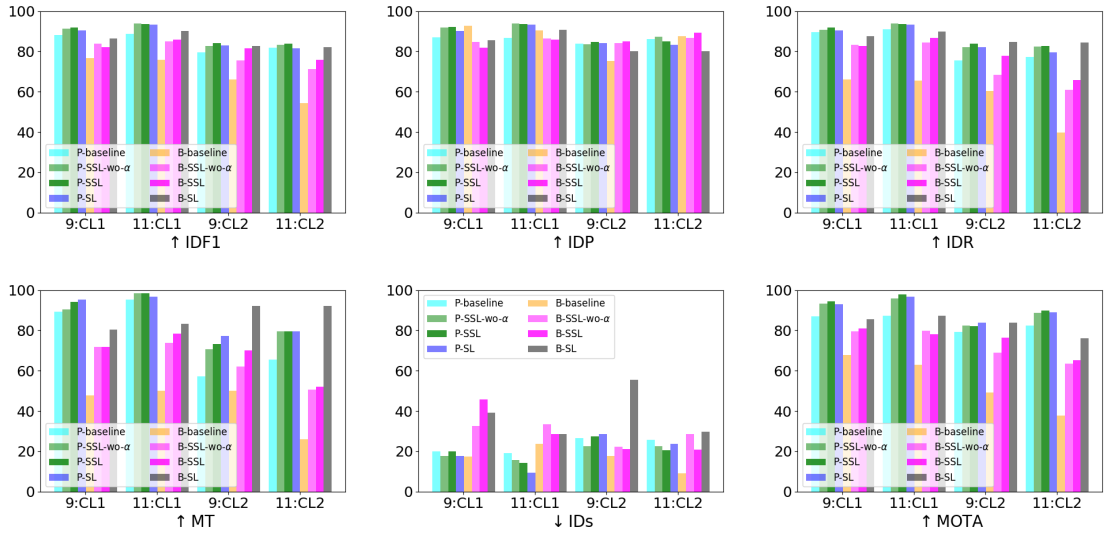


Figure 6.4: Comparison of SCT performance of person and baggage classes in individual cameras of the CLASP1 and CLASP2 datasets using the Baseline, SSL-wo-α, SSL, and SL detectors.

performance of our algorithm for passengers and baggage items in the individual cameras of the CLASP1 and CLASP2 datasets. For passenger tracking, the SSL methods outperform the SL approach in terms of IDF1, IDP, and IDR in all the scenarios under consideration. In both datasets, the SL approach shows slightly higher MT results for camera 9, largely due to the partial passenger detection problem. Since CLASP1 has lower object density, we observe more consistent performance among different methods for both cameras in that dataset. While all the methods

perform better on the CLASP1 dataset, the benefits of SSL training compared to the baseline detector are particularly evident in the MT results on the CLASP2 dataset.

Regarding baggage items, although the SSL models lead to a moderate increase in the number of IDs, these switches are offset by substantial gains in MT. As a matter of fact, the SL model shows a much more significant degradation in IDs for the more complex CLASP2 dataset. This is particularly evident for camera 9, and it explains the lower IDP obtained by the SL method in that dataset. The most evident performance gains for baggage tracking are observed in camera 11 on the CLASP2 dataset because of the difficulty introduced by partially visible baggage items using the baseline model.

As Table 6.1 shows, the SSL-wo-α and SSL approaches outperform the tracker using the baseline detector by a large margin, and the notable improvements in identity-based F_1 (IDF1), recall (IDR), and precision (IDP) [248] as well as in standard recall and precision are primarily a result of the reduction in false positives and false negatives generated by the SSL model. Self-supervision also improves the tracking-specific metrics of mostly tracked (MT), mostly lost (ML), identity switches (IDs), and fragmented (FM) trajectories [193]. As a result, our method produces substantial gains in MOTA. Again, both SSL models perform on par with the SL model for person tracking. For baggage items, we see similar performance improvements; however the challenges illustrated in Fig. 3.9 again preclude the SSL models from reaching the performance of the SL strategy. Finally, our SCA algorithm leads to further performance gains, particularly in terms of IDs.

6.3.2.2 Multi-Camera Tracklet Association

From the stereo geometry (discussed in Chapter 2), we can associate the targets in two adjacent cameras using Eq. 2.5, where we need the calibrated intrinsic and extrinsic parameters, which is one of the limitations for developing real-world

Table 6.2: MCTA evaluation. The column labeled Dist. indicates whether we employ the Hausdorff (d_h) or Fréchet (d_f) distance to evaluate tracklet similarity.

Dist.	SL	SSL-wo-α	SSL	MCTA	↑IDF1	↑IDR	↑IDP	↓IDs	↑MOTA
	✗	✓	✗	✗	82.1	83.2	81.0	157	88.2
-	✗	✗	✓	✗	82.0	83.1	80.9	170	88.5
	✓	✗	✗	✗	81.5	82.8	80.4	170	88.0
	✗	✓	✗	✓	87.4	88.9	86.4	<u>115</u>	88.8
d_h	✗	✗	✓	✓	84.8	88.6	86.2	140	89.0
	✓	✗	✗	✓	86.2	87.5	85.0	134	88.6
	✗	✓	✗	✓	<u>88.0</u>	<u>89.3</u>	<u>86.8</u>	**115**	88.9
d_f	✗	✗	✓	✓	**88.2**	**89.5**	**87.0**	132	**89.1**
	✓	✗	✗	✓	86.7	88.1	85.5	122	<u>89.0</u>

large multi-camera surveillance systems. We use the image feature correspondence-based homography [249] to address this problem, which has significant projection error. However, our proposed tracklet association algorithm can handle this error by leveraging the Fréchet polygonal curve distance as a matching metric. We also observe a considerable camera distortion, which we correct by estimating the distortion coefficient and the camera intrinsic parameters by calibrating a pair of cameras. Since the cameras used in CLASP1 and CLASP2 datasets are at ceiling height, the approximate distortion coefficient works for all the cameras where the camera height from the ground is $Z = 3,000$ pixels (see Fig. 6.3). We find a focal length $f = 1,217$ pixels and estimate the distortion coefficients from one camera pair calibration to use for all other cameras. Fig. 6.3 shows that this approximation generates promising results without calibrating all the camera pairs. For fully overlapped WILDTRACK datasets, we use the calibration parameters in our MCTA algorithm to compute the matching metric. As shown in Fig. 6.5, there is no significant projection error after correcting the distortion using calibration parameters. The limitation of manual calibration is that we may need to repeat the manual calibration procedure to extend the camera networks, or deploy the MCTA algorithm in different camera networks.

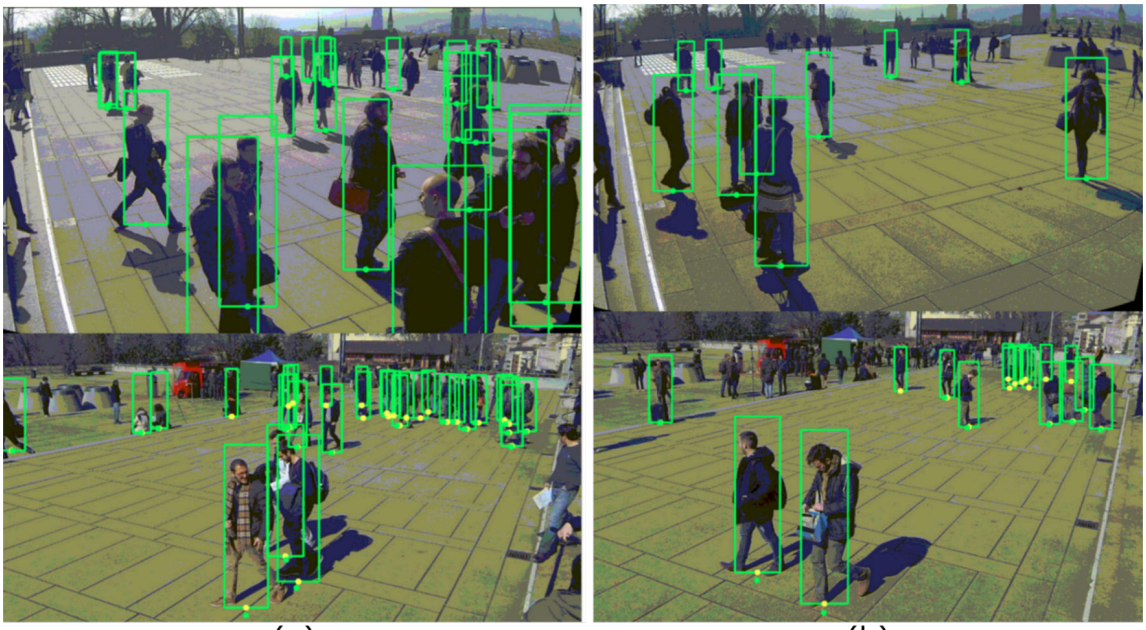


Figure 6.5: 2D homography projection on distortion corrected frames in WILDTRACK datasets: (a) C2-to-C1, (b) C3-to-C1, where the green dots in the bottom cameras (slightly deviated from the bottom centroids) represent the projection from the corresponding top cameras.

We evaluate the performance of our MCTA algorithm using the same experimental procedure described in the previous section, with the exception that passengers are now assigned unique identifiers as they leave and re-enter the fields of view of the cameras. Based on the overall flow of passengers through our simulated checkpoint, cameras 9 and 11 are the primary cameras for our tracklet association method (Alg. 7). Cameras 2 and 5, the cameras immediately below them in Fig. 2.13, are the respective auxiliary cameras. For a fair comparison among the detectors, we generate tracklets in the auxiliary cameras using the corresponding SL or SSL model used in the primary cameras (i.e., trained using only frames from the primary camera). To provide a set of reference performance measures, we first evaluate our tracking algorithms in the absence of an MCTA mechanism. We then assess the performance of our association method when tracklet similarity is computed using the Fréchet

distance and the more traditional Hausdorff distance [153] with $f_{max} = 0.25$ in Eq. 6.2.

Table 6.2 shows that tracklet association improves the IDF1 measure by up to 6.2%, which is mainly a consequence of the dramatic reduction in the number of identity switches. Using the Fréchet distance to determine tracklet similarity provides consistent performance improvements in all the metrics under consideration, especially for the SSL strategy. The more modest gains in MOTA (up to 1.0%) demonstrate the need for measures that focus specifically on the impact of identity switches on tracking performance. Fig. 6.6(a) illustrates the tracklet association

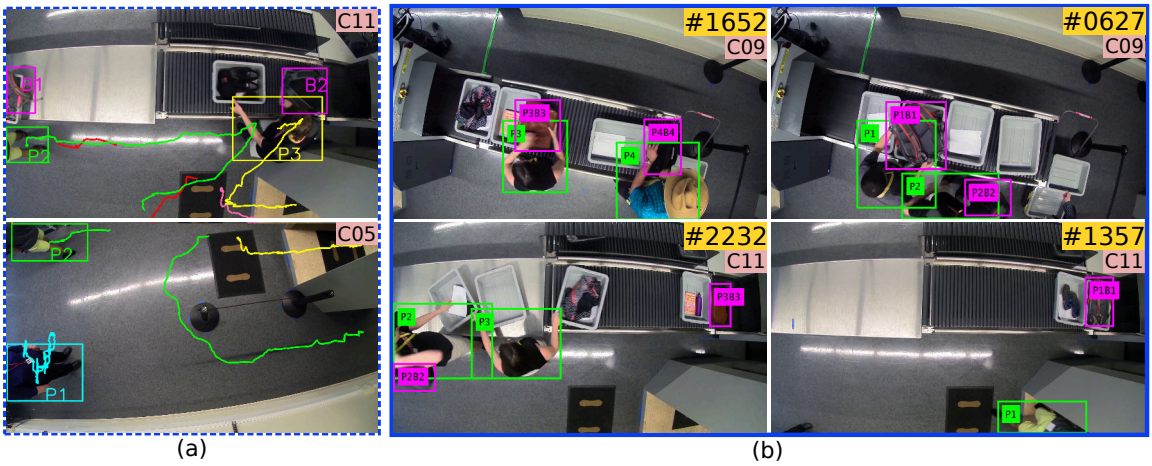


Figure 6.6: Sample results showing (a) cross-camera passenger association between cameras 5 and 11 using MCTA, and (b) tracking and association between passengers and baggage items where the top and bottom rows show image sequences from cameras 9 and 11 respectively. We associate passenger tracklets in cameras 9 and 11 by leveraging the associations between cameras 2 and 5 (passengers flow in Fig. 2.13: C9→C2→C5→C11). Baggage items are associated using temporally constrained distance-based matching when each item receives a unique identifier $PiBj$, representing the j-th item from the i-th passenger.

procedure between cameras 5 and 11. As the passengers with identities P2 and P3, whose trajectories are represented in green and yellow, move from the field of view of camera 5 to camera 11, their tracklets are projected from the former camera to the

latter. The projected trajectories (red for P2 and pink for P3) are successfully associated with the tracklets from camera 11 based on the Fréchet distances among their temporally overlapping segments. In the instant shown in the figure, passenger P2 is re-entering the field of view of camera 5, and the corresponding tracklet is also correctly associated with that passenger's tracklet in camera 11. Hence, the passenger's identity is successfully handed off between the cameras. Fig. 6.6(b) demonstrates a potential application of the proposed system. Baggage items are associated with passengers when they are divested in camera 9 and their identifiers can be verified at retrieval time, which is observed in camera 11.

6.3.3 Association in Lateral Camera Networks

We conduct slightly different experiments to see how multi-camera detections, tracking, and MCTA work in fully calibrated lateral camera networks, so as to compare the performance with the baseline.

Multi-camera Detection. We have addressed the perspective distortions problem in the WILDTRACK datasets, since the projections from the cameras show significant localization error, which reduces the tracking performance under severe occlusion. However, in the WILDTRACK datasets, the cameras are not top-down, so we address severe occlusion challenges without using any sophisticated algorithm [37]. In our approach, we employ Faster-RCNN (FRCNN) [1] with Feature Pyramid Network (FPN) [49] to train on the subsets (90%) of WILDTRACK multi-camera annotations, while the remaining portion is used for testing, similar to [11]. We project the bottom centroids of all the detected bounding boxes from all the cameras C1-C7 to the common ground plane as 3D points ($Z = 0$). Since the distributions of the projections are separable in 3D space, we apply a non-parametric mean-shift algorithm to obtain the possible modes as final detections. We called the mean-shift-based multi-camera detection method as *FRCNN+MS*. The main challenge of using the mean-shift

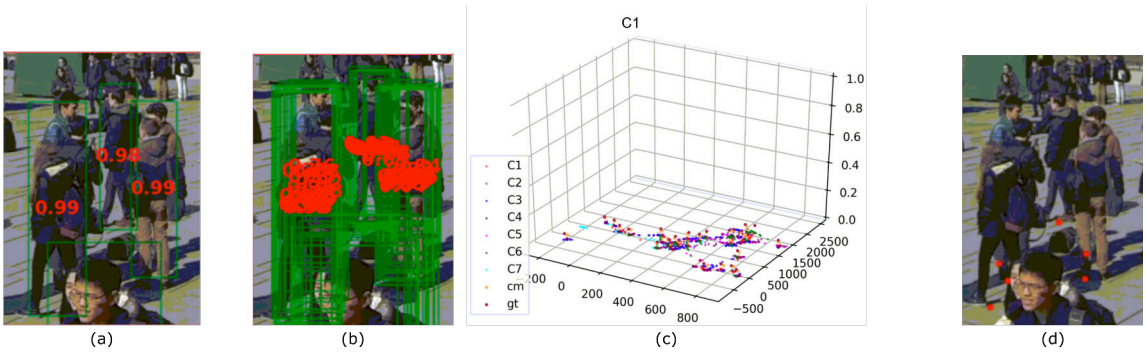


Figure 6.7: Occlusion reasoning by clustering the projections in a 3D map: (a) NMS fails to recover the occluded target in a single-camera view, (b) set of predictions after applying a detection threshold 0.5 without NMS, (C) clustered projections in the 3D map, (d) cluster modes projected back to the image plane.

clustering algorithm in *FRCNN+MS* is the bandwidth selection for a Gaussian kernel (see Eq. 3.1). As shown in Table 6.3, we observe competitive results using an empirically determined bandwidth of 0.029. To ignore the dependency on optimum bandwidth selection, we propose an iterative clustering approach by a matching algorithm which we called *FRCNN+CM* that iteratively matches the detections from the primary (C1) and auxiliary (C2-C7) cameras as bipartite matching using Hungarian [126] to update the final set of detections. After finishing the iterations, we can easily discard false detections if any detection does not find any correspondence. This algorithm also depends on the maximum matching distance threshold, which we set at 50 cm (the average footprint of a human body in the 3D world). In Table 6.3, we find that the *FRCNN+CM* method is 1.3% better than *FRCNN+MS* in terms of MODA. Again, in the case of NMS, when the persons are occluded from each other, it is common for two or more detections to significantly overlap, and the NMS algorithm cannot keep all of the predictions during occlusions. To overcome this challenge, we propose a new multi-view NMS approach (*FRCNN+MV-NMS*) in which we keep all the detections at a particular detection probability score and project them onto 3D world coordinates using accurate camera calibrations. To find

Table 6.3: Performance evaluation of our multi-camera detection algorithm on the WILDTRACK datasets.

Method	MODA	MODP	Prcn	Rcll
Deep-Occlusion+KSP	0.75	-	-	-
Deep-Occlusion	0.74	0.54	**0.95**	0.80
ResNet-DeepMCD	0.67	0.64	0.85	0.82
DenseNet-DeepMCD	0.63	0.66	0.87	0.74
RCNN-projected	0.11	0.18	0.68	0.43
FRCNN+MV-NMS	**0.78**	**0.93**	0.94	0.82
FRCNN+MS	0.76	0.85	0.87	**0.89**
FRCNN+CM	0.77	0.88	0.90	0.88

the modes of the distributions of all the projected detections from all the cameras in a network, we leverage the mean-shift algorithm and obtain the unknown modes of the distributions as final detections. Furthermore, we compute the cluster scores using Eq. 3.2, an emphirical optimum bandwidth of 0.028 for mean-shift clustering, and an optimum clustering score threshold of 0.016 from the precision-recall curve for the validation set. In Table 6.3, we report a 3.9% improvement over the baseline Deep-Occlusion+KSP [37]. Fig. 6.7 demonstrates the predicted cluster modes in the 3D projection map and their corresponding localization in the 2D image plane after applying the inverse projection.

Multi-camera Tracking. To demonstrate the benefits of the MCTA algorithm in lateral camera networks, we match tracklets using an approach similar to the one described in section 6.3.2.2. For WILDTRACK datasets, we consider camera C1 the primary camera and all others auxiliary cameras, so that the homographies can project SCT tracks onto a common image plane. As our goal is to evaluate MCTA, which is designed to track multi-camera tracklets in the image plane, instead of tracking multiple objects in the 3D map, we apply MCTA to measure the tracking performance in each camera's image plane. Table 6.4 shows that Tracktor with our

MCTA mechanism improves IDF1 by 8.4%, and MOTA by 2.8% in C7. Compared with the SCT baseline algorithm, our real-time MCTA shows improvements for all the MOT metrics over all the cameras. Fig. 6.8 shows the qualitative results of our MCTA algorithm on the WILDTRACK datasets where the same persons are detected with unique identities when visible across multiple cameras.

Table 6.4: Monocular tracking evaluation for the person class in the WILDTRACK datasets. Bold faced entries indicate methods augmented with our proposed algorithm.

Cam.	Method	↑IDF1	↑IDP	↑IDR	↑MT	↓PT	↓ML	↓FP	↓FN	↓IDs	↓FM	↑MOTA	↑MOTP
1	KSP	28.5	18.7	**60.7**	**34**	**4**	0	10050	257	162	58	-140.9	59.0
	KSP+ptrack	30.2	20.1	60.6	30	8	**0**	9173	410	131	51	-123.5	58.0
	Tracktor	63.8	71.7	57.5	22	14	2	40	207	52	46	64.7	**70.5**
	Tracktor+MCTA	**65.9**	**74.1**	59.4	22	14	2	**40**	**207**	**48**	**46**	**65.2**	69.5
2	KSP	29.1	19.4	58.8	28	6	1	8428	265	172	47	-121.3	50.7
	KSP+ptrack	**31.4**	21.2	**60.6**	**28**	**6**	**1**	7698	249	128	31	-101.6	50.2
	Tracktor	24.7	29.0	21.5	5	12	18	304	509	66	50	-11.7	35.1
	Tracktor+MCTA	24.7	**29.0**	21.5	5	12	18	**304**	**509**	**66**	**50**	**-11.7**	**35.1**
3	KSP	25.8	17.0	53.7	**35**	**4**	**1**	9874	286	177	52	-133.5	51.2
	KSP+ptrack	27.2	18.1	54.2	33	5	2	9208	402	150	46	-120.5	49.1
	Tracktor	62.1	69.4	56.2	21	16	3	55	221	47	52	62.8	**76.9**
	Tracktor+MCTA	**64.1**	**74.1**	**58**	21	16	3	**55**	**221**	**45**	**52**	**63.1**	66.9
4	KSP	20.5	12.6	54.4	14	5	1	4362	137	42	16	-255.3	60.5
	KSP+ptrack	22.1	13.9	**54.4**	**13**	**2**	5	3904	180	32	11	-222.1	60.3
	Tracktor	**52.8**	**78.1**	39.8	7	4	8	16	139	9	**1**	34.7	**58.5**
	Tracktor+MCTA	52.2	77.3	39.4	7	4	8	**16**	**139**	**9**	2	**34.7**	57.4
5	KSP	39.7	32.6	50.9	**20**	13	**3**	2560	598	117	79	5.8	54.2
	KSP+ptrack	41.7	35.0	**51.7**	18	**12**	6	2334	672	94	54	10.9	55.2
	Tracktor	53.0	61.7	46.5	13	16	7	17	186	51	40	63.0	75.7
	Tracktor+MCTA	**54**	**64.6**	47.4	13	16	7	**17**	**186**	**47**	**40**	**63.6**	**75.7**
6	KSP	26.6	17.5	55.4	**34**	**4**	1	10200	375	172	77	-136.1	52.2
	KSP+ptrack	29.4	19.9	**56.4**	30	8	**1**	8860	498	127	51	-108.4	52.8
	Tracktor	46.3	53.2	40.9	18	17	4	103	308	102	81	41.9	**61.9**
	Tracktor+MCTA	**48.9**	**57.3**	43.3	18	17	4	**103**	**308**	**98**	**81**	**42.4**	61.6
7	KSP	38.6	27.1	67.0	**22**	3	**0**	4488	171	72	28	-61.1	65.1
	KSP+ptrack	41.7	30.3	66.8	19	**3**	3	3791	253	49	21	-39.4	64.9
	Tracktor	68.7	69.4	68.0	17	6	2	55	66	36	18	72.9	**80.1**
	Tracktor+MCTA	**74.5**	**78.9**	**73.7**	17	6	2	**55**	**66**	**24**	**18**	**75.0**	80.0

Figure 6.8: Sample MCTA results on WILDTRACK datasets. The red arrows indicate the same person with the same identities across multiple cameras, and the colors of the bounding boxes also represents unique identities.

6.3.4 Computational Complexity Analysis

To achieve real-time performance, we down-sample the 30 fps synchronized input videos of size 1920×1080 to 10 fps and size 1080×720 without hurting overall tracking performance (shown in Table 6.6). We process input videos using a 4-seconds batches of 40 frames and assign each camera to a single RTX-2080 GPU for detection and tracking (see Fig. 6.1). After independently generating single-camera tracks, MCTA generates cross-camera tracks in the CPU using Alg. 7.

The execution time of the proposed MCTA algorithm depends on the average length of the overlapping tracklet segments in each camera pair. The algorithm can be executed in real-time in the CLASP1 dataset, which contains fewer and shorter tracklets. In CLASP2, it can run at approximately 12 fps. However, the current implementation described in section 6.3.2.2 uses the entire life-span of a tracklet to compute the Fréchet or Hausdorff association distance in the MCTA algorithm. Hence, the offline-MCTA in Table 6.5 shows 3 fps processing speed for the pre-computed SCT tracklets in a camera pair where the total number tracklets for Hungarian matching are $\mathcal{T}_p = 56$, $\mathcal{T}_a = 52$ and the corresponding total unique video frames are $12,534$ and $11,185$ respectively. It is possible to substantially reduce computation time by

Table 6.5: Computation complexity of offline-MCTA

Dist. Metric	Max. Track Size	Infer. Time (secs)	Memory (MB)
Hausdorff	–	94	18.3
	240	16	18.3
Fréchet	–	3995	18.3
	240	3779	18.3

Table 6.6: Computation time of the proposed online-MCTA.

Data.	Dist. Metric	Max. Size	Infer. Time (ms)	Memory (MB)	MOTA
CLASP1	Hausdorff	–	0.52	4.5	94.5
		240	0.50	4.5	94.5
	Fréchet	–	25.6	9.1	94.6
		240	8.20	4.3	94.6
CLASP2	Hausdorff	–	0.64	16.3	78.4
		240	0.61	16.3	78.4
	Fréchet	–	83.3	22.7	78.5
		240	19.6	16.3	78.7

limiting the length of single-camera tracklets compared by the algorithm. Table 6.6 shows that if we limit the size of the tracklets to 240 frames (or eight seconds), it is possible to achieve real-time performance for both datasets without degrading the accuracy of the algorithm.

In Table 6.7, we report the evaluation results of a two camera system where the real-time MCTA integrated with other detector variants such as FRCNN. Here, we observe real-time tracking by assigning one camera to each GPU and resizing input images to 1080×720 with frame rate 10 fps. The results of our multi-camera systems shown in Fig. 6.1 can track any number of cameras in real-time. However, the bottleneck is the multi-stage detector, which can be replaced with a lighter backbone or a single-stage real-time detector, such as [54].

Table 6.7: Computation time of the proposed tracking-by-detection framework where we employ an online-MCTA with FRCNN [1] detector.

#GPU	Model	Infer. Time (fps)	Peak Memory (MiB)
1	Detector	10	8006
	SCT	36	864
	MCTA	320	9
2	Detector	19.8	6827
	SCT	51	861
	MCTA	333	9

6.4 Conclusion

In this chapter, we propose an unsupervised MCTA method that only requires the homographies among neighboring cameras. We developed and applied both offline and online MCTA in overhead and lateral views. Our offline MCTA uses the pre-computed single-camera tracks to generate and associate all the overlapping tracklets across the cameras in a single run. However, to apply the MCTA in a real multi-camera system, we extend it to an online version by processing a four-second batch at a time. A graph optimization method performs the connection between the current and previous multi-camera batch results. Our experiments show that the proposed framework can accurately detect and track multiple objects across camera views in airport checkpoint scenarios and campus environments. Our MCTA framework is flexible and scalable, i.e., we can include any number of cameras based on the available computing resources. Since our MCTA employs SSL detectors and a pre-trained tracker to generate single-camera tracklets, it requires no training data, and its performance can be further improved by incorporating multi-view SSL-based appearance models [108].

Our framework can also be extended for a continuous learning scheme, where multi-view pseudo-labels are generated from the global tracks in the first batch run.

Since the single-camera tracker already leverages a triplet loss-based Re-ID, we can update the positive (same camera track state) and negative (different camera track state) examples to update the Re-ID model and use the SCT embeddings to perform appearance matching in cross-camera association. Thus, the same Re-ID model can be used for single and multi-camera associations without additional computation overhead.

CHAPTER 7

CONCLUSION & FUTURE WORK

In this chapter, we summarize the contributions and give an overview of the findings from all the objectives described in chapter 1 as well as possible directions for future work.

Objective 1: Test-time Data Augmentation for Unfamiliar Camera Perspectives

We have proposed a test-time data augmentation approach to enhance the detection rate in overhead camera views where state-of-the-art pre-trained models struggle due to the unfamiliar perspective. From the extensive experiments on the multiple version of the CLASP datasets, we found that the proposed test-time augmentation technique improves passenger and baggage detection in overhead views, and can discard low score predictions, mostly corresponding to false positives. This approach can be useful for generating high-quality predictions from unseen data where manual labels and retraining the pre-trained model for every unseen test perspective are expensive. Despite its satisfactory performance, our algorithm has two main limitations. First, we only consider the rotation augmentation, since the dataset requires the rotation invariance property, i.e., the appearance remains recognizable even when we rotate images at multiple random angles. We can address this challenge by incorporating data-specific augmentation such as motion-blur, color, synthetic objects, and so on. Second, the computational complexity increases linearly as we call a pre-trained model for each augmented version of the input frames. Even though distributing the computations across processing units can address this bottleneck to apply the proposed technique during inference, the findings from this objective clearly indicate that the pre-trained CNN model should learn about rotation invariance to allow the model to be deployed in real-world surveillance applications. We have ad-

dressed the pre-trained model uncertainty problem in object detection and panoptic segmentation downstream tasks along this direction. In the future, we intend to further improve the performance of our test-time augmentation-based detection algorithm by employing dynamic mechanisms for choosing the most informative rotation angles.

Objective 2: Self-supervised Learning for Object Detection

We developed an SSL framework to address the limitations of multiple-object detection algorithms in an unfamiliar perspective when the availability of manual training labels is limited. Our SSL approach can generate high-quality pseudo-labels and incorporate the estimated uncertainty into the pre-trained model to iteratively update the model weights for unseen perspectives. Our experimental evaluation shows that a pre-trained model with high uncertainty on unseen multi-camera perspectives converges after a few iterations and successfully detects the classes of interest in multi-camera views of complex real-world scenarios. Our SSL framework is flexible and scalable, i.e, we can apply it for most downstream tasks learning in machine vision applications, since it requires no training data, works without computational overhead during inference, and can include any number of cameras in the network. Our framework can also be extended for semi-supervised learning using some manual labels if available, and can further improve performance compared to fully self-supervised models.

In the currently proposed SSL method for object detection and instance segmentation, we generate pseudo-labels for all unseen data, which is an inefficient approach as we need to process significant amounts of redundant information during SSL training. Instead, we could sample only the most complex samples where the model needs to improve the image understanding ability of the pre-trained model. However, unlike image region ranking in semantic segmentation [103] for retraining, for detection or instance segmentation models, it is difficult to establish image-level rank-

ing criteria to consider for SSL retraining. To address these challenges, we may use image-level out-of-distribution (OOD) predictor results (embeddings and free energy scores) to identify the similarity between the closed set (data used in the pre-trained model) and open set (unseen data). Furthermore, we can sample hard and easy examples based on similarity scores, instead of random sampling or using large unseen datasets.

Objective 3: Self-supervised Learning for Panoptic Segmentation

In Chapter 4, we introduced a self-supervised learning technique to accurately segment multiple fruit flower species without significant manual labeling efforts. Similar to our SSL technique for object detection, we automatically generate panoptic pseudo-labels by leveraging data augmentation during test time. However, we extend the SSL approach by including a semantic segmentation refinement strategy that produces accurate panoptic pseudo-labels for iterative training.

The proposed SSL framework for multi-species flower segmentation can be extended to design generalized deep learning tools to provide real-time insights into the growth and health of the fruits/flowers by analyzing the agricultural visual data, so that farmers can make more informed decisions, improve the quality of the fruits/flowers, and increase their yield, even though the sensor data drifts significantly across different species, locations, weather condition, and harvesting seasons. Machine vision algorithms specifically fail to detect dense and cluttered flowers/fruits for multiple-species orchard fields when the training and testing data are significantly different. Since agricultural field data distributions drift with time, location, and weather conditions, the machine learning model deployment must be robust to data drift challenges. To address these challenges, we can extend our multi-species flower detection approach as a counting model that can also employ a representative sampling technique for new information, so as to continuously update the predictive model when data drift takes place in different locations or under various weather conditions.

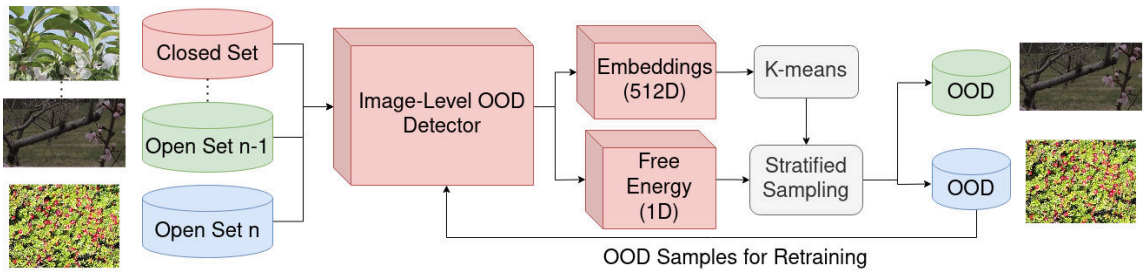


Figure 7.1: Continuous learning framework using OOD detections. A closed-set classification model generates the free energy scores and the latent embeddings for each input from unseen open-set datasets. The embeddings are used to compute the distribution of the k-means distances from the closed set clusters. Stratified sampling leverages the weighted free energy scores to select representative OOD frames for retraining the closed-set classification model.

As shown in Fig. 7.1, a continuous learning framework can be designed based on state-of-the-art object detection [7, 9] and recognition architectures [32, 250]. We can also incorporate an OOD classifier [32] to intelligently sample outliers from new datasets and employ unsupervised clustering methods [34] to enhance the robustness of outlier predictions of the OOD model.

The proposed research can significantly benefit agricultural automation and robotics by estimating the blooming intensity of fruit flowers or counting the density of multiple fruit species. Our SSL learning framework with representative sampling can help farmers to assess the yield of the upcoming harvest and plan their harvesting schedule accordingly. Moreover, by monitoring the number of fruits/flowers on a tree or a branch, farmers can identify areas of an orchard that need more attention in terms of fertilization, watering, thinning, and pruning. Also, the proposed models can be trained to recognize the patterns of diseases in the leaves or fruits and alert farmers to take action before the infection spreads further. By analyzing the data from the sensors, the models can provide insights into the best conditions for the growth of the fruits and alert farmers to take action if any of the factors are not optimal. Overall, Our proposed models can be a valuable tool for optimizing fruit

production in agriculture and enhancing safety of deploying SSL methods without using a significant amount of field data to pre-train the initial models.

Objective 4: Unsupervised Learning for Multiple Object Tracking and Segmentation

Downstream tasks such as object detection, segmentation, motion prediction, and appearance similarity assessment are widely used computer vision tools for segmenting and tracking multiple objects simultaneously. In Chapter 5, we proposed an unsupervised multi-task learning method that leverages task-dependent uncertainties to learn the contribution of shape or appearance and location features from multiple-objects video datasets. We incorporate the temporal relationships among sequential multiple-object embeddings as a simple constraint in the clustering algorithm, which can accommodate various object tracking challenges such as detection mistakes and entry into or exit from the scenes. Our experimental results show that our approach outperforms several state-of-the-art MOTS algorithms. However, even though our proposed technique can cluster multiple object embeddings, it sometimes fails to associate cluster heads with detections in subsequent frames due to significant motion and severe occlusions. Therefore, to address these issues, we can extend our algorithm to include motion prediction as an additional task.

In the future, our goal is to use more robust entry/exit/occlusion detection techniques to increase the accuracy of data association mechanism to the problems of multiple-object tracking as well as video instance segmentation [26]. We can also incorporate our SSL framework for the multi-task feature extractor to deploy the proposed technique in unseen data. Our unsupervised approach can also be extended to a self-supervised method to capture abrupt object appearance variations in crowded scenarios. Furthermore, we can train a self-supervised triplet loss model by automatically mining the clusters of our jointly learned embeddings.

Objective 5: Unsupervised Multi-camera Tracklet Association

Finally, we propose an unsupervised MCTA method that employs our SSL detector models in a single-camera tracker and uses 2D homographies among neighboring cameras to perform cross-camera associations. We demonstrate the effectiveness of our MCTA algorithm by evaluating the performance in both fully (lateral camera views) and partially overlapping (overhead camera views) camera networks. Our experiments show that the proposed framework can accurately detect multiple objects across camera views in airport checkpoint and campus environment scenarios. As our method requires no training data nor camera calibration, it can be applied to any new camera system and scaled for any number of cameras. We implement and experiment with both offline and online versions of MCTA to verify our algorithm's flexibility.

In the future, our goal is to improve the 2D homographies using deep coarse feature-based correspondences [168], as we have observed significant projection errors with the use of a simple RANSAC-based approach. Although we avoid multi-view appearance modeling due to the challenging overhead perspective and to achieve real-time inference, we can also include an appearance model using perspective proof detection [251] and incorporate triplet loss-based cross-camera feature learning [108, 107] to enhance cross-camera association. To address the impact of perspective changes in multi-camera appearances, we can project image patches from one camera to another to compute appearance similarity, or apply a learning technique to estimate appearance similarity between two cameras at different planes. Finally, we can employ the idea of SSL for MCTA feature learning. We can efficiently sample positive and negative examples from multi-camera global tracks for a video segment. Whereas the easy positive examples would come from the same camera, the complex examples would come from other cameras. Then we can train a triplet loss-based deep neural network model to generate embeddings that will be used with motion features in our current algorithm.

REFERENCES

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[2] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[3] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *International Conference on Learning Representations*, 2018.

[4] M. Vo, E. Yumer, K. Sunkavalli, S. Hadap, Y. Sheikh, and S. G. Narasimhan, "Self-supervised multi-view person association and its applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 8, pp. 2794–2808, 2021.

[5] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.

[7] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.

[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[9] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6392–6401.

[10] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 7482–7491.

[11] T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, T. Bagautdinov, L. Lettry, P. Fua, L. Van Gool, and F. Fleuret, "Wildtrack: A multi-camera HD

dataset for dense unscripted pedestrian detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5030–5039.

[12] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "Mots: Multi-object tracking and segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7934–7943.

[13] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 4037–4058, 2021.

[14] M. Xu, Z. Zhang, H. Hu, J. Wang, L. Wang, F. Wei, X. Bai, and Z. Liu, "End-to-end semi-supervised object detection with soft teacher," in *IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3040–3049.

[15] S. Belharbi, I. Ben Ayed, L. McCaffrey, and E. Granger, "Deep active learning for joint classification & segmentation with weak annotator," in *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.

[16] A. Siddique, A. Tabb, and H. Medeiros, "Self-supervised learning for panoptic segmentation of multiple fruit flower species," *IEEE Robotics and Automation Letters*, pp. 1–8, 2022.

[17] A. Siddique, R. J. Mozhdehi, and H. Medeiros, "Unsupervised spatio-temporal latent feature clustering for multiple-object tracking and segmentation," in *British Machine Vision Conference*, 2021.

[18] A. Siddique and H. Medeiros, "Tracking passengers and baggage items using multiple overhead cameras at security checkpoints," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–13, 2022.

[19] T. Simon, H. Joo, I. A. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 4645–4653.

[20] M. Cai, M. Luo, X. Zhong, and H. Chen, "Uncertainty-aware model adaptation for unsupervised cross-domain object detection," *arXiv preprint arXiv:2108.12612*, 2021.

[21] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *European Conference on Computer Vision*, 2018, pp. 139–156.

[22] Y. Wang, J. Zhang, M. Kan, S. Shan, and X. Chen, "Self-supervised equivariant attention mechanism for weakly supervised semantic segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 272–12 281.

[23] P. Bergmann, T. Meinhardt, and L. Leal-Taixé, "Tracking without bells and whistles," in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 941–951.

[24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*, 2014, pp. 740–755.

[25] P. A. Dias, A. Tabb, and H. Medeiros, "Multispecies fruit flower detection using a refined semantic segmentation network," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3003–3010, 2018.

[26] L. Porzi, M. Hofinger, I. Ruiz, J. Serrat, S. R. Bulò, and P. Kontschieder, "Learning multi-object tracking and segmentation from automatic annotations," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6845–6854.

[27] J. Luiten, T. Fischer, and B. Leibe, "Track to reconstruct and reconstruct to track," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1803–1810, 2020.

[28] Y. Xu, X. Liu, Y. Liu, and S.-C. Zhu, "Multi-view people tracking via hierarchical trajectory composition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4256–4265.

[29] R. C. González and R. E. Woods, "Digital image processing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, pp. 242–243, 1981.

[30] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[31] T. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[32] X. Du, Z. Wang, M. Cai, and Y. Li, "Vos: Learning what you don't know by virtual outlier synthesis," *International Conference on Learning Representations*, 2022.

[33] R. Lopez, J. Regier, M. I. Jordan, and N. Yosef, "Information constraints on auto-encoding variational bayes," *Advances in neural information processing systems*, vol. 31, 2018.

[34] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07, 2007, pp. 1027–1035.

[35] L. McInnes, J. Healy, N. Saul, and L. Grossberger, "Umap: Uniform manifold approximation and projection," *The Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.

[36] S. M. Khan and M. Shah, "A multiview approach to tracking people in crowded scenes using a planar homography constraint," in *European Conference on Computer Vision*, 2006, pp. 133–146.

[37] P. Baqué, F. Fleuret, and P. Fua, "Deep occlusion reasoning for multi-camera multi-target detection," in *IEEE International Conference on Computer Vision*, 2017, pp. 271–279.

[38] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*. Cham: Springer International Publishing, 2020, pp. 213–229.

[39] W. Lee, J. Na, and G. Kim, "Multi-task self-supervised object detection via recycling of bounding box annotations," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4979–4988.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[41] Z. He, J. Li, D. Liu, H. He, and D. Barber, "Tracking by animation: Unsupervised learning of multi-object attentive trackers," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1318–1327.

[42] S. Herdade, A. Kappeler, K. Boakye, and J. Soares, "Image captioning: Transforming objects into words," *Advances in neural information processing systems*, vol. 32, 2019.

[43] L. Hoyer, D. Dai, Y. Chen, A. Köring, S. Saha, and L. Van Gool, "Three ways to improve semantic segmentation with self-supervised depth estimation,"

in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 130–11 140.

[44] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *arXiv e-prints*, vol. abs/1812.11941, 2018.

[45] D. Wang and S. Zhang, "Contextual instance decoupling for robust multi-person pose estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2022, pp. 11 060–11 068.

[46] J. Sock, S. H. Kasaei, L. S. Lopes, and T.-K. Kim, "Multi-view 6d object pose estimation and camera motion planning using rgbd images," in *2017 IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2228–2235.

[47] Y. Du, Z. Chen, C. Jia, X. Yin, T. Zheng, C. Li, Y. Du, and Y.-G. Jiang, "Svtr: Scene text recognition with a single visual model," in *International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 7 2022, pp. 884–890.

[48] H. Wang, X. Bai, M. Yang, S. Zhu, J. Wang, and W. Liu, "Scene text retrieval via joint text detection and similarity learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2021, pp. 4558–4567.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

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

[52] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[53] X. Chen, C.-J. Hsieh, and B. Gong, "When vision transformers outperform resnets without pretraining or strong data augmentations," *arXiv preprint arXiv:2106.01548*, 2021.

[54] Y. Lee, C. Lee, H.-J. Lee, and J.-S. Kim, "Fast detection of objects using a yolov3 network for a vending machine," in *IEEE International Conference on Artificial Intelligence Circuits and Systems*, 2019, pp. 132–136.

[55] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2018.

[56] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[57] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[58] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[59] 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," *arXiv preprint arXiv:1704.04861*, 2017.

[60] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, 2019, pp. 6105–6114.

[61] G. Sidorov, A. Gelbukh, H. Gómez-Adorno, and D. Pinto, "Soft similarity and soft cosine measure: Similarity of features in vector space model," *Computacion y Sistemas*, vol. 18, no. 3, pp. 491–504, Jul. 2014.

[62] A. A. Taha and A. Hanbury, "An efficient algorithm for calculating the exact hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2153–2163, Nov 2015.

[63] P. C. Besse, B. Guillouet, J.-M. Loubes, and F. Royer, "Review and perspective for distance-based clustering of vehicle trajectories," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3306–3317, 2016.

[64] K. Buchin, M. Buchin, and C. Wenk, "Computing the Fréchet distance between simple polygons," *Computational Geometry*, vol. 41, no. 1, pp. 2–20, 2008.

[65] T. Eiter and H. Mannila, "Computing discrete fréchet distance," *Technische Universität Wien, Vienna*, 1994.

[66] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, "A high-bias, low-variance introduction to machine learning for physicists," *Physics Reports*, vol. 810, pp. 1–124, 2019.

[67] H. Yao, D.-l. Zhu, B. Jiang, and P. Yu, "Negative log likelihood ratio loss for deep neural network classification," in *Future Technologies Conference 2019: Volume 1*. Springer, 2020, pp. 276–282.

[68] D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *British Machine Vision Conference*. BMVA Press, September 2016, pp. 119.1–119.11.

[69] P. Dendorfer, A. Ošep, A. Milan, K. Schindler, D. Cremers, I. Reid, S. Roth, and L. Leal-Taixé, "MOTChallenge: A benchmark for single-camera multiple target tracking," *Int. J. of Computer Vision*, vol. 129, pp. 845–881, 2020.

[70] A. Geiger, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[71] Z. Xu, W. Zhang, X. Tan, W. Yang, H. Huang, S. Wen, E. Ding, and L. Huang, "Segment as points for efficient online multi-object tracking and segmentation," in *European Conference on Computer Vision*, 2020.

[72] Z. Xu, W. Yang, W. Zhang, X. Tan, H. Huang, and L. Huang, "Segment as points for efficient and effective online multi-object tracking and segmentation," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 01, pp. 1–1, 2021.

[73] R. Stiefelhagen, K. Bernardin, R. Bowers, J. Garofolo, D. Mostefa, and P. Soundararajan, "The CLEAR 2006 evaluation," in *CLEAR*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 1–44.

[74] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[75] A. Newell and J. Deng, "How useful is self-supervised pretraining for visual tasks?" in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2020, pp. 7343–7352.

[76] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in

IEEE International Conference on Robotics and Automation, 2018, pp. 1134–1141.

[77] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *IEEE Workshops on Applications of Computer Vision*, vol. 1, 2005, pp. 29–36.

[78] A. Chowdhury, D. Chaudhari, S. Chaudhuri, and C. Jermaine, "Meta-meta classification for one-shot learning," in *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 1628–1637.

[79] C. Yang, L. Huang, and E. J. Crowley, "Plug and play active learning for object detection," *arXiv preprint arXiv:2211.11612*, 2022.

[80] S. Vaze, K. Han, A. Vedaldi, and A. Zisserman, "Open-set recognition: a good closed-set classifier is all you need?" in *International Conference on Learning Representations*, 2022.

[81] V. Kaushal, R. Iyer, S. Kothawade, R. Mahadev, K. Doctor, and G. Ramakrishnan, "Learning from less data: A unified data subset selection and active learning framework for computer vision," in *IEEE Winter Conference on Applications of Computer Vision*, 2019, pp. 1289–1299.

[82] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv:2304.02643*, 2023.

[83] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," in *Neural Information Processing Systems*, 2020.

[84] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *Journal of Big Data*, vol. 5, no. 1, p. 42, Nov 2018.

[85] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 12 104–12 114.

[86] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, p. 60, 2019.

[87] C. Luo, Y. Zhu, L. Jin, and Y. Wang, "Learn to augment: Joint data augmentation and network optimization for text recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 743–13 752.

[88] G. Wang, W. Li, M. Aertsen, J. Deprest, S. Ourselin, and T. Vercauteren, "Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks," *Neurocomputing*, vol. 338, pp. 34–45, 2019.

[89] S. Gupta, J. Hoffman, and J. Malik, "Cross modal distillation for supervision transfer," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2827–2836.

[90] P. Goyal, M. Caron, B. Lefaudeux, M. Xu, P. Wang, V. Pai, M. Singh, V. Liptchinsky, I. Misra, A. Joulin, and P. Bojanowski, "Self-supervised pretraining of visual features in the wild," *arXiv preprint arXiv:2103.01988*, 2021.

[91] I. Radosavovic, P. Dollár, R. Girshick, G. Gkioxari, and K. He, "Data distillation: Towards omni-supervised learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4119–4128.

[92] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Computational Learning Theory*, ser. COLT' 98, New York, NY, USA, 1998, pp. 92–100.

[93] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, pp. 1171–1179.

[94] P. A. Dias, H. Medeiros, and F. Odone, "Fine segmentation for Activity of Daily Living analysis in a wide-angle multi-camera," in *5th AMMDS in conjunction with The British Machine Vision Conference*, 2017.

[95] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *IEEE International Conference on Computer Vision*, dec 2015, pp. 1422–1430.

[96] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., 2014.

[97] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[98] O. J. Hénaff, A. Srinivas, J. De Fauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. Van Den Oord, "Data-efficient image recognition with contrastive predictive coding," in *International Conference on Machine Learning*, ser. ICML'20, 2020.

[99] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *European conference on computer vision*. Cham: Springer International Publishing, 2020, pp. 776–794.

[100] M. Taj and A. Cavallaro, "Multi-camera track-before-detect," in *International Conference on Distributed Smart Cameras*, 2009.

[101] P. Goyal, Q. Duval, J. Reizenstein, M. Leavitt, M. Xu, B. Lefaudeux, M. Singh, V. Reis, M. Caron, P. Bojanowski, A. Joulin, and I. Misra, "Vissl," 2021.

[102] C.-I. Lai, "Contrastive predictive coding based feature for automatic speaker verification," *arXiv preprint arXiv:1904.01575*, 2019.

[103] S. A. Golestaneh and K. Kitani, "Importance of self-consistency in active learning for semantic segmentation," in *British Machine Vision Conference*, 2020.

[104] Y. Wang, J. Peng, and Z. Zhang, "Uncertainty-aware pseudo label refinery for domain adaptive semantic segmentation," in *IEEE/CVF International Conference on Computer Vision*, October 2021, pp. 9092–9101.

[105] Q. Li, A. Arnab, and P. H. Torr, "Weakly- and semi-supervised panoptic segmentation," in *European Conference on Computer Vision*, September 2018.

[106] N. Peri, P. Khorramshahi, S. S. Rambhatla, V. Shenoy, S. Rawat, J.-C. Chen, and R. Chellappa, "Towards real-time systems for vehicle re-identification, multi-camera tracking, and anomaly detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 2648–2657.

[107] X. Sun, M. Cheng, C. Min, and L. Jing, "Self-supervised deep multi-view subspace clustering," in *Asian Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 101, 17–19 Nov 2019, pp. 1001–1016.

[108] C.-H. Ho, B. Liu, T.-Y. Wu, and N. Vasconcelos, "Exploit clues from views: Self-supervised and regularized learning for multiview object recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020.

[109] P. Sermanet, C. Lynch, J. Hsu, and S. Levine, "Time-contrastive networks: Self-supervised learning from multi-view observation," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 486–487.

[110] Y. Gan, R. Han, L. Yin, W. Feng, and S. Wang, "Self-supervised multi-view multi-human association and tracking," in *ACM International Conference on Multimedia*, 2021, pp. 282–290.

[111] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1386–1383.

[112] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman, "Temporal cycle-consistency learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1801–1810.

[113] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[114] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb 2015.

[115] D. Dwibedi, J. Tompson, C. Lynch, and P. Sermanet, "Learning actionable representations from visual observations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 1577–1584.

[116] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[117] S. Tonekaboni, D. Eytan, and A. Goldenberg, "Unsupervised representation learning for time series with temporal neighborhood coding," in *International Conference on Learning Representations*, 2020.

[118] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Byol for audio: Self-supervised learning for general-purpose audio representation," in *IEEE International Joint Conference on Neural Networks*, Jul 2021.

[119] J. Mao, Q. Yu, Y. Yamakata, and K. Aizawa, "Noisy annotation refinement for object detection," in *British Machine Vision Conference*, 2021.

[120] B. Cheng, G. Liu, J. Wang, Z. Huang, and S. Yan, "Multi-task low-rank affinity pursuit for image segmentation," in *IEEE International Conference on Computer Vision*, 2011, pp. 2439–2446.

[121] R. Mohan and A. Valada, "EfficientPS: Efficient panoptic segmentation," *International Journal of Computer Vision*, vol. 129, no. 5, pp. 1551–1579, 2021.

[122] Z. Li, W. Wang, E. Xie, Z. Yu, A. Anandkumar, J. M. Alvarez, P. Luo, and T. Lu, "Panoptic segformer: Delving deeper into panoptic segmentation with transformers," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1270–1279.

[123] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited," in *IEEE International Conference on Computer Vision*, 2015.

[124] R. Jalil Mozhdehi, Y. Reznichenko, A. Siddique, and H. Medeiros, "Deep convolutional particle filter with adaptive correlation maps for visual tracking," in *IEEE International Conference on Image Processing*, 2018.

[125] R. J. Mozhdehi, Y. Reznichenko, A. Siddique, and H. Medeiros, "Convolutional adaptive particle filter with multiple models for visual tracking," in *International Symposium on Visual Computing*, 2018.

[126] H. W. Kuhn and B. Yaw, "The hungarian method for the assignment problem," *Naval Res. Logist. Quart*, pp. 83–97, 1955.

[127] R. Henschel, L. Leal-Taixé, D. Cremers, and B. Rosenhahn, "A novel multi-detector fusion framework for multi-object tracking," *arXiv preprint arXiv:1705.08314*, pp. 1–10, 2017.

[128] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *IEEE International Conference on Computer Vision*, Oct 2017, pp. 300–311.

[129] Y. min Song and M. Jeon, "Online multi-object tracking and segmentation with GMPHD filter and simple affinity fusion," *arXiv preprint arXiv:2009.00100*, 2020.

[130] S. Qiao, Y. Zhu, H. Adam, A. Yuille, and L.-C. Chen, "Vip-deeplab: Learning visual perception with depth-aware video panoptic segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3996–4007.

[131] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *International Conference on Machine Learning*, 2010, pp. 663–670.

[132] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[133] R. Vidal, "Subspace clustering," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2011.

[134] M. Bäuml, M. Tapaswi, and R. Stiefelhagen, "Semi-supervised learning with constraints for person identification in multimedia data," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3602–3609.

[135] P. J.Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," in *Computational and Applied Mathematics*, vol. 20, November 1987, pp. 53–65.

[136] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," in *19th International Conference on Neural Information Processing Systems*, 2006, pp. 1137–1144.

[137] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, Aug 1995.

[138] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International Conference on Machine Learning*, 2016, pp. 478–487.

[139] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, "Deep subspace clustering networks," in *31st International Conference on Neural Information Processing Systems*, 2017, pp. 23–32.

[140] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *European Conference on Computer Vision*, 2018, pp. 139–156.

[141] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "Clustergan: Latent space clustering in generative adversarial networks," in *AAAI Conference on Artificial Intelligence*, 2019, pp. 4610–4617.

[142] P. Zhou, Y. Hou, and J. Feng, "Deep adversarial subspace clustering," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018,

pp. 1596–1604.

[143] T. Zhang, P. Ji, M. Harandi, R. Hartley, and I. Reid, "Scalable deep k-subspace clustering," in *Asian Conference on Computer Vision*, 2018, pp. 466–481.

[144] Y. Reznichenko, E. Prampolini, A. Siddique, H. Medeiros, and F. Odone, "Visual tracking with autoencoder-based maximum a posteriori data fusion," in *The IEEE Computer Society Computers, Software, and Applications Conference*, 2019, pp. 501–506.

[145] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained k-means clustering with background knowledge." in *International Conference on Machine Learning*. Morgan Kaufmann, 2001, pp. 577–584.

[146] S. Tierney, J. Gao, and Y. Guo, "Subspace clustering for sequential data," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1019–1026.

[147] M. Tapaswi, M. Law, and S. Fidler, "Video face clustering with unknown number of clusters," in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5026–5035.

[148] P. Kulshreshtha and T. Guha, "An online algorithm for constrained face clustering in videos," in *IEEE International Conference on Image Processing*, 2018, pp. 2670–2674.

[149] R. Jalil Mozhdehi and H. Medeiros, "Deep convolutional likelihood particle filter for visual tracking," in *The International Conference on Image Processing, Computer Vision, and Pattern Recognition*, 2020, pp. 27–38.

[150] N. Anjum and A. Cavallaro, "Trajectory association and fusion across partially overlapping cameras," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2009.

[151] K. Nithin and F. Bremond, "Multi-camera tracklet association and fusion using ensemble of visual and geometric cues," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 3, pp. 431–440, 2017.

[152] S. Bae and K. Yoon, "Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1218–1225.

[153] A. S. Hassanein, M. E. Hussein, and W. Gomaa, "Semantic analysis of crowded

scenes based on non-parametric tracklet clustering," in *International Joint Conference on Artificial Intelligence*, 2016.

[154] P. Li, J. Zhang, Z. Zhu, Y. Li, L. Jiang, and G. Huang, "State-aware re-identification feature for multi-target multi-camera tracking," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 1506–1516.

[155] J. Si, H. Zhang, C.-G. Li, and J. Guo, "Spatial pyramid-based statistical features for person re-identification: A comprehensive evaluation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 7, pp. 1140–1154, 2018.

[156] H.-M. Hsu, T.-W. Huang, G. Wang, J. Cai, Z. Lei, and J.-N. Hwang, "Multi-camera tracking of vehicles based on deep features re-id and trajectory-based camera link models," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[157] A. Zheng, X. Zhang, B. Jiang, B. Luo, and C. Li, "A subspace learning approach to multishot person reidentification," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 149–158, 2020.

[158] H. Germain, V. Lepetit, and G. Bourmaud, "Neural reprojection error: Merging feature learning and camera pose estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 414–423.

[159] J. Zhang, C. Wang, S. Liu, L. Jia, N. Ye, J. Wang, J. Zhou, and J. Sun, "Content-aware unsupervised deep homography estimation," in *European conference on computer vision*, 2020, pp. 653–669.

[160] Y. Bok, D.-G. Choi, P. Vasseur, and I. S. Kweon, "Extrinsic calibration of non-overlapping camera-laser system using structured environment," in *IEEE/RSJ International Conference on Intelligence Robots and Systems*, 2014, pp. 436–443.

[161] H. Medeiros, H. Iwaki, and J. Park, "Online distributed calibration of a large network of wireless cameras using dynamic clustering," in *International Conference on Distributed Smart Cameras*, 2008.

[162] Q. You and H. Jiang, "Real-time 3d deep multi-camera tracking," *arXiv preprint arXiv:2003.11753*, 2020.

[163] P. Khorramshahi, V. Shenoy, M. Pack, and R. Chellappa, "Scalable and real-time multi-camera vehicle detection, re-identification, and tracking," *arXiv preprint arXiv:2204.07442*, 2022.

[164] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[165] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *IEEE International Conference on Image Processing*, 2017, pp. 3645–3649.

[166] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *European conference on computer vision*, 2018, pp. 734–750.

[167] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[168] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "Loftr: Detector-free local feature matching with transformers," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8918–8927.

[169] Z. Wu and R. J. Radke, "Real-time airport security checkpoint surveillance using a camera network," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, June 2011, pp. 25–32.

[170] A. Islam, Y. Zhang, D. Yin, O. Camps, and R. J. Radke, "Correlating belongings with passengers in a simulated airport security checkpoint," in *International Conference on Distributed Smart Cameras*, 2018.

[171] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," in *European Conference on Computer Vision*, 2016.

[172] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017, pp. 5168–5177.

[173] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[174] R. Mazzon and A. Cavallaro, "Multi-camera tracking using a multi-goal social force model," *Neurocomputing*, vol. 100, pp. 41 – 50, 2013.

[175] S. Zhang, Y. Zhu, and A. Roy-Chowdhury, "Tracking multiple interacting targets in a camera network," *Computer Vision and Image Understanding*, vol. 134, pp. 64 – 73, 2015.

[176] K. Hong, H. Medeiros, P. J. Shin, and J. Park, "Resource-aware distributed particle filtering for cluster-based object tracking in wireless camera networks," *Int. J. of Sensor Networks*, vol. 21, no. 3, pp. 137–156, 2016.

[177] H. Medeiros, J. Park, and A. Kak, "Distributed object tracking using a cluster-based kalman filter in wireless camera networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 4, pp. 448–463, 2008.

[178] N. Wang, Y. Song, C. Ma, W. Zhou, W. Liu, and H. Li, "Unsupervised deep tracking," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1308–1317.

[179] R. Jalil Mozhdehi and H. Medeiros, "Deep convolutional correlation iterative particle filter for visual tracking," *Computer Vision and Image Understanding*, pp. 103–479, 2022.

[180] M. Chuang, J. Hwang, J. Ye, S. Huang, and K. Williams, "Underwater fish tracking for moving cameras based on deformable multiple kernels," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 9, pp. 2467–2477, 2017.

[181] R. Ding, M. Yu, H. Oh, and W. Chen, "New multiple-target tracking strategy using domain knowledge and optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 4, pp. 605–616, 2017.

[182] R. Henschel, Y. Zou, and B. Rosenhahn, "Multiple people tracking using body and joint detections," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 770–779.

[183] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "Sophie: An attentive GAN for predicting paths compliant to social and physical constraints," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1349–1358.

[184] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 961–971.

[185] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

[186] M. Babaee, A. Athar, and G. Rigoll, "Multiple people tracking using hierarchical deep tracklet re-identification," *arXiv preprint arXiv:1811.04091*, 2018.

[187] M. Keuper, S. Tang, B. Andres, T. Brox, and B. Schiele, "Motion segmentation & multiple object tracking by correlation co-clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 1, pp. 140–153, 2020.

[188] K. Zhang, J. Chen, G. Yu, X. Zhang, and Z. Li, "Visual trajectory tracking of wheeled mobile robots with uncalibrated camera extrinsic parameters," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 7191–7200, 2021.

[189] Z. Liu, J. Zhang, and L. Liu, "Upright orientation of 3D shapes with convolutional networks," *Graphical Models*, vol. 85, pp. 22 – 29, 2016.

[190] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017, pp. 5987–5995.

[191] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016, pp. 770–778.

[192] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017, pp. 936–944.

[193] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics." *EURASIP J. on Image and Video Processing*, vol. 2008, 2008.

[194] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," in *European conference on computer vision*, 2016.

[195] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited," in *IEEE International Conference on Computer Vision*, 2015, pp. 4696–4704.

[196] G. Farjon, O. Krikeb, A. B. Hillel, and V. Alchanatis, "Detection and counting of flowers on apple trees for better chemical thinning decisions," *Precision Agriculture*, vol. 21, no. 3, pp. 503–521, 2020.

[197] K. Sun, X. Wang, S. Liu, and C. Liu, "Apple, peach, and pear flower detection using semantic segmentation network and shape constraint level set," *Computers and Electronics in Agriculture*, vol. 185, p. 106150, 2021.

[198] P. Akiva, K. Dana, P. Oudemans, and M. Mars, "Finding berries: Segmentation and counting of cranberries using point supervision and shape priors," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[199] U. Bhattarai and M. Karkee, "A weakly-supervised approach for flower/fruit counting in apple orchards," *Computers in Industry*, vol. 138, p. 103635, 2022.

[200] W. H. Maes and K. Steppe, "Perspectives for remote sensing with unmanned aerial vehicles in precision agriculture," *Trends in Plant Science*, vol. 24, no. 2, pp. 152–164, 2019.

[201] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *European Conference on Computer Vision*, 2018.

[202] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv Preprint*, vol. abs/2004.10934, 2020.

[203] I. H. Laradji, D. Vázquez, and M. Schmidt, "Where are the masks: Instance segmentation with image-level supervision," in *British Machine Vision Conference*, 2019.

[204] X. A. Wang, J. Tang, and M. Whitty, "Side-view apple flower mapping using edge-based fully convolutional networks for variable rate chemical thinning," *Computers and Electronics in Agriculture*, vol. 178, p. 105673, 2020.

[205] D. Wu, S. Lv, M. Jiang, and H. Song, "Using channel pruning-based YOLOv4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments," *Computers and Electronics in Agriculture*, vol. 178, p. 105742, 2020.

[206] G. Li, R. Suo, G. Zhao, C. Gao, L. Fu, F. Shi, J. Dhupia, R. Li, and Y. Cui, "Real-time detection of kiwifruit flower and bud simultaneously in orchard using YOLOv4 for robotic pollination," *Computers and Electronics in Agriculture*, vol. 193, p. 106641, 2022.

[207] A. Koirala, K. Walsh, Z. Wang, and C. McCarthy, "Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of 'mangoyolo'," *Precision Agriculture*, vol. 20, p. 1107–1135, 12 2019.

[208] D. Riehle, D. Reiser, and H. W. Griepentrog, "Robust index-based semantic plant/background segmentation for RGB- images," *Computers and Electronics in Agriculture*, vol. 169, p. 105201, 2020.

[209] S. Bargoti and J. Underwood, "Deep fruit detection in orchards," in *IEEE International Conference on Robotics and Automation*, 2017.

[210] H. Kang and C. Chen, "Fast implementation of real-time fruit detection in apple orchards using deep learning," *Computers and Electronics in Agriculture*, vol. 168, p. 105108, 2020.

[211] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint*, vol. abs/1712.04621, 2017.

[212] P. A. Dias and H. Medeiros, "Semantic segmentation refinement by monte carlo region growing of high confidence detections," in *Asian Conference on Computer Vision (ACCV)*, 2018, pp. 131–146.

[213] C. Tang, H. Chen, X. Li, J. Li, Z. Zhang, and X. Hu, "Look closer to segment better: Boundary patch refinement for instance segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[214] A. Kirillov, Y. Wu, K. He, and R. Girshick, "PointRend: Image segmentation as rendering," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[215] M. Teichmann and R. Cipolla, "Convolutional CRFs for semantic segmentation," in *British Machine Vision Conference*, 2020.

[216] K. Saito, D. Kim, S. Sclaroff, and K. Saenko, "Universal domain adaptation through self-supervision," in *Neural Information Processing Systems (NeurIPS)*, 2020.

[217] M. R. Vyas, H. Venkateswara, and S. Panchanathan, "Leveraging seen and unseen semantic relationships for generative zero-shot learning," in *European Conference on Computer Vision*, 2020.

[218] M. Larsson, E. Stenborg, C. Toft, L. Hammarstrand, T. Sattler, and F. Kahl, "Fine-grained segmentation networks: Self-supervised segmentation for improved long-term visual localization," in *IEEE/CVF International Conference on Computer Vision*, 2019.

[219] Z. Li, W. Wang, E. Xie, Z. Yu, A. Anandkumar, J. M. Alvarez, P. Luo, and T. Lu, "Panoptic SegFormer: Delving deeper into panoptic segmentation

with transformers," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[220] H. Caesar, J. Uijlings, and V. Ferrari, "COCO-stuff: Thing and stuff classes in context," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[221] J. Yuan, Y. Liu, C. Shen, Z. Wang, and H. Li, "A simple baseline for semi-supervised semantic segmentation with strong data augmentation," in *IEEE/CVF International Conference on Computer Vision*, 2021.

[222] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[223] M. Karkee, Q. Zhang, and A. Silwal, "Agricultural robots for precision agricultural tasks in tree fruit orchards," in *Innovation in Agricultural Robotics for Precision Agriculture*, 2021, pp. 63–89.

[224] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *IEEE International Conference on Robotics and Automation*, May 2016, pp. 512–519.

[225] Y. Guo, J. Gao, and F. Li, "Spatial subspace clustering for drill hope spectral data," *Journal of Applied Remote Sensing*, vol. 8, p. 083644, 2014.

[226] C. Payer, D. Stern, T. Neff, H. Bischof, and M. Urschler, "Instance segmentation and tracking with cosine embeddings and recurrent hourglass networks," *CoRR*, vol. abs/1806.02070, 2018.

[227] A. R. Zamir, A. Dehghan, and M. Shah, "GMCP-Tracker: Global multi-object tracking using generalized minimum clique graphs," in *Proceedings of the European Conference on Computer Vision*, 2012, pp. 343–356.

[228] B. Kim and J. C. Ye, "Mumford–Shah loss functional for image segmentation with deep learning," *IEEE Transactions on Image Processing*, vol. 29, pp. 1856–1866, 2019.

[229] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[230] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[231] A. Strehl and J. Ghosh, "Cluster ensembles — a knowledge reuse framework for combining multiple partitions," *The Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2003.

[232] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[233] B. Wu and R. Nevatia, "Tracking of multiple, partially occluded humans based on static body part detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2006, pp. 951–958.

[234] A. Kim, A. Ošep, and L. Leal-Taixé, "Eagermot: 3d multi-object tracking via sensor fusion," in *IEEE International Conference on Robotics and Automation*, 2021.

[235] Z. Xu, W. Zhang, X. Tan, W. Yang, H. Huang, S. Wen, E. Ding, and L. Huang, "Segment as points for efficient online multi-object tracking and segmentation," in *European Conference on Computer Vision*, 2020, pp. 264–281.

[236] S. Sun, N. Akhtar, H. Song, A. Mian, and M. Shah, "Deep affinity network for multiple object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 104–119, 2021.

[237] J. Luiten, P. Torr, and B. Leibe, "Video instance segmentation 2019: A winning approach for combined detection, segmentation, classification and tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

[238] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, "Online multi-object tracking with dual matching attention networks," in *European Conference on Computer Vision*, 2018, pp. 379–396.

[239] J. Son, M. Baek, M. Cho, and B. Han, "Multi-object tracking with quadruplet convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3786–3795.

[240] H. Sheng, J. Chen, Y. Zhang, W. Ke, Z. Xiong, and J. Yu, "Iterative multiple hypothesis tracking with tracklet-level association," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 12, pp. 3660–3672, 2019.

[241] H. Sheng, Y. Zhang, J. Chen, Z. Xiong, and J. Zhang, "Heterogeneous association graph fusion for target association in multiple object tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 11, pp. 3269–3280, 2019.

[242] C. Kim, F. Li, and J. M. Rehg, "Multi-object tracking with neural gating using bilinear LSTM," in *European Conference on Computer Vision*, 2018, pp. 200–215.

[243] K. Fang, Y. Xiang, X. Li, and S. Savarese, "Recurrent autoregressive networks for online multi-object tracking," in *IEEE Winter Conference on Applications of Computer Vision*, 2018.

[244] S. Schulter, P. Vernaza, W. Choi, and M. Chandraker, "Deep network flow for multi-object tracking," in *IEEE Conference on Computer Vision and Pattern Recog.*, 2017.

[245] S. Jin, W. Liu, W. Ouyang, and C. Qian, "Multi-person articulated tracking with spatial and temporal embeddings," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5657–5666.

[246] R. Ding, M. Yu, H. Oh, and W.-H. Chen, "New multiple-target tracking strategy using domain knowledge and optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 4, pp. 605–616, 2016.

[247] D. Stadler and J. Beyerer, "Improving multiple pedestrian tracking by track management and occlusion handling," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 958–10 967.

[248] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European conference on computer vision Workshops*, 2016, pp. 17–35.

[249] R. Raguram, J.-M. Frahm, and M. Pollefeys, "A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus," in *European conference on computer vision*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 500–513.

[250] V. Besnier, A. Bursuc, D. Picard, and B. Alexandre, "Triggering failures: Out-of-distribution detection by learning from local adversarial attacks in semantic segmentation," in *IEEE International Conference on Computer Vision*, 2021.

[251] Y. Hou, L. Zheng, and S. Gould, "Multiview detection with feature perspective transformation," in *European Conference on Computer Vision*, 2020.

[252] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *IEEE International Conference on Computer Vision*, 2017.

[253] D. Guan, J. Huang, A. Xiao, S. Lu, and Y. Cao, "Uncertainty-aware unsupervised domain adaptation in object detection," *IEEE Transactions on Multimedia*, 2021.

[254] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.

[255] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, "Deep subspace clustering networks," in *31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, 2017, pp. 23–32.

[256] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[257] P. Zhou, Y. Hou, and J. Feng, "Deep adversarial subspace clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[258] R. J. Mozhdehi and H. Medeiros, "Deep convolutional particle filter for visual tracking," in *IEEE International Conference on Image Processing*, 2017, pp. 3650–3654.

[259] B. Schölkopf, J. Platt, and T. Hofmann, *Efficient Learning of Sparse Representations with an Energy-Based Model*. MITP, 2007.

[260] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, Nov 2013.

[261] B. Wang, L. Wang, B. Shuai, Z. Zuo, T. Liu, K. L. Chan, and G. Wang, "Joint learning of convolutional neural networks and temporally constrained metrics for tracklet association," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, June 2016, pp. 386–393.

[262] J. Son, M. Baek, M. Cho, and B. Han, "Multi-object tracking with quadruplet convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017, pp. 3786–3795.

[263] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *28th International Conference on International Conference on Machine Learning*, ser. ICML'11, 2011, pp. 689–696.

[264] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *IEEE/CVF Conference on*

Computer Vision and Pattern Recognition. IEEE Computer Society, 2018, pp. 7482–7491.

[265] V. Bruni and D. Vitulano, "An improvement of kernel-based object tracking based on human perception," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 11, pp. 1474–1485, 2014.

[266] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *IEEE International Conference on Computer Vision*, 2017.

[267] J. KJ, J. Rajasegaran, S. Khan, F. Khan, and V. N. Balasubramanian, "Incremental object detection via meta-learning," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, nov 2021.

[268] X. Du, X. Wang, G. Gozum, and Y. Li, "Unknown-aware object detection: Learning what you don't know from videos in the wild," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[269] W. Liu, X. Wang, J. Owens, and Y. Li, "Energy-based out-of-distribution detection," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 21 464–21 475.

[270] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *International Conference on Learning Representations*, 2017.

[271] A. Bendale and T. E. Boult, "Towards open set deep networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, jun 2016, pp. 1563–1572.

[272] A. Paszke, S. Gross *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[273] D. Marcos, M. Volpi, and D. Tuia, "Learning rotation invariant convolutional filters for texture classification," in *International Conference on Pattern Recog.*, 2016.

[274] Y. A. Sheikh and M. Shah, "Trajectory association across multiple airborne cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 361–367, Feb 2008.

[275] P. A. Dias, H. Medeiros, and F. Odone, "Fine segmentation for Activity of Daily Living analysis in a wide-angle multi-camera," in *5th AMMDS in conjunction*

with The British Machine Vision Conference, 2017.

[276] W. Liu, O. I. Camps, and M. Sznaier, "Multi-camera multi-object tracking," *arXiv preprint arXiv:1709.07065*, 2017.

[277] T. Chavdarova and F. Fleuret, "Deep multi-camera people detection," in *IEEE International Conference on Mach. Learning and Applications*, 2017.

[278] J. Deng, W. Dong *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recog.*, 2009.