Surveillance Video Analysis with External Knowledge and Internal Constraints

Shoou-I Yu

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA, 15213, USA
www.lti.cs.cmu.edu

Thesis Committee:
Dr. Alexander G. Hauptmann, Carnegie Mellon University
Dr. Abhinav Gupta, Carnegie Mellon University
Dr. Yaser Sheikh, Carnegie Mellon University
Dr. Rahul Sukthankar, Google Research

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Language and Information Technologies
Copyright © 2016 Shoou-I Yu
Abstract

The automated analysis of video data becomes ever more important as we are inundated by the ocean of videos generated every day, thus leading to much research in tasks such as content-based video retrieval, pose estimation and surveillance video analysis. Current state-of-the-art algorithms in these tasks are mainly supervised, i.e. the algorithms learn models based on manually labeled training data. However, it is difficult to manually collect large quantities of high quality labeled data. Therefore, in this thesis, we propose to circumvent this problem by automatically harvesting and exploiting useful information from unlabeled video based on 1) out-of-domain external knowledge sources and 2) internal constraints in video. Two tasks from the surveillance domain are targeted: multi-object tracking and pose estimation.

Being able to localize where each individual is at each time instant would be extremely useful in surveillance video analysis. We tackled this challenge by formulating the problem as an identity-aware multi-object tracking problem. An existing out-of-domain knowledge source: face recognition, and an internal constraint: the spatial-temporal smoothness constraint were used in a joint optimization framework to localize each person. The spatial-temporal smoothness constraint was further utilized to automatically collect large amounts of multi-view person re-identification training data. This data was utilized to train deep person re-identification networks which further enhanced tracking performance on our 23 day 15 camera data set which consists of 4,935 hours of video. Experiments showed that our tracker has the ability to locate a person 57% of the time with 70% precision.

Reliable pose estimation in video enables us to understand the actions of a person, which would be very useful in surveillance video analysis. However, domain differences between surveillance videos and the pose detector’s training set often cause degradation in pose estimation performance. Therefore, an unsupervised domain adaptation method based on constrained self-training was proposed. By utilizing an out-of-domain image-based pose detector and spatial-temporal smoothness constraints, our method can automatically collect in-domain pose estimation training data from video for domain adaptation. Results showed that the pose detector trained on in-domain data collected with our unsupervised approach is significantly more effective than models trained on more out-of-domain data.

Finally, based on our improved multi-object tracker and pose detector, long-term analyses of nursing home resident behavior were performed. Results showed that the output of our tracker was accurate enough to generate for each nursing home resident a reasonable “visual diary”, which not only shows the activities performed throughout the
day, but also accumulated long-term statistics which are simply too tedious to compute manually. Furthermore, pose detectors were utilized to detect eating behavior of nursing home residents, which would also have the potential to aid the assessment of health status of nursing home residents.

In conclusion, our results demonstrate the effectiveness of utilizing external knowledge and internal constraints to enhance multi-object tracking and pose estimation. The methods proposed all attempt to automatically harvest useful information directly from unlabeled videos. Based on the promising experimental results, we believe that the lessons learned could be generalized to other video analysis problems, which could also benefit from utilizing external knowledge or internal constraints in an unsupervised manner, thus reducing the need to manually label data. Furthermore, our proposed methods potentially open the door to automated analysis on the ocean of surveillance video generated every day.
Contents

Abstract i

1 Introduction 1

2 Related Work - From Weak Supervision to No Supervision 7
  2.1 From Object Recognition to Object Detection and Semantic Segmentation with Weak Supervision 8
  2.2 Noisy Internet-Retrieved Labels as Weak Supervision 10
  2.3 Domain Adaptation with No Supervision 10
  2.4 Representation Learning with No Supervision 13
  2.5 Other Tasks Utilizing Weak or No Supervision 14
  2.6 Summary 15

3 Multi-Object Tracking with Face Recognition 17
  3.1 Related Work - Multi-Object Tracking 19
    3.1.1 Object Localization 19
    3.1.2 Appearance Models 20
    3.1.3 Motion Models 21
    3.1.4 Data Association 21
  3.2 Tracker Overview 24
  3.3 Notations 25
  3.4 Manifold Construction based on Appearance and Spatial Affinity 26
    3.4.1 Modeling Appearance Affinity 26
    3.4.2 Modeling Spatial Affinity 28
  3.5 Spatial Locality Constraint 29
  3.6 Nonegative Matrix Optimization 30
  3.7 Solution Path Algorithm Optimization 33
    3.7.1 Block Coordinate Descent 36
    3.7.2 Iterative Projection Method 37
  3.8 Comparing Nonnegative Matrix Optimization and Solution Path Algorithm 39
  3.9 Experiments and Results 39
    3.9.1 Data Sets 39
    3.9.2 Baselines 41
    3.9.3 Implementation Details 43
    3.9.4 Evaluation Metrics 44
Chapter 1

Introduction

Automatic understanding of video content has a wide variety of applications including, video retrieval [1], surveillance [2] and health care [3]. Furthermore, automated analysis becomes ever more important as we are inundated by the ocean of user-generated and surveillance videos created everyday. Therefore, much research have focused on designing effective algorithms to analyze the content of videos.

In general, supervised methods have achieved promising results in image and video analysis tasks [4–8]. Supervised methods heavily rely on training data to create models which can better generalize to unseen data. To achieve effective generalization, the training data set needs to 1) be large enough in quantity and 2) match the distribution of the testing data. Large enough training data is the key to the success of many algorithms [9], such as the success of deep-neural networks in object recognition [5], and according to statistical machine learning theory [10], more data can decrease the difference between testing error and training error. Also, it is required that the training data has the same distribution as the testing data. If the distributions do not match, performance will significantly degrade [11]. The two aforementioned points motivate us to collect more training data from diverse sources such that the training distribution can better approximate the true testing distribution.

Training data can be collected and labeled manually, which is how many existing image and video data sets were collected. However, manual labeling is a difficult and very labor intensive process. To alleviate the tediousness of labeling, some researchers have created data sets such as ImageNet [12] with crowdsourcing based on Amazon Mechanical Turk. Much research have also gone into how to maximize the utility of crowdsourcing [13, 14]. Another direction of data collection is through “gamification” [15, 16], where the data annotation task is transformed into a game so that as humans play the game, more annotations are collected. Though these methods have shown to be effective,
crowdsourcing requires actual money, and gamification requires designing an interesting game, which is challenging. Also, the manual annotation process is required whenever one acquires data from a new domain. Furthermore, for privacy sensitive data, it may not be convenient for the public to annotate the data. Finally, data evolve over time, thus forming new labels of interest. So what was labeled in the past may no longer be of interest now, and to stay up-to-date, the labeling process needs to be run continuously. Therefore, based on the aforementioned shortcomings, one interesting question becomes whether one could alleviate the process of data collection by trying to directly collect useful information from unlabeled data.

In this thesis, we are interested in exploring the idea of automatically extracting useful information directly from unlabeled data to enhance the task at hand. We believe that the key to unlocking the useful information hidden in the large amounts of unlabeled data available is to rely on two main sources of information: 1) existing out-of-domain knowledge sources and 2) internal constraints.

### Exploiting External Knowledge

Instead of collecting manual annotations for each specific task, one can try utilizing the already existing highly related external resources to tackle the task. For example, if one would like to train a person detector for a specific traffic scene, instead of directly annotating people in the scene, one can start with an external generic object detector and adapt it to the scene with the aid of some internal constraints [17, 18]. Another example is tackling the semantic segmentation task with existing external but related knowledge. Collecting semantic segmentation training data is very expensive, because it requires delineating the boundaries of each object. Therefore, [19, 20] utilized existing cheap image/video-level labels and other constraints to learn a classifier to perform semantic segmentation. [21] utilized detection proposals from an external object detector as a starting point for their semantic segmentation algorithm. In sum, there are already many existing resources available to aid us in tackling a new task on a new data set, thus potentially alleviating the need for manual labeling.

### Utilizing Internal Constraints

Internal constraints refer to specific patterns which the data set of interest usually follows. For example, objects in a video should move smoothly is an internal constraint heavily used in tracking. A very exciting scenario to exploit internal constraints is the automatic collection of training data from unlabeled data. Many automated systems have been proposed to acquire knowledge from the vast amount of unstructured and
unlabeled information readily available on the Internet. These systems heavily rely on internal constraints of the data to harvest useful information. For text, the Never Ending Language Learner (NELL) [22] automatically learns facts of different entities from unstructured web documents. An example constraint which was used in web documents is entities enclosed in the same HTML list structure are highly likely to be entities of similar types [23]. In images, the Never Ending Image Learner (NEIL) [24] learns facts of different entities from unlabeled images. Constraints utilized are common sense relationships between categories such as “wheel has round shape” or “pyramid is found in Egypt”. Learn EVerthing about ANything (LEVAN) [25] is another image based system which utilizes image search engines and text language models to automatically learn object detectors. Other work have proposed to utilize unlabeled video data to enhance image analysis tasks. [26] utilized unlabeled videos to collect training examples to learn improved static image action detectors. [27–29] improves object detection by harvesting positive examples from unlabeled videos in an unsupervised fashion. Overall, the aforementioned systems share two core concepts:

- They all can operate on unlabeled data in a (nearly)\(^1\) automatic way.
- They all utilize assumptions or constraints to enable them to extract useful information from unlabeled data.

These two points combined together enable these methods to exploit very large amounts of unlabeled data readily available on the Internet. A very attractive property of these systems is that even if the assumptions or constraints utilized are very strict, i.e. a high precision low recall scenario, the system is still able to collect a lot of useful information because there are a near infinite amount of unlabeled data. Colloquially speaking, even if 99.99% of the data was thrown away, the remaining 0.01% will still give us very large amounts of useful information if the system had access to a very big collection of unlabeled data. This spirit is what makes these methods very attractive and useful. Therefore, the key advantages of these systems are three fold:

1. Unsupervised methods which can exploit very large amounts of unlabeled data have the potential to collect a lot of useful information such as training data.
2. Training data can be directly extracted from unlabeled test sets, thus decreasing domain difference between training and testing sets.
3. Training data can be automatically updated as the unlabeled data evolve over time.

\(^1\)These systems may face the problem of semantic drift, and in the case of NELL, humans manually clean the database periodically.
Motivated by the success in previous systems which utilized external knowledge or internal constraints, we explored applying these principles to surveillance video analysis, specifically on multi-object tracking and pose estimation.

**Thesis Overview**

In this thesis, we tackled two surveillance video analysis tasks: multi-object tracking and pose estimation, with two information sources: external knowledge and internal constraints, in a (nearly) unsupervised manner. In this thesis, the term “unsupervised” is interpreted in a slightly relaxed manner, as some of our methods still require minimal human effort. In our view, a method is unsupervised if the amount of human labor required by the method does not scale with the amount of data utilized by the algorithm. Also, our methods are unsupervised in that no video-level annotations are required. Our methods still utilized pre-existing out-of-domain static-image resources, which are viewed as external knowledge to train the necessary detectors.

It would be very useful in surveillance video analysis if one was able to locate and identify each person at each time instant. We tackled this challenge by formulating the problem as an identity-aware multi-object tracking task. Our multi-object tracker was augmented with an external resource: face recognition, which provides identity information to person detections with a recognizable face. This can be viewed as label information directly extracted from unlabeled surveillance video. However, most person detections do not have a visible or recognizable face. Therefore face recognition is only sparse label information and additional cues including appearance features and motion constraints were utilized to perform multi-object tracking. The motion constraints correspond to an internal constraint in video: spatial temporal smoothness, which assumes that a person should move smoothly in the video and cannot be at two places at the same time. The tracking problem was formulated as a constrained quadratic optimization problem, which we solved by two proposed algorithms: nonnegative matrix optimization and the solution path algorithm. The final tracking output will provide access to the location of each individual at each time instant. Multi-object tracking experiments were run on up to 4,935 hours of surveillance video data, which is to the best of our knowledge the biggest multi-object tracking experiment to date. More details are in Chapter 3.

Another direction to enhance multi-object tracking is by replacing hand-crafted appearance features with discriminatively learned deep features used for person re-identification. Person re-identification is the task of distinguishing whether two person detections belong to the same individual or not. However, learning such deep features requires a lot of
training data. Therefore, to collect data, we also utilized the spatial-temporal smoothness constraint. In multi-camera surveillance scenarios, if two cameras both independently detect there is a person at a specific location, then it is highly likely that they are viewing the same individual from two different views. The same individual viewed from two different angles is exactly the training data used for person re-identification. Thus a standard person re-identification network can be trained to learn a discriminative representation. Experiments showed that the deep representations learned further improved tracking performance. More details are in Chapter 4.

Reliable pose estimation enables us to understand the actions of a person, which would be very useful in surveillance video analysis. One issue is that the domain of the pose detector’s training data, which is often static-images, does not match the surveillance video domain, thus motivating us to perform unsupervised domain adaptation. We proposed to utilize constrained self-training to directly collect in-domain samples from the testing set. Self-training \cite{30, 31} is the process of adding testing instances with high confidence predictions into the training set to enhance the current model. However, instances with high confidence may not always be correct. Therefore, the main idea is to utilize the spatial-temporal smoothness constraints to perform checks on whether a pose estimation result on the testing set is correct. The assumption is that pose estimation results in neighboring frames should vary smoothly. If drastic change is observed, then at least one of the pose estimation results are incorrect. Based on this assumption, pose estimations instances which not only have high prediction scores but also pass the smoothness check are even more likely to be correct. These instances from the testing set are eligible to be added into the training set to automatically adapt the pose estimation model to the testing set. Results showed that the pose detector trained on in-domain data collected with our unsupervised approach was significantly more effective than models trained on more out-of-domain data. More details are in Chapter 5.

Based on the multi-object tracker and pose detector that was developed, we performed long-term surveillance video analysis on nursing home surveillance data. Based on the output of multi-object tracking, our system was able to detect events-of-interest such as room changes, sit down and stand up (with an aid of a sitting detector), and interactions. With tracking of each person over 23 days, long-term statistics such as the distance walked per day and total time spent in interactions were accumulated. With our pose detector, a “take a bite” detector was created. This detector could facilitate the analysis of eating behavior of nursing home residents. More details are in Chapter 6.
In summary, the thesis statement is as follows.

**Thesis Statement:**
Through utilizing 1) existing out-of-domain knowledge sources and 2) internal constraints in video, we can design algorithms which enhance the performance of surveillance video analysis tasks in an unsupervised fashion.

**Thesis Contributions**

In this thesis, we present our proposed approaches for unsupervised surveillance video analysis in multi-object tracking and pose estimation.

1. Multi-object tracking with face recognition: we designed a method which utilizes an external knowledge source: face recognition to enhance multi-object tracking. The tracking problem was solved with two different optimization techniques (CVPR ’13 [2], CVPR ’16 [32]).

2. Unsupervised collection of person re-identification training data for tracking: we present an unsupervised method based on the spatial-temporal smoothness constraint to collect person re-identification data, which enabled us to learn deep appearance features to further enhance multi-object tracking.

3. Unsupervised domain adaptation for pose estimation with constrained self-training: we proposed to automatically develop effective video pose detectors by adapting existing image pose detectors to video in a constrained self-training framework (ECCV ’14 [33]). Spatial-temporal constraints were utilized for effective self-training. Experiments were performed on a newly annotated Caremedia nursing home pose data set with 3.2K poses.

4. Long-term surveillance video analysis: we utilized our multi-object tracker and pose detector on analyzing 23 days of nursing home surveillance data. Experiments on surveillance video summarization, eating detection, and long-term statistics analysis were performed.

The thesis is organized as follows. Chapter 2 surveys related work which have a similar unsupervised spirit as our proposed approaches. Chapter 3 presents work on multi-object tracking with face recognition. Chapter 4 details how we collected person re-identification training data from multi-camera surveillance environments. Chapter 5 reports work on unsupervised adaptation of image-based pose estimators to video. Chapter 6 presents long-term surveillance video analysis based on our multi-object tracker and pose detector. Finally Chapter 7 concludes the thesis.
Chapter 2

Related Work - From Weak Supervision to No Supervision

This thesis focuses on analyzing surveillance video by utilizing external knowledge and internal constraints, which have also been widely used in other domains. In this chapter, we present an overview of existing computer vision work which have a similar spirit as this thesis. The existing work is categorized into methods which used either weak supervision or no supervision.

Weakly supervised learning, in contrast to fully supervised learning, means that the learner is somehow handicapped in terms of the training data. There are many different ways to be handicapped in terms of training data, thus the terms “weakly supervised”, “weak supervision” or “distance supervision” have been interpreted in various ways in the literature. Possible interpretations are as follows:

1. Training data is scarce [34, 35].
2. Training data is noisy [36, 37].
3. Training data is only provided for another related task, which usually have labels with less granularity than the task of interest [38–41].

Overall, in-domain training data is still provided, but the labeled data is not as comprehensive and perfect as the labeled data used during full supervision.

Taking weak supervision one step further are methods that require no supervision. For these methods, in-domain training data is not provided at all. Note that this does not necessarily mean that the method did not use any manually labeled training data. A pre-trained model trained on existing out-of-domain instances can still be used. A popular example of this class is unsupervised object detector adaptation [27, 28, 42, 43]. Given an object detector trained on labeled training data in the source domain, the goal
Related Work - From Weak Supervision to No Supervision

is to automatically adapt the detector to the target domain. As the adaptation process requires no human intervention, these methods require no supervision.

In order to deal with the lack of complete labeled data, methods utilizing either weak supervision or no supervision often require additional assumptions or constraints to overcome this handicap. The assumptions or constraints include real-world physical constraints (i.e. an object cannot move too quickly in video), assumptions on the structure of the image/video (i.e. the size of objects), or assumptions on classifier confidence being correlated with accuracy. These assumptions play a very important role in methods which are weakly supervised or unsupervised, and often the main novelty of a paper is the formulation and exploitation of these assumptions. In the following sections, we will review existing computer vision work which utilize weak supervision or no supervision.

2.1 From Object Recognition to Object Detection and Semantic Segmentation with Weak Supervision

There can be 3 different granularities of object labels when labeling an image/video. An object label can be on an image/video-level, which indicates the existence of the label but not the spatial locations. These labels are suitable for the object recognition task, which only tries to recognize what objects are visible. A more fine-grained labeling is by adding a bounding box which indicates the location of the object. These labels are suitable for the object detection task, which further tries to localize each object. The most fine-grained labeling is by further delineating the pixel-level boundaries of the object. This is suitable for the semantic segmentation task, which requires not only localizing the object but also finding the boundaries of each object. Clearly, having the boundaries of the object provide the richest information, but it is also the most time consuming to annotate. Therefore, an interesting question becomes whether it is possible to use less fine-grained but cheaper labels and output results with higher granularity, i.e. utilize object recognition labels to learn a model which outputs a semantic segmentation. The main intuition and assumption behind why this may work is that an object recognition system which works well implies that the system has some idea of the location of the object. This task is weakly supervised because though labels are provided for in-domain data, the labels utilized are from a related task which has labels with less granularity than the target task. One can also view the labels from the related tasks as external knowledge to the target task.

In the following few paragraphs, we will briefly describe the related work in this direction. For clarity, related work are categorized into methods which operate on either static
images or video. They are discussed separately as video has an extra temporal aspect and different methods were used. Also, methods which have different output granularities are also separated.

**Image-level labels to bounding boxes:** The main goal is to utilize image-level labels and localize the objects in the image. One popular method is to treat the bounding box of the object as a hidden variable, which can then be found with variants of SVMs [41, 44], probabilistic latent semantic analysis [45], or deep learning [46]. A more thorough survey is presented in [47].

**Image-level labels or bounding boxes to segmentation:** Previous work have proposed to utilize image-level labels [19, 48, 49] or object bounding boxes [50] combined with constraints such as the size of the foreground and background [51] to perform semantic segmentation. Common learning methods utilized include Multiple Instance Learning [19] combined with multi-task learning [48] or deep learning [49]. Other methods took this one step further by jointly learning over weak image-level labels and strong pixel-level labels in a semi-supervised framework [52, 53].

**Video-level labels to bounding boxes:** According to [54], there is a clear domain difference in videos and images, which causes degradation in object detection performance when an object detector trained on still-images are predicted on video frames. Therefore, [55] proposed to automatically find bounding boxes of objects in videos based on video-level labels. Assuming that the video only contains objects of a target class, the algorithm first computed spatio-temporal tubes based on motion segmentation for each video. Then a joint one-tube-per-video selection was performed to find the most coherent set of tubes across all videos. These tubes were then used to train an object detector.

**Video-level labels to segmentation:** Another popular topic is to perform spatio-temporal segmentation of objects in video with either very few frame-level labels [56] or video-level labels [20, 57]. [21] on the other hand utilizes a generic object detector as weak supervision. Regardless of the source of supervision, all these methods utilize some sort of motion consistency, spatial-temporal smoothness or tracking constraint to guide their learning process to find the relevant semantic segments in the video.

Some methods take weakly-supervised object detection one step further by utilizing no training data at all and relying only on large amounts of unlabeled data. The intuition is that if enough data is provided, then commonly appearing objects can still be grouped together and “discovered”. There are two tasks under this definition: co-localization and object discovery. For image and video co-localization, the input is a set of unlabeled images or video, and the output are bounding boxes which localize objects of the same
class. It is unknown to the algorithm which object is in the data, but it is assumed that the object of interest is visible in most of the unlabeled images/video. [58] proposed to solve image co-localization by jointly utilizing an image similarity/discriminability model and a bounding box similarity/discriminability model in a constrained quadratic optimization framework. [59] further extends this to video by adding the temporal consistency constraint. Object discovery takes co-localization one step further by operating on large amounts of videos which include different classes of objects. [60] utilizes inter-video matching and intra-video tracking to find spatio-temporal tubes which localizes different objects.

2.2 Noisy Internet-Retrieved Labels as Weak Supervision

Another weakly supervised learning approach to learn object detectors or even segmentation masks of each object is through utilizing noisy training data crawled from the search engine or photo-sharing sites, thus less or even no manual annotation is required. [24, 25, 36, 61, 62] can utilize noisy training data crawled from Flickr or top results of an image search engine to learn multiple object classifiers. [37, 63, 64] takes this one step further by performing simultaneous object discovery and segmentation. In the advent of deep learning, [65] demonstrated how to train a CNN-based object detector on search engine results. [65] first trained an initial CNN on the easy images downloaded from Google image search. Then, the network was fine-tuned on the more realistic images from Flickr plus some modifications based on the confusion matrix of the initial CNN.

Similar work have also been done on training video-level event detectors and semantic concept detectors. [66] augmented manually labeled event video with top-ranked videos from YouTube search to enhance event recognition. [67] utilized tags provided in the YFCC data set [68], which includes 0.8 million weakly-annotated videos from Flickr to train semantic concept detectors. These detectors have shown to be very effective in low-resource event detection [1, 7].

2.3 Domain Adaptation with No Supervision

The goal of domain adaptation with no supervision, which is more commonly known as unsupervised domain adaptation, is to adapt a model trained on the source domain to an unlabeled target domain without any supervision. There are many high-level directions to perform unsupervised domain adaptation. One direction is to somehow “connect” or “re-align” the source and target domains in an unsupervised fashion. Another direction
tries to collect training samples from the testing set and perform self-training. In this section, we will detail existing work in these two directions, especially with a focus on self-training because it is also utilized in this thesis. For a more comprehensive survey on domain adaptation, we refer the readers to the following survey papers, which talks about unsupervised domain adaptation for person detection [69], visual domains in general [70] and natural language processing [31].

Given the source and target distributions, one can design algorithms to try to align or minimize the difference between the two domains. [71] modeled sampling bias between the source and target domain with covariate shift. [72, 73] found subspaces in the geodesic path which connected the source and target domains on a Grassmanian manifold. [74] found latent domains which have maximum distinctiveness and learnability. [75] directly learned max-margin non-orthogonal hyperplanes in the target domain. [76] tried to find a single deep feature representation for both the source and target domain. The deep feature representation was designed so that it is difficult to distinguish from the representation alone whether the current instance came from the source or target domain. [77] learned a transformation such that the source domain samples can be reconstructed by nearest neighbor samples found in the target domain. Unlike self-training, none of these methods tried to collect “training data” from the testing set.

A large portion of unsupervised domain adaptation approaches fall into the self-training (i.e. bootstrapping) paradigm. The first step of self-training is to perform prediction on the testing set to acquire pseudo-labels for the testing instances. Then, self-training assumes that high confidence pseudo-labels in the testing data are highly likely to be correct, and one could add these labels back in the training set and reap additional gains [78, 79]. However, high confidence pseudo-labels may not necessarily be correct, thus “checks” which are independent of the classifier itself are utilized to ensure that the pseudo-labels of the instances added are even more likely to be correct. Taking [30] as an example, the main idea is to collect potential training data for eye detection from unlabeled images based on different selection metrics. A classifier confidence based metric and a classifier independent metric were proposed. The classifier confidence based metric selected testing instances with high confidence pseudo-labels. The classifier independent metric computed a classifier independent distance between the testing instances and all the current training data. Testing instances with smaller distance than other training data were selected. Experiments showed that the classifier independent metric was more reliable than the classifier confidence metric, which demonstrates the importance of independent checks.

A myriad of self-training approaches have been proposed to tackle unsupervised domain adaptation in multiple vision tasks, especially in the task of adapting object detectors.
to video. As this is highly related to our constrained self-training for pose estimation, we will describe this in more detail in the following sections.

Self-training for Adapting Object Detectors to Video

Object detectors have achieve impressive results on static-images [4], but due to domain differences between images and video [54], the object detectors only trained on static images may not work well on video. Therefore, much work have proposed to utilize self-training to adapt an image-based object detector to video. [80] utilized an Adaboost [81] framework combined with Co-training [82] to progressively improve object detectors. As the base classifiers learned from Adaboost are uncorrelated by design, they can be used as independent checks for the output of other base classifiers. [83] proposed that a detected body part is only confident when all other parts also have confident detections. This criteria was used when adding new training data to enhance a part-based person detector. [84] utilized objects detected with high confidence combined with Multiple Kernel Learning to adapt detectors to different lighting condition in video. [85] proposed to utilize dense patch sampling and sparse binary vector encoding to more accurately identify low-confidence but correct person detections. [86] improved an object detector by utilizing superpixels instead of the common Haar-like features. To create an object detector for video, a bag-of-superpixel-based SVM was trained, and instances with confidence in the top one-third were added into the positive set. Instances in the bottom one-third were added into the negative set, and the SVM was retrained and this process repeated.

Adaptation with the Tracking or Spatial-Temporal Smoothness Check

A crucial independent check in video is the tracking or spatial-temporal smoothness check, i.e. an object should not move too far in adjacent frames, which is conveniently available in nearly all unedited videos. This check or variants of it was heavily utilized in unsupervised adaptation of object detectors to video.

For person detector adaptation, [17, 18] adapted a generic pedestrian detector to a specific traffic scene by collecting training data which passed through multiples checks, including checking the aspect ratio of the detected bounding box, checking whether the detection belongs to a big cluster (if yes, that means it is static background and not a person), comparing with the usual walking paths of the pedestrians in the scene, and making sure the pedestrian does not overlap with a tracked moving vehicle. [87] utilized boosting combined with network-flow tracking to find confident training data that is fed back into the boosting classifier. [28, 88] filtered noisy pseudo-labels by only
selecting person detections which are coherent with tracking results. A multiple instance learning boosting classifier or a random fern classifier was employed to perform learning on top of noisy labels. [89] proposed to combine two independent sources: person detector and background subtraction tracking to collect in-domain person detection training data. [42] proposed to transform the noisy pseudo-label selection problem to a classifier selection problem, where multiple classifiers were trained on subsets of pseudo-labels, and only the classifiers which perform well on the source domain were selected. These classifiers then become an ensemble of weak detectors for the testing domain. [43] proposed to utilize online multi-task learning to adapt an object detector. Each tracked instance had its own set of detector weights, which were updated online yet regularized by the weights of the other detectors which detect the same class of objects. The weights for these instances were averaged to form a generic object detector for that class. Note that most of these methods are not limited to person detection and potentially can be applied to other classes of objects.

For generic object detector adaptation, [27, 90, 91] proposed to utilize self-paced learning, adaptive SVMs and large-margin embedding classifiers respectively to adapt a static-image detector to video. This is based on the training samples collected by confident detections and tracking in the video. Tracking is advantageous in that new views of the object can potentially be harvested. [29] proposed an effective way to bootstrap new object detectors through a combination of pre-trained CNNs, exemplar learning and region-based tracking.

2.4 Representation Learning with No Supervision

In the advent of very large image and video collections, much research have been done to learn discriminative representations on unlabeled data. The main advantage of these methods is that since absolutely no supervision is required, the method can be applied to any data set, and the representations learned will be in-domain for that data set. [92] utilized sparse coding on large numbers of unlabeled images to learn a robust set of bases, which were used to effectively encode other images for image classification. [93] learned discriminative image patches for object detection by selecting patches which are both representative and discriminative. The intuition is that a cluster of patches was discriminative if a SVM could not only be trained based on it, but also find a reasonable number of similar looking patches in the validation set. [94] utilized an autoencoder learned on auxiliary natural images to learn a representation which is more robust to variations in single-object tracking. [95] utilized patch tracking to automatically learn a representation for video. The intuition is that the similarity between two tracked patches
should be larger than two randomly selected patches. [96] utilized context to learn an image representation. The intuition is that given a small image patch, a good set of features should have the ability to predict the content in the near vicinity of the image. [97] learned a representation based on sequential verification for video. The intuition is that given three not necessarily consecutive but ordered frames $a$, $b$, and $c$, a good representation should be able to verify whether frame $b$ follows $a$ and precedes frame $c$. Though the aforementioned methods tackle their respective tasks very differently when compared with weakly supervised learning methods, they all still utilized some sort of assumption (i.e. sparsity, compactness, tracking, context) to constrain the learner to learn something useful.

### 2.5 Other Tasks Utilizing Weak or No Supervision

In this section we report on other works which also utilized weak or no supervision.

In order to collect person detection training data in a cheaper way, [98] utilized background subtraction in unlabeled videos to automatically find training samples for person detection. [40] proposed to decrease annotation cost of person detection by only marking the approximate center of mass of a person instead of an entire bounding box. Since the annotations were weaker, background subtraction and an automatically learned person prior was utilized to train the person detector.

For action recognition related tasks, [38] utilized video-level labels and Multiple Instance Learning (MIL) to learn a model which localized actions in both spatial and temporal space in videos. [99] proposed to utilize matrix completion instead of MIL for weakly supervised action localization. [26] proposed to augment its image-based action recognition training data by looking for samples which were very similar to the training set in unlabeled videos. [39] utilized image-level action recognition labels to locate which objects were important for this action. [100] utilized ordering constraints of actions acquired from a movie script as weak label information to learn action detectors.

To collect more action recognition training examples in a cheaper way, our work [101] proposed to utilize the inherent speech-action correlation in instructional videos. An example is shown in Figure 2.1a, where a “cut onion” action is shown. By looking for action signifiers such as “we will” and utilizing a dependency parser to find verb-noun pairs, action examples were collected with the pipeline shown in Figure 2.1b. With this pipeline, we were able to collect a large variety of action training examples as shown in Figure 2.2. For example, instead of just learning the action “cut”, our method provided training data which included the objects associated with the action, such as “cut onion”,
“cut paper” and “cut hair” as shown in Figure 2.2b. Also, in Figure 2.2a, our system was able to collect diverse positive examples for “drill hole”. Holes could either be drilled in wood, in a wall or even in ice. In sum, our unsupervised method is advantageous in that as more instructional videos are uploaded, our system can acquire more diverse action training examples for free. Related work [102, 103] also utilized visual and speech cues to better understand cooking or other instructional videos.

2.6 Summary

This chapter presents an overview of methods which utilized weak supervision or no supervision. As these methods were handicapped in terms of training data, external knowledge combined with task-specific assumptions or constraints were placed to guide the learning process to converge to a reasonable result. In many cases, how the external knowledge or constraints were utilized were the main novelties of the paper. For example, the same spatial-temporal smoothness constraint can be utilized in object discovery, adapting object detectors to video, and learning unsupervised representations. In this thesis, we further extended the same high-level idea of utilizing external knowledge or internal constraints to multi-object tracking and pose estimation.
(A) A collected “cut onion” action example based on the speech-action correlation in instructional videos.

(B) Figure of our pipeline for unsupervised harvesting of action examples from instructional videos.

Figure 2.1: Figures depicting the intuition and more detailed pipeline of our method which utilized instructional videos to collect action examples.

(A) Positive “drill hole” examples harvested (B) Positive examples for verb “cut” collected from instructional videos in an unsupervised manner. Our method collected drilling holes in diverse contexts such as on wood, in walls and on ice; from cutting with scissors, knives to paper cutters.

Figure 2.2: Figures showing the action examples found within instructional video in an unsupervised manner.
Chapter 3

Multi-Object Tracking with Face Recognition

Surveillance cameras have been widely deployed to enhance safety in our everyday life. In addition, surveillance camera video footage can be used to analyze long term trends in the environment. One first step to analyzing surveillance video is to locate and identify each person in the video. This could be achieved if one had pre-trained identity-specific person detectors, i.e. each detector recognizes a single individual, or if one had other sources such as gait [104] to recognize a person. However, these detectors are nearly impossible to train as there will many unique individuals in surveillance scenes, and it will be very difficult and tedious to collect training data for each identity-specific person detector or gait classifier. Therefore, in order to obtain identity information in an automated way from unlabeled videos and utilize this information to locate each individual at each time instant, we propose to combine face recognition with multi-person tracking.

Face recognition, which is an external knowledge source, enables us to extract identity information from unlabeled videos. However, face recognition is not available in most frames. Therefore, a multi-object tracking algorithm which can utilize face recognition information was proposed to perform identity-aware tracking. Face recognition not only assigns identities to each tracked individual, but also enhances tracking performance by pinpointing the location of a specific individual at a given time. This additional information can lower the chance of identity confusions during the tracking process. The final result of combining face recognition and multi-person tracking are trajectories of different people with identities assigned to each trajectory. This effectively provides the location of each individual at each time instant. The only downside to our approach
is that a gallery is required to perform face recognition, which can be created either manually or through face clustering. More details on this issue are in Section 3.9.7.

Our proposed identity-aware tracking algorithm is as follows. Under the tracking-by-detection framework [105], the tracking task can be viewed as assigning each person detection result to a specific individual. Label information for each specific person are acquired from face recognition. However, as face recognition is not available in most frames, face recognition information is propagated to other frames using a manifold learning approach, which captures the manifold structure of the appearance and spatial layout of the detected people. The manifold learning approach is formulated as a constrained quadratic optimization problem. The constraints included are the mutual exclusion and spatial locality constraints to constrain the final solution to be a reasonable multi-person tracking output, i.e. a person detection can only belong to one individual and an individual cannot be at multiple places at the same time. These constraints make the optimization problem very difficult to solve, so we proposed two optimization methods to solve this problem: nonnegative matrix optimization and the solution path algorithm.

Tracking experiments were performed on challenging data sets including a 116.25 hour and a 4,325 hour complex indoor tracking data set. Our experiments show that our
method is effective in localizing and tracking each individual in long-term surveillance video. An example output of our algorithm is shown in Figure 3.1, which shows the location of each identified person on a map in the middle of the image. This is analogous to the *Marauder’s Map* described in the Harry Potter book series [106].

In sum, the main contributions of this chapter are as follows:

1. We propose an identity-aware multi-object tracking formulation which leverages identity information as label information in a manifold learning framework. The algorithm is formulated as a constrained quadratic optimization problem.
2. We propose two methods to optimize a constrained quadratic optimization problem: nonnegative matrix optimization and the solution path algorithm.
3. Multi-camera multi-object tracking experiments on 4,325 hours of surveillance video in a complex indoor 15-camera environment were performed. To the best of our knowledge, this is the longest multi-camera multi-object tracking experiment to date.

In the following sections, Section 3.1 gives an overview of existing multi-object tracking methods. Section 3.2 gives an high-level overview of the two trackers developed. Details of the two trackers are described from Section 3.3 to Section 3.7. Then experimental results are presented in Section 3.9 and Section 3.10 concludes this chapter.

### 3.1 Related Work - Multi-Object Tracking

A main line of multi-object tracking work follows the tracking-by-detection paradigm [105], which has four main components: object localization, appearance modeling, motion modeling and data association. The object localization component generates a set of object location hypotheses for each frame. The localization hypotheses are usually noisy and contain false alarms and misdetections, so the task of the data association component is to robustly group the location hypotheses which belong to the same physical object to form many different object trajectories. The suitability of the grouping can be scored according to the coherence of the object’s appearance and the smoothness of the object’s motion, which correspond to appearance modeling and motion modeling respectively. We now describe the four components in more detail.

#### 3.1.1 Object Localization

There are mainly three methods to find location hypotheses: using background subtraction, using object detectors, and connecting single-frame detection results into tracklets.
The Probabilistic Occupancy Map (POM, [107]) combines background subtraction information from multiple cameras to jointly locate multiple objects in a single frame. It has been shown to be very effective in multi-camera environments [107–110]. However, POM requires the discretization of the tracking space, and some precision may be lost. Also, when the placement of cameras is non-ideal, such as on long corridors where cameras only view the principal direction of the corridor [2], the POM localization results are not as accurate. Lastly, when there are different kinds of moving objects in the scene, POM cannot distinguish between the different kinds of tracked objects.

Utilizing object detector output is one of the most common ways to localize tracking targets [2, 105, 111–118]. The main advantages of using object detectors are 1) enables the automatic initialization and termination of trajectories, and 2) alleviates template drift as the same detector is used for all frames. The main disadvantage is that a reliable general-purpose detector is required for the object to be tracked.

Localized objects in each frame could be connected to create tracklets [109, 119–124], which are short tracks belonging to the same physical object. Tracklets are formed in a very conservative way to avoid connecting two physically different objects. As tracklets merge multiple location hypotheses, they can be used to enhance the efficiency of the tracking process [109].

### 3.1.2 Appearance Models

Appearance models discriminate between detections belonging to the same physical object and other objects. Color histograms [2, 109, 111, 113, 119, 123, 125–127] have been widely used to represent the appearance of objects, and the similarity of the histograms is often computed with the Bhattacharyya distance [113, 127]. Other features such as Histogram of Oriented Gradients [128] have also been used [120, 121].

Appearance models can also be learned from tracklets. The main assumption of tracklets is that all detections in a tracklet belong to the same object, and [120–122, 124, 129, 130] exploit this assumption to learn more discriminative appearance models. Note that the “identity” in our work is different from [121], which utilized person re-identification techniques to improve the appearance model. We, however, focus on the “real-world identity” of the person, which is acquired from face-recognition.

In this work, we utilized color histograms combined with manifold learning to perform tracking. Manifold learning has also been utilized in previous work such as [131, 132] to learn subspaces for appearance features that can better differentiate the tracked target from other targets or background in single object or multi-object tracking settings.
However, the multi-object tracking performed in [132] utilized multiple independent particle filters, which may have the issue of one particle filter “hijacking” the tracking target of another particle filter [133, 134]. Therefore, to fix this issue, our method has the mutual exclusion and spatial locality constraint encoded in the optimization framework, which jointly optimizes for all trajectories to acquire a potentially more reasonable set of trajectories.

### 3.1.3 Motion Models

Objects usually move in a smooth manner, and effective motion models can capture this assumption to better model the likely movement of objects. [2, 108, 109, 112, 113] use the bounded velocity model to model motion: given the current location of the object, the location in the next frame is constrained by the maximum velocity of the object. [111, 118, 126] improve upon this by modeling motion with the constant velocity model, which is able to model the smoothness of the object’s velocity change. Higher order methods such as spline-based methods [115, 117] and the Hankel matrix [135] can model even more sophisticated motions. [124] assumes that different objects in the same scene move in similar but potentially non-linear ways, and the motion of highly confident tracklets can be used to infer the motion of non-confident tracklets.

### 3.1.4 Data Association

A data association algorithm takes the object location hypotheses, appearance model and motion model as input to find a disjoint grouping of the object location hypotheses which best describes the motion of objects in the scene. Intuitively, the data association algorithm will decide whether to place two object location hypotheses in the same group based on their affinity, which is computed from the appearance and motion models.

For the association between multiple frames, there are two popular formulations: the Hungarian algorithm and the network flow, which are both Integer Linear Programs (ILP) with a special form. Given the pair-wise affinities, the Hungarian algorithm can find the optimal bipartite matching between two sets of object location hypotheses in polynomial time [117, 119–121, 123]. In the network flow formulation [108, 110, 112, 113], each path from source to sink corresponds to the trajectory of an object. The network flow problem can also be solved optimally in polynomial time, but this formulation and the Hungarian algorithm formulation makes a number of assumptions. The first assumption is that each physical object can only be associated with one location hypothesis at each time instant to enforce the constraint that an object cannot be at multiple places at the same time. Therefore, in multi-camera environments, location
hypotheses from multiple cameras need to be consolidated first with methods such as POM [107] before these methods can be used. The second assumption is that the cost function of each trajectory can be decomposed into a series of products (or additions) of pair-wise terms [117]. Therefore, most network flow-based methods are limited to the bounded velocity model, i.e. velocity from the previous time instant is not taken into account. [118, 127, 136] have improved on this by taking into account three location hypotheses at once, so the constant velocity model can be utilized. However, this comes at the cost of using more complex algorithms such as Lagrangian Relaxation [118] or using a Linear Program solver to approximately solve an ILP [127], where finding the global optimum is no longer guaranteed. Another method to incorporate velocity in such a framework is to utilize tracklets as the basic unit of location hypotheses [119].

Many trackers have been formulated as a general Integer Linear Programming (ILP) problem. [109, 125, 127] solved the ILP by relaxing the integral constraints to continuous constraints and optimizing a Linear Program, where the solution can be computed efficiently. A subsequent branch-and-cut method to find the global optimal to the ILP [127] or a simple rounding step [109] is used to acquire a final discrete solution. [137, 138] formulated tracking as clique partitioning, which can also be formulated as an ILP problem and solved by a heuristic clique merging method. [135] formulated tracking as a General Linear Assignment ILP problem, which was approximately solved with a deterministic annealing “softassign” algorithm [139].

More complex data association methods have also been used, including continuous energy minimization [114], discrete-continuous optimization [115], Block-ICM [117], conditional random fields [116, 122], generalized minimum clique [111] and quadratic programming [126, 140].

Even though each data association method has different merits, many of the aforementioned methods do not utilize actual person identity information such as face recognition, and in many cases it is non-trivial to incorporate the identity information into the previously proposed data association frameworks. One quick way to incorporate identity information may be to assign identities to trajectories after the trajectories have been computed. However, problems occur if two different identities are assigned to the same trajectory, and the true identity of the trajectory is no longer clear. Another approach may be to follow [107] and utilize the Viterbi algorithm to find a trajectory which passes through all the identity observations of each person. However, Viterbi search cannot be performed simultaneously over all individuals, and [107] proposed to perform Viterbi search sequentially, i.e. one individual after another. This greedy approach can lead to “hijacking” of another person’s trajectory [107], which is not ideal. Therefore, to achieve effective identity-aware tracking, it is more ideal to specially design a data association
framework which can directly incorporate identity information into the optimization process.

Identity-Aware Data Association

Previously proposed data association methods [109, 141], [142] and [2] utilize identity information for tracking. There have been other work which utilizes transcripts from TV shows to perform face recognition and identity-aware face tracking [143, 144], but this is not the main focus of our work.

[109, 141] formulated identity-aware tracking as an ILP and utilized person identification information from numbers written on an athlete’s jersey or from face recognition. Results show that the method is very effective in tracking basketball and soccer players, even when there are many occlusions. [109, 141] depends on the global appearance term for assigning identities to detections. However, the global term assumes a fixed appearance template for an object, which may not be applicable in surveillance scenes recorded over many hours as the appearance of the same person may change drastically.

[142] utilizes online structured learning to learn a target-specific model, which is used to compute the edge weights in a network flow framework. Though [142] has a stronger appearance model to compensate for drawbacks of network flow methods, it utilizes densely-sampled windows instead of person bounding boxes as input, which may be too time-consuming to compute in hundreds of hours long video sequences.

The work in [2] shows that a semi-supervised tracker which utilizes face-recognition as sparse label information for each class/individual achieves good tracking performance in a complex indoor environment. However, [2] does not incorporate the spatial locality constraint during the optimization step. Without the constraint, the solution acquired from the optimization step might show a person being in multiple places at the same time, thus this method does not work well for crowded scenes. Also, the method needs a Viterbi search to compute the final trajectories. The Viterbi search requires the start and end locations of all trajectories, which is an unrealistically restrictive assumption for long-term tracking scenarios. We enhance this tracker by adding the spatial-locality constraint term, which enables tracking in crowded scenes and also removes the need for the start and end locations of a trajectory.
Multi-Object Tracking with Face Recognition

Figure 3.2: Illustration of the input and output of our tracking algorithm. Each person detection is a point in the \((x, y, t)\) space. We assume that people walk on the ground plane, so the \(z\) axis is irrelevant. The figures are drawn based on the person detections from the \textit{terrace1} data set [107].

3.2 Tracker Overview

Tracking-by-detection-based multi-object tracking can be viewed as a constrained clustering problem as shown in Figure 3.2. Each location hypothesis, which is a person detection result, can be viewed as a point in the spatial-temporal space, and our goal is to group the points so that the points in the same cluster belong to a single trajectory. A trajectory should follow the mutual exclusion constraint and spatial-locality constraint, which are defined as follows.

- **Mutual Exclusion Constraint**: a person detection result can only belong to at most one trajectory.
- **Spatial-Locality Constraint**: two person detection results belonging to a single trajectory should be reachable with reasonable velocity, i.e. a person cannot be in two places at the same time. This is an instantiation of the spatial-temporal smoothness constraint.

Sparse label information acquired from sources such as face recognition are used to assign real-world identities and also enhance tracking performance.

To compute the trajectories, our tracking algorithm has three main steps.

1. **Manifold construction based on appearance and spatial affinity**: The appearance and spatial affinity respectively assumes that 1) similar looking person detections are likely to be of the same individual and 2) person detections which are spatially and temporally very close to each other are also likely to be of the same individual.
2. **Spatial locality constraint**: This constraint encodes the fact that a person cannot be at multiple places at the same time. In contrast to the manifold created in the
previous step which encodes the similarity of two person detections, this constraint encodes the dissimilarity of two person detections.

3. Data association: Two different optimization approaches: nonnegative matrix optimization (NMO) and solution path algorithm (SPA), were proposed to acquire a solution which simultaneously satisfies the manifold assumption, the mutual exclusion constraint and the spatial-locality constraint.

In the following sections, we first define our notations, then the 3 aforementioned steps are detailed.

3.3 Notations

Given a matrix $A$, let $A_{ij}$ denote the element on the $i$-th row and $j$-th column of $A$. Let $A_i$ denote the $i$-th row of $A$. Let $a_i$ denote the $i$-th element of vector $a$. $Tr(\cdot)$ denotes the trace operator. $|\cdot|_F$ is the Frobenius norm of a matrix. Given an integer $m$, $1_m \in \mathbb{R}^m$ is a column vector with all ones.

Hereafter, we call a person detection result as an observation. Suppose the person detector detects $n$ observations. Let $c$ be the number of tracked individuals, which can be determined by either a pre-defined gallery of faces or the number of unique individuals identified by the face recognition algorithm. Let $F \in \mathbb{R}^{n \times c}$ be the label assignment matrix of all the observations. Without loss of generality, it is assumed that the observations are reorganized such that the observations from the same class are put together. The $j$-th column of $F$ is given by:

$$F_{sj} = \begin{bmatrix} 0, \ldots, 0, 1, 1, \ldots, 0 \end{bmatrix}^T, \quad \sum_{i=1}^{j-1} m(i) \leq F_{sj} \leq \sum_{i=j+1}^{c} m(i)$$

(3.1)

where $m(j)$ is the number of observations in the $j$-th class. If the $p$-th element in $F_{sj}$, i.e. $F_{pj}$ is 1, it indicates that the $p$-th observation corresponds to the $j$-th person. According to Equation 3.1, it can be verified that

$$F^TF = \begin{bmatrix} F_{s1}^T \\
\vdots \\
F_{sc}^T \end{bmatrix} \begin{bmatrix} F_{s1} & \ldots & F_{sc} \end{bmatrix} = \text{diag}\left( \begin{bmatrix} m(1) \\
\vdots \\
 m(c) \end{bmatrix} \right) = J,$n

(3.2)

where $J \in \mathbb{R}^{c \times c}$. The $i$-th observation is described by a $d$ dimensional color histogram $x_{(i)} \in \mathbb{R}^d$, frame number $t_{(i)}$, and 3D location $p_{(i)} \in \mathbb{R}^3$ which corresponds to the 3D location of the bottom center of the bounding box. In most cases, people walk on the
ground plane, and the $z$ component becomes irrelevant. However, our method is not constrained to only tracking people on the ground plane.

### 3.4 Manifold Construction based on Appearance and Spatial Affinity

There are two aspects we would like to capture with manifold learning: 1) appearance affinity and 2) spatial affinity, which will be detailed in the following sections.

#### 3.4.1 Modeling Appearance Affinity

Appearance affinity assumes that if two observations are similar in appearance, then it is very likely that the two points correspond to the same person. This assumption can be captured with manifold learning, which is usually done in a two step process. First, suitable nearest neighbors for each observation are found. The assumption is that the nearest neighbors are highly likely to be of the same class as the current observation. Second, the nearest neighbor information of each point are used to encode the manifold structure into the Laplacian matrix.

Given an observation, suitable nearest neighbors are other similar-looking observations which are spatially and temporally “nearby”. More specifically, for the $i$-th observation, let the set of spatial-temporal neighbors be $\mathcal{M}_{(i)}$. $\mathcal{M}_{(i)}$ contains observations which are not only less than $T$ frames away from the point, but also reachable from location $p_{(i)}$ with reasonable velocity. This is another instantiation of the spatial-temporal smoothness constraint. To avoid edge cases in computing velocity, we define velocity between observations $i$ and $l$ as follows:

$$v_{(il)} = \frac{\max (\|p_{(i)} - p_{(l)}\|_2 - \delta, 0)}{|t_{(i)} - t_{(l)}| + \epsilon}.$$ (3.3)
$\epsilon$ is a small number to avoid division by zero. $\delta$ models the maximum localization error of the same person from different cameras due to calibration and person detection errors, so when $t_{(i)} = t_{(l)}$, if the two data points are less than $\delta$ apart, these two points are still spatial-temporal neighbors. Therefore, $\mathcal{M}_{(i)}$ is defined as follows:

$$
\mathcal{M}_{(i)} = \{ l \mid v_{(il)} \leq V, |t_{(i)} - t_{(l)}| \leq T, 1 \leq l \leq n \},
$$

where $V$ is the maximum possible velocity of a moving person. If the velocity required to move between two points is too large, then the two points cannot be of the same individual. Given $\mathcal{M}_{(i)}$, we look for data points in the set which have color histograms similar to data point $i$, as it is likely these points will belong to the same physical individual. To compute the similarity between two color histograms $H_i$ and $H_j$, the exponential-$\chi^2$ metric is used:

$$
\chi^2(H_i, H_j) = \exp \left( -\frac{1}{2} \sum_{l=1}^{d} \frac{(H_{il} - H_{jl})^2}{H_{il} + H_{jl}} \right),
$$

Based on Equation 3.5, two color histograms are similar only if their similarity is above a certain threshold $\gamma$. Finally, the set of nearest neighbors for data point $i$ is found by selecting the top $k$ nearest neighbors in $\mathcal{M}_{(i)}$ which have a similarity score larger than $\gamma$. We denote this set as $N_{(i)} \subset \mathcal{M}_{(i)}$.

This method of finding neighbors makes our tracker more robust to occlusions. Occlusions may cause the tracking target to be partially or completely occluded. However, the tracking target usually reappears after a few frames. Therefore, instead of trying to explicitly model occlusions, our system tries to connect the observations of the tracking target before and after the occlusion. As shown in Figure 3.3, despite heavy occlusions in a time segment, the algorithm can still link the correct detections after the occlusion. The window size $T$ affects the tracker’s ability to recover from occlusions. If $T$ is too small, the method will have greater difficulty recovering from occlusions that last longer than $T$. However, a large $T$ may increase chances of linking two different objects.

Once the nearest neighbors for each data point have been computed, the manifold structure can be encoded with a Laplacian matrix as follows. We first compute the affinity matrix $W$, where $W_{ij} = \chi^2(H_i, H_j)$ if $j \in N_{(i)}$ and 0 otherwise. Then, the diagonal degree matrix $D$ of $W$ is computed, i.e. $D_{ii} = \sum_{l=1}^{n} W_{il}$. Finally, the Laplacian matrix $L$ which captures the manifold structure in the appearance space is $L = D - W$. 
3.4.2 Modeling Spatial Affinity

Other than modeling person detections of similar appearance, person detections which are “very close” (e.g. a few centimeters apart) in the same or neighboring frames are also very likely to belong to the same person. This assumption is reasonable in a multi-camera scenario because multiple detections will correspond to the same person, and due to calibration and person detection errors, not all detections will be projected to the exact same location. In this case, regardless of the appearance difference which may be resulting from non-color-calibrated cameras, these detections should belong to the same person. We therefore encode this information with another Laplacian matrix $K \in \mathbb{R}^{n \times n}$ defined as follows. Let $K^{(i)}_l$ be the set of observations which are less than distance $\tilde{D}$ away and less than $\tilde{T}$ frames away, i.e.,

$$K^{(i)}_l = \left\{ l \mid \| p^{(i)}_l - p^{(i)}_l \|_2 \leq \tilde{D}, |t^{(i)}_l - t^{(i)}_l| \leq \tilde{T}, 1 \leq l \leq n \right\}. \quad (3.6)$$

We compute the affinity matrix $A \in \mathbb{R}^{n \times n}$ from $K^{(i)}_l$ by setting $A_{ij} = 1$ if $j \in K^{(i)}_l$ and $A_{ij} = 0$ otherwise. Define $\tilde{D} \in \mathbb{R}^{n \times n}$ as a diagonal matrix where $\tilde{D}_{ii}$ is the sum of $A$’s $i$-th row. Following [145], the normalized Laplacian matrix is computed: $K = I - \tilde{D}^{-\frac{1}{2}}A\tilde{D}^{-\frac{1}{2}}$. Compared to computing the appearance affinity, the spatial affinity does not take into account appearance at all, thus the parameters $\tilde{D}$ and $\tilde{T}$ are all set very conservatively to avoid connecting to person detections from different individuals. For example, in our experiments $\tilde{D} = 20$ centimeters and $\tilde{T} = 6$ frames, while $T = 240$ frames.

Then the loss function which combines the appearance and spatial affinity is as follows:

$$\min_{F} Tr \left( F^T (L + K) F \right)$$

$$s.t. \text{ columns of } F \text{ satisfy Equation 3.1}, \forall i \in \mathcal{Y}, F_i = Y_i. \quad (3.7)$$

Minimizing the loss term will result in a labeling which follows the manifold structure specified by appearance and spatial affinity. The first term in the constraint specifies that the label assignment matrix $F$ should be binary and have a single 1 per row. The second term in the constraints of the loss function is the face recognition constraint. Face recognition information is recorded in $Y \in \mathbb{R}^{n \times c}$, where $Y_{ij} = 1$ if the $i$-th data point belongs to class $j$, i.e. the face of data point $i$ is recognized as person $j$. $Y_{ij} = 0$ if we do not have any label information. There should only be at most a single 1 in each row of $Y$. $\mathcal{Y} = \{ i \mid \exists j s.t. Y_{ij} = 1 \}$ are all rows of $Y$ which have a non-zero element (i.e. a recognized face). As face recognition is approaching human-level performance [146], it is in most cases reasonable to treat it as a hard constraint. Experiments analyzing the effect of face recognition errors on tracking performance are detailed in Section 3.9.7.
Figure 3.4: Example of synthetic tracking output with and without the spatial locality constraint. Without the spatial locality constraint (e.g., [2]), the green trajectory takes over the blue trajectory at around time 80. The green trajectory is at two places at the same time, which is wrong. With the spatial locality constraint, the results are correct.

3.5 Spatial Locality Constraint

A person cannot be in multiple places at the same time. A tracker which cannot model this constraint, such as [2], might unreasonably state that a person is in multiple places at the same time. Figure 3.4 shows an example of results with and without the constraint. The results computed with the spatial locality constraint is much more realistic. We incorporate the spatial locality constraint into our tracker by modeling pairwise person detection constraints. Given a pair of person detections \((i, j)\), if the speed \(v_{(ij)}\), which is defined in Equation 3.3, required to move from one person detection to the other is too large, then it is highly unlikely that the pair of person detections will belong to the same person. We denote \(T = \{(i, j) \mid v_{(ij)} > V\}\) as all the person detection pairs which are unlikely to be of the same individual. Then the updated loss function is as follows.

\[
\min_{F} Tr\left(F^T (L + K) F\right) \\
\text{s.t.} \quad \forall (i, j) \in T, \ F_{il} F_{jl} = 0, \ 1 \leq l \leq c \\
\text{& columns of } F \text{ satisfy Equation } 3.1, \ \forall i \in \mathcal{Y}, F_i = Y_i,
\]

(3.8)

where the term \(F_{il} F_{jl} = 0\) models the spatial locality constraint.

Note that our spatial-locality constraint is a generalization to what is used in many single-camera multi-object network flow-based trackers [112, 113], where a person not being at two places at the same time is enforced by assuming that a trajectory can only be assigned to a single person detection at one time instant. However, in a multi-camera scenario, it is often the case that multiple detections from different cameras will correspond to the same individual, and the assumption used by network flow-based trackers may not be applicable here. Therefore, we propose a more general spatial-locality constraint, which can handle the case where multiple detections from multiple
cameras all correspond to the same individual. In the current formulation, we did not explicitly model the fact that two detections from the same frame cannot belong to the same person, which could be easily added to our method. Also, experiments show that our method already achieves competitive results without this additional constraint, demonstrating the effectiveness of our spatial locality constraint.

Also note that the purpose of the affinity-based Laplacian matrix $L$ and $K$ are completely opposite of the purpose of $S$. $L$ and $K$ specifies which two data points should be in the same cluster, while $S$ enforces the must-not-link constraint, i.e. these two points cannot be in the same cluster. Though both $L$ and $S$ utilize the assumption that a person cannot be at multiple places at the same time, these two matrices have completely different purpose in the loss function.

The loss function shown in Equation 3.8 strives to find a suitable labeling of observations along the manifold while not violating both constraints: the mutual exclusion constraint and the spatial locality constraint. However, Equation 3.8 is a combinatorial problem as the values of $F$ are limited to zeros and ones. This is very difficult to solve and certain relaxation is necessary to efficiently solve the objective function. Therefore, two different optimization strategies are presented to solve Equation 3.8: nonnegative matrix optimization (NMO) and the solution path algorithm (SPA).

### 3.6 Nonegative Matrix Optimization

The first optimization methodology utilizing nonnegative matrix optimization techniques is detailed. We relax the form of $F$ such that the values are continuous, but to enforce the mutual exclusion constraint, certain constraints still need to be enforced. We first observe that according to Equation 3.2, the columns of $F$ are orthogonal to each other, i.e. $F^T F = J$ is a diagonal matrix. Also, according to the definition of $F$, $F$ is nonnegative. Furthermore, according to [147], if both the orthogonal and nonnegative constraints are satisfied for a matrix, then there will only be at most one non-zero entry in each row of the matrix, which is still sufficient in discretizing $F$ and identifying the class-membership of each observation, i.e. the mutual exclusion constraint still holds. Therefore, we relax the form of $F$, which originally is a binary label-assignment matrix, by only keeping the column orthogonal and nonnegative constraint. This leads to solving Equation 3.9.

$$
\begin{align*}
\min_{F} & \quad \text{Tr} \left( F^T (L + K) F \right) \\
\text{s.t.} & \quad \forall (i,j) \in T, F_{il} F_{jl} = 0, 1 \leq l \leq c, \\
& \quad F^T F = J, F \geq 0, \forall i \in Y, F_i = Y_i,
\end{align*}
$$

(3.9)
where the mutual exclusion constraint is enforced by $F^T F = J$ and $F \geq 0$. Under these constraints, the values in $F$ are continuous and no longer binary, but there will still only be at most one non-zero entry per row. One big advantage of this relaxation is that now our method can naturally handle false positive detections, because $F$ is now also allowed to have a row where all elements are zeros, which corresponds to a person detection not being assigned to any class.

For convenience, we organize the spatial locality constraint in a more concise form. We aggregate all the person detection pairs in $T$ and encode it in the matrix $\tilde{S}$, as shown in Equation 3.10.

$$\tilde{S}_{ij} = \begin{cases} 0 & \text{if } v(\text{ij}) \leq V \\ 1 & \text{otherwise} \end{cases}, \quad 1 \leq i, j \leq n, \quad (3.10)$$

where $V$ is the maximum possible velocity of a moving person. $\tilde{S}$ is defined so that if none of the person detection velocity constraints were violated, then $F^T \tilde{S}F = 0$, where $F_{ij}$ is the label assignment vector (column vector of $F$) for the $j$-th person. We gather this constraint for all individuals and obtain $Tr(F^T SF) = 0$ if none of the constraints were violated. The scale of $\tilde{S}$ is normalized to facilitate the subsequent optimization step. Let $D'$ be a diagonal matrix where $D'_{ii}$ is the sum of row $i$ of $\tilde{S}$, then the normalized $S = D'^{-\frac{1}{2}} \tilde{S} D'^{-\frac{1}{2}}$. Finally, we update Equation 3.9 to Equation 3.11.

$$\begin{array}{ll}
\min_{F} & Tr (F^T (L + K) F) \\
\text{s.t.} & Tr(F^T SF) = 0, \quad F^T F = J, F \geq 0, \forall i \in Y, F_i = Y_i. 
\end{array} \quad (3.11)$$

The constraint $F^T F = J$ is a difficult constraint to optimize. If $J$ is the identity matrix, then $F^T F = I$ forms the Stiefel manifold [148]. Though a few different methods have been proposed to perform optimization with the orthogonal constraint [148–151], many methods require a specific form of the objective function for the optimization process to converge. Therefore, we instead employ the simple yet effective quadratic penalty method [147, 152] to optimize the loss function. The quadratic penalty method incorporates the equality constraints into the loss function by adding a quadratic constraint violation error for each equality constraint. The amount of violation is scaled by a weight $\tau$, which gradually increases as more iterations of the optimization are performed, thus forcing the optimization process to satisfy the constraints. More details on the convergence properties of the quadratic penalty method can be found in [152]. To solve Equation 3.11, we move the constraints $F^T F = J$ and $Tr(F^T SF) = 0$ into the loss function as a penalty term. We rewrite the objective function as follows:

$$\begin{array}{ll}
\min_{F} & f(F) = \min_{F} Tr (F^T (L + K + \tau S) F) + \tau ||F^T F - J||_F^2 \\
\text{s.t.} & F \geq 0, \forall i \in Y, F_i = Y_i. 
\end{array} \quad (3.12)$$
For each $\tau$, we minimize Equation 3.12 until convergence. Once converged, $\tau$ is multiplied by 2 and Equation 3.12 is minimized again.

To solve for Equation 3.12 given a fixed $\tau$, we perform projected nonnegative gradient descent [153], which iteratively updates the solution at iteration $l$ ($F^{(l)}$) to $F^{(l+1)}$ as follows:

$$F^{(l+1)} = P \left[ F^{(l)} - \alpha^{(l)} \nabla f(F^{(l)}) \right]$$  \hspace{1cm} (3.13)

where the projection function $P$:

$$P[F_{ij}] = \begin{cases} F_{ij} & \text{if } F_{ij} > 0 \\ 0 & \text{otherwise} \end{cases},$$  \hspace{1cm} (3.14)

is an element-wise function which maps an element back to the feasible region, i.e. in this case a negative number to zero. The step size $\alpha^{(l)}$ is found in a line search-like fashion, where we search for an $\alpha^{(l)}$ which provides sufficient decrease in the function value:

$$f(F^{(l+1)}) - f(F^{(l)}) \leq \sigma \text{Tr} \left( \nabla f(F^{(l)})^T (F^{(l+1)} - F^{(l)}) \right).$$  \hspace{1cm} (3.15)

Following [153], $\sigma = 0.01$ in our experiments. The gradient of our loss function $f$ is

$$\nabla f(F) = 2 \left( L + K + \tau S \right) F + 4 \tau F (F^T F - J).$$  \hspace{1cm} (3.16)

Details on convergence guarantees are shown in [153]. To satisfy the face recognition constraints, the values of $F$ for the rows in $\mathcal{Y}$ are set according to $Y$ and never updated by the gradient.

The main advantage of projected nonnegative gradient descent over the popular multiplicative updates for nonnegative matrix factorization [148, 154] is that elements with zero values will have the opportunity to be non-zero in later iterations. However, for multiplicative updates, zero values will always stay zero. In our scenario, this means that if $F^{(l)}_{ij}$ shrinks to 0 at iteration $l$ in the optimization process, the decision that “observation $i$ is not individual $j$” is final and cannot be changed, which is not ideal. The projected nonnegative gradient descent method does not have this issue as the updates are additive and not multiplicative.

$J$ is a diagonal matrix, where each element on the diagonal $J_{ii}$ corresponds to the number of data points belonging to class $i$, i.e. $m_i$. As $m_i$ is unknown beforehand, $m_i$ is estimated by the number of recognized faces belonging to class $i$ plus a constant $\beta$, which is proportional to the number of data points $n$. In our experiments we set $\beta = \frac{n}{1000}$.

To initialize NMO, we temporarily ignore the mutual exclusion and spatial locality constraint and only use the manifold and face recognition information to find the initial
Data: Location hypothesis $p_{(i)}$, $t_{(i)}$, and appearance $x_{(i)}$, $1 \leq i \leq n$. Face recognition matrix $Y \in \mathbb{R}^{n \times c}$.

Result: Final label assignment matrix $F$

Compute Laplacian matrices $L$, $K$; // Sec. 3.4
Compute spatial locality matrix $S$; // Sec. 3.5
Compute diagonal matrix $J$; // Sec. 3.6
Compute diagonal matrix $U$ from $Y$; // Sec. 3.6
Initialize $F^{(0)}$ with Equation 3.17;
$l \leftarrow 0$; // iteration count
$\tau \leftarrow 10^{-4}$; // initial penalty
repeat // Solve for Equation 3.12 with penalty method
  $\tau \leftarrow \tau \times 2$; // gradually increase penalty $\tau$
  repeat // projected gradient descent
    Compute $F^{(l+1)}$ from $F^{(l)}$ with Equation 3.13;
    $l \leftarrow l + 1$;
  until convergence;
until $\tau \geq 10^{11}$;
return $F^{(l)}$

Algorithm 1: Main steps in nonnegative matrix optimization tracking algorithm.

value $F^{(0)}$. $F^{(0)}$ is obtained by minimizing Equation 3.17.

$$\min_{F^{(0)}} Tr \left( (F^{(0)})^T(L + K)F^{(0)} + (F^{(0)} - Y)^T(U(F^{(0)} - Y)) \right).$$

$U \in \mathbb{R}^{n \times n}$ is a diagonal matrix. $U_{ii} = \infty$ (a large constant) if $i \in Y$, i.e. the $i$-th observation is a ground truth positive for any class. Otherwise $U_{ii} = 1$. $U$ is used to enforce the consistency between prediction results and face recognition label information. The global optimal solution for Equation 3.17 is $F^{(0)} = (L + K + U)^{-1}UY$ [155].

Finally, once the optimization is complete, we acquire a $F$ which satisfies the mutual exclusion and spatial locality constraint. Therefore, trajectories can be computed by simply connecting neighboring observations belonging to the same class. At one time instant, if there are multiple detections assigned to a person, which is common in multi-camera scenarios, then the weighted average location is computed. The weights are based on the scores in the final solution of $F$. A simple filtering process is also utilized to remove sporadic predictions. In sum, the main steps of our NMO tracker are shown in Algorithm 1.

3.7 Solution Path Algorithm Optimization

We present a different method to perform optimization of Equation 3.8. The label matrix $F$ is also relaxed so that the values are continuous, but different constraints are utilized
to enforce the mutual exclusion and spatial locality constraint. The mutual exclusion constraint signifies that each row of $F$ can have at most one non-zero value. The spatial locality constraint specifies that for $(i, j) \in T$, $F_{il}F_{jl} = 0$, or in other words there can only be at most one non-zero value in the vector $[F_{il}, F_{jl}]$. These constraints can be modeled with $\ell_0$ norm constraints. Given a vector $x \in \mathbb{R}^b$, the $\ell_p$ norm of $x$ is defined as $\|x\|_p = \left( \sum_{l=1}^b x_l^p \right)^{1/p}$. Then the constraint that the $i$-th row of $F$ can have at most one non-zero value can be modeled as $\|F_i\|_0 \leq 1$. Similarly, for the spatial locality constraint, the constraint can be modeled as $\| [F_{il}, F_{jl}] \|_0 \leq 1$. Therefore, we update Equation 3.8 to acquire the following equation.

$$
\min_F \text{Tr} \left( F^T (L + K) F \right) \\
\text{s.t.} \forall i \in \mathcal{Y}, F_i = Y_i \\
\|F_r\|_0 \leq 1, 1 \leq r \leq n \\
\forall (i, j) \in T, \left\| [F_{il}, F_{jl}] \right\|_0 \leq 1, 1 \leq l \leq c.
$$

(3.18)

The three constraints in the loss function corresponds to face recognition, mutual exclusion and spatial locality constraints. Note that this formulation can also handle false positive detections, because $\|F_r\|_0 \leq 1$ allows all elements in row $r$ to be zero, i.e. this person detection does not belong to any class.

However, due to the $\ell_0$ norm constraints, Equation 3.18 is still difficult to optimize. One naïve fix is to relax the $\ell_0$ norm constraints to $\ell_1$ norm constraints. This makes the problem convex and easy to solve. However, the relaxation comes at the price that the mutual exclusion and spatial locality constraint will no longer hold in the final solution, which may cause degradation of tracking performance. Therefore, we utilized the solution path algorithm to bridge the solution between the $\ell_1$ norm and $\ell_0$ norm constraints.

Unlike traditional algorithms which only looks for the solution under a single parameter setting, the solution path algorithm finds the solution under many different parameters. This is done efficiently by utilizing the solution from the previous parameter setting as the initial value for the next parameter setting. The solution path algorithm has been successfully utilized in the machine learning community on problems such as the Lasso [156]. In the Lasso case\(^1\), the algorithm in [156] can solve for the solution of the Lasso for all regularization parameters $\lambda \in [0, \infty]$ [157]. The solution for all the $\lambda$’s are solved sequentially as the algorithm gradually increases the value of $\lambda$. The solution for $\lambda^{(t)}$ from the $t$-th iteration is initialized with the solution acquired with $\lambda^{(t-1)}$ from the $(t-1)$-th iteration. In this way, since $\lambda^{(t-1)}$ and $\lambda^{(t)}$ have similar values, the algorithm

\(^1\)The loss function of Lasso can be written as $\min_w \frac{1}{2} \|y - Xw\|_2^2 + \lambda \|w\|_1$, where $y \in \mathbb{R}^m$ is the label, $X \in \mathbb{R}^{m \times d}$ is the feature matrix, and $w \in \mathbb{R}^d$ is the weight vector to be learned.
(A) Feasible region of 3 dimensional $\|v\|_p \leq 1$, $v \geq 0$ constraint as $p$ decreases from 1, 0.5 to 0.2.

(b) Under 10 dimensional $\|v\|_p \leq 1$ constraints, average distribution of randomly sampled vectors where the values of each vector are sorted in descending order.

**Figure 3.5:** Images showing that as $p$ decreases, at most 1 non-zero value can remain, thus enforcing the mutual exclusion and spatial-locality constraints.

will converge faster to the new solution compared to random initialization. In sum, the intuition is that if the previous and current parameter settings do not differ by too much, the solution from optimizing with the previous parameter setting can be used as a good initial starting point for the next parameter setting.

In our proposed method, we utilize the solution path algorithm to compute the solution for Equation 3.18. The solution path algorithm acts like a bridge between the two solutions: one with $p = 1$ norm constraints, and the other with $p = 0$ norm constraints. Starting from the solution under the convex $\ell_1$ norm constraints, the algorithm successively solves the same optimization problem but under different $\ell_p$ norm constraints, where $p$ gradually decreases from 1 to 0. The solution path ends as $p$ approaches to 0, and a better solution to our original problem can be obtained. The bridge itself, which consists of the solutions under different $\ell_p$ norm constraints, is the solution path of the whole optimization process.

More specifically, we denote $p^{(m)}$ as the $p$ used during the $m$-th iteration of the path algorithm. Then $p^{(1)} = 1$, $p^{(M)} \to 0$, and $p^{(m-1)} > p^{(m)}$ for $2 \leq m \leq M$. At each $p^{(m)}$, our system computes the solution $F^{(m)}$ of the loss function in Equation 3.18, but under $\ell_{p^{(m)}}$ norm constraints instead. The solution $F^{(m-1)}$ acquired using $\ell_{p^{(m-1)}}$ norm constraints is used to initialize the optimization process under $\ell_{p^{(m)}}$ norm constraints. The solution $F^{(1)}$ under $\ell_1$ norm constraints is solved directly, because the Equation 3.18 under $\ell_1$ norm constraints is convex and easy to solve. We denote set $\mathcal{F} = \{F^{(1)}, F^{(2)}, \ldots, F^{(M)}\}$ as the solution path.
The physical meaning of gradually shrinking $p$ is depicted in Figure 3.5. When $p$ is still large, the tracker is not very strict on the mutual exclusion and spatial locality constraint. However, as $p$ decreases, the $\ell_0$ norm constraints are gradually satisfied as at most one value stays non-zero.

In each iteration $m$ of the solution path algorithm, we need to optimize Equation 3.18 under a fixed $p^{(m)}$ to acquire $F^{(m)}$, and block coordinate descent is utilized.

### 3.7.1 Block Coordinate Descent

In each iteration $m$ of the solution path algorithm, we need to solve Equation 3.18 with a fixed $p^{(m)} \in [0, 1]$ to acquire $F^{(m)}$. Initialized with $F^{(m-1)}$, block coordinate descent [158] was utilized to solve Equation 3.18. Block coordinate descent only updates variables of a single observation $i$ while keeping variables of all other observations fixed. The updating of a single observation $i$ is as follows. For notational clarity, we denote $G = F^{(m)}$ and $p$ refers to $p^{(m)}$ in this section. We denote the $i$-th row of $G$ as $G_i = [G_{i1}, \ldots, G_{ic}]$. Since all other observations are kept fixed, the loss function for $G_i$ is simplified from Equation 3.18 to the following quadratic function.

$$
\min_{G_i} \frac{1}{2} \|G_i - a\|_2^2 \text{ s.t. } \|G_i\|_p \leq 1,
$$

$$
\forall (i, j) \in T, \|\begin{bmatrix} G_{il} & G_{jl} \end{bmatrix}\|_p \leq 1, 1 \leq l \leq c,
$$

(3.19)

where $a \in \mathbb{R}^c$. Each element $l$ in $a$ is computed as follows.

$$
a_l = \frac{\sum_{j=1}^n (W_{ij} + A_{ij})G_{jl}}{\sum_{j=1}^n (W_{ij} + A_{ij})}.
$$

(3.20)

$a$ encodes the label information of the neighbors of observation $i$. $W$ and $A$ are the similarity matrices defined in Section 3.4. If there were no constraints, the optimal value of the loss function will be $G_i = a$, which strives for consistent labeling of an observation and its neighbors. If $G_i$ contains a recognized face, then row $i$ is not updated.

The advantage of block coordinate descent is that non-neighboring observations can be solved in parallel, which makes the optimization process efficient.

To solve Equation 3.19, we notice that the spatial locality constraint only consists of two numbers $G_{il}$ and $G_{jl}$, where $G_{jl}$ is fixed if only observation $i$ is optimized. Therefore, the constraint $\|\begin{bmatrix} G_{il} & G_{jl} \end{bmatrix}\|_0 \leq 1$ can be converted to the linear constraint $G_{il} \leq (1 - G_{jl}^p)^{\frac{1}{p}}$, where, for convenience, $u \in \mathbb{R}^c$ represents the upper bound of each element in $G_i$. However, with this simplification, Equation 3.19 is still difficult to solve due to the $\ell_p$ norm constraint from the mutual exclusion constraint. Therefore the
optimization process has two steps. We first clip the values in \( a \) to be at most \( u \) and get the clipped vector \( a' \). Then the following simplified equation is optimized.

\[
\min_{G_i} \|G_i - a'\|_2^2 \quad \text{s.t.} \quad \|G_i\|_p \leq 1.
\] (3.21)

If \( \|a'\|_p \leq 1 \), then the solution of \( G_i = a' \). Otherwise, Equation 3.21 can be solved by iteratively projecting \( a' \) onto the region \( \|G_i\|_p \leq 1 \).

### 3.7.2 Iterative Projection Method

We propose a iterative projection method to solve Equation 3.21 when \( \|a'\|_p > 1 \). For convenience, the region which satisfies \( \{ v \mid \|v\|_p \leq 1 \} \) is refered to as the \( \ell_p \) norm ball. Since, \( \|a'\|_p > 1 \), it is clear that all local optimal \( G_i^* \) are on the boundary of the \( \ell_p \) norm ball. We denote \( \pi^* \) as the normal of the tangent plane of the \( \ell_p \) norm ball at \( G_i^* \). According to the KKT conditions [158], a (local) optimal point \( G_i^* \) should have the following property: \( \pi^* \) is parallel to the gradient direction of the quadratic function \( \|G_i - a'\|_2^2 \) at \( G_i^* \). Therefore, we propose the following iterative projection method to find a local optimal as shown in Figure 3.6.

To initialize the algorithm, given \( a' \) which has \( \|a'\|_p > 1 \), we first draw a line between \( a' \) and the origin. Let \( G_i^{(1)} \in \mathbb{R}^c \) be the place where the line intersects the boundary of the \( \ell_p \) norm ball, i.e. \( \|G_i^{(1)}\|_p = 1 \). \( G_i^{(1)} \) serves as the initialization. This intersection can be found efficiently with binary search. Now, given a \( G_i^{(l-1)} \) from iteration \( l - 1 \), we compute \( G_i^{(l)} \) for iteration \( l \) as follows. We first compute the tangent plane \( \pi^{(l-1)} \) of the \( \ell_p \) norm ball at \( G_i^{(l-1)} \). Next \( a' \) is projected onto \( \pi^{(l-1)} \) and the projection is denoted as \( x^{(l-1)} \). We then draw a line between \( a' \) and \( x^{(l-1)} \) and find the intersection of the line and the \( \ell_p \) norm ball. The intersection is denoted as \( G_i^{(l)} \), which is the value of \( G_i \) for iteration \( l \). These steps are repeated till convergence. The loss function monotonically decreases with such an update rule, which is proved in Appendix A.
Data: Location hypothesis $p(i), t(i)$, and appearance $x(i)$, $1 \leq i \leq n$. Face recognition matrix $Y \in \mathbb{R}^{n \times c}$, $p(1), \ldots, p(M)$

Result: solution path $\mathcal{F}$

Compute Laplacian matrices $L$, $K$; // Sec. 3.4

// Sol. of convex $\ell_1$ norm constraint as initialization

$\mathcal{F} \leftarrow \{ \mathbf{F}^{(1)} \}$; // Solution path set

$m \leftarrow 2$; // path algorithm iteration count

repeat // Solution path algorithm, Sec. 3.7

$\mathbf{G} \leftarrow \mathbf{F}^{(m-1)}$ // Initialize with $\mathbf{F}^{(m-1)}$

// Solve Equation 3.18 under $\ell_{p(m)}$ norm constraint

repeat // Block coordinate descent, Sec. 3.7.1

for $\forall$ observations $i$ do

| Update $\mathbf{G}_i$ by solving Equation 3.19 |

end

until convergence;

$\mathbf{F}^{(m)} \leftarrow \mathbf{G}$;

Add $\mathbf{F}^{(m)}$ into $\mathcal{F}$;

$m \leftarrow m + 1$;

until $m \leq M$;

return $\mathcal{F}$

Algorithm 2: Solution path tracking algorithm. Note that $1 = p^{(1)} > \cdots > p^{(M)} = 0.01$.

Finally, once the solution from Equation 3.18 is acquired, we can compute the trajectories from the solution the same way it was computed for the nonnegative matrix optimization method.

The steps of solution path tracking algorithm are summarized in Algorithm 2, and the time complexity is as follows. We optimize Equation 3.18 by updating $\mathbf{G}_i$ with Equation 3.19 for each of the $n$ observations. As proved in Appendix A, the iterative projection algorithm requires $\mathcal{O} (c(k + q + \log(c)))$ per iteration, where $q$ is the average number of constraints per observation, and $k$ is the number of nearest neighbors of each observation. Thus, solving Equation 3.18 is $\mathcal{O} (nc(k + q + \log(c)) \ast \text{MaxIter})$, which is efficient as it is approximately linear in the number of observations ($n$), classes ($c$), nearest neighbors ($k$), and constraints ($q$).

The solution path algorithm has two key benefits. First, it can optimize loss functions with $\ell_0$ norm constraints, which is a common but difficult-to-optimize constraint in sparsity and relaxed combinatorial problems. Second, the solution path can be viewed as the “decision making process” of the tracker, which can be utilized to automatically pinpoint uncertainty in tracking for manual correction in an active learning scenario [159]. More details are in Section 3.9.8.
3.8 Comparing Nonnegative Matrix Optimization and Solution Path Algorithm

The two proposed methods: NMO and SPA, all solve the same general loss function: Equation 3.8. The main difference is in how the mutual exclusion constraints were enforced to solve the loss function. For SPA, the mutual exclusion is modeled by $\|F_i\|_0 \leq 1$, $1 \leq i \leq n$, which is precisely the mutual exclusion constraint: each row can have at most one non-zero value. For NMO, mutual exclusion is achieved by the constraints $F^T F = J$ and $F \geq 0$, which is more complex than the constraints of SPA. For SPA, elements on different rows of $F$ are decoupled. However, for NMO, $F^T F = J$ means that elements on column $i$ have to sum to $J_{ii}$, thus coupling all elements of $F$ in the same column. In general, having more complex constraints such as the constraints for NMO often lead to more complex optimization landscapes, thus causing the algorithm to more likely enter a non-ideal local minima. So we would expect that SPA to perform better than NMO in the tracking task as SPA models the constraints in a simpler and more precise form than NMO.

3.9 Experiments and Results

3.9.1 Data Sets

As our goal is identity-aware tracking, we only focused on tracking sequences in which identity information such as face recognition was available. Therefore, many popular tracking sequences such as the PETS 2009 sequences [160], Virat [161], TRECVID 2008 [162] and Town Centre [163] were not applicable as the faces in these sequences were too small to be recognized. The following four data sets were utilized in our experiments.

**terrace1**: The 4 camera terrace1 [107] data set has 9 people walking around in a 7.5m by 11m area for 5000 frames under 25fps, which corresponds to a total of around 13 minutes of video. The scene is very crowded, thus putting the spatial locality constraint to test. The POM grid computed had width and height of 25 centimeters per cell. Person detections were extracted for every frame. As the resolution of the video is low, one person did not have a recognizable face. For the sake of performing identity-aware tracking on this dataset, we manually added two identity annotations for each individual at the start and end of the person’s trajectory to guarantee that each individual had identity labels. None of the trackers utilized the fact that these two additional annotations were the start and end of a trajectory. In total, there were 794 identity labels out of 57,202 person detections.
**Caremedia 6m:** The 15 camera *Caremedia 6m* [2, 3] data set has 13 individuals performing daily activities in a 15 camera nursing home scene for 6 minutes 17 seconds, which corresponds to a total of around 94 minutes of video. The data set was first used in [2]. Manual annotations of people standing or walking in the scene were provided every second and further interpolated to every frame. Sitting people were not annotated. The 15 surveillance cameras are setup on the ceilings of the public areas in a nursing home as shown in Figure 3.7. There are many occlusions caused by walls which are typical in indoor scenes. There are also many challenging scenes such as long corridors with sparse camera setups, where considerable occlusion was observed. There is also no single camera which has a global view of the whole environment, which is typical in many surveillance camera setups, but atypical in the data sets that have been used to perform multi-camera tracking. The data set records activities in a nursing home where staff maintain the nursing home and assist residents throughout the day. As the data set covers a larger area and is also longer than *terrace1*, we ran into memory issues for trackers which take POM as input when our cell size was 25 centimeters. Therefore, the POM grid computed in our experiments had width and height of 40 centimeters per cell. Person detections were extracted from every sixth frame. In total, there were 2,808 recognized faces out of 12,129 person detections. Though on average there was a face for every 4 person detections, but recognized faces were usually found in clusters and not evenly spread out over time. So there were still long periods of time when no faces were recognized.

**Caremedia 8h:** The 15 camera *Caremedia 8h* data set [164] has 49 individuals performing daily activities in the same nursing home as *Caremedia 6m*. There are a total of 116.25 hours of video, which corresponds to 7 hour 45 minutes wall time. The videos were record on 2005/10/06 from 13:15 to 21:00. Ground truth of standing or walking...
people was annotated every minute. Sitting people were not annotated. Person detections were extracted from every sixth frame. In total, there were 70,994 recognized faces out of 402,833 person detections.

**Caremedia 23d**: The 15 camera Caremedia 23d data set consists of nursing home recordings spanning over a 26 day window: 2005/10/06 to 2005/10/31, with 3 days missing (10/13, 10/14, 10/22) due to hardware issues. For 2005/10/06, the recordings are the same as Caremedia 8h. For 2005/10/31, the recordings starts at 6am and ends at 12pm. For the remaining 21 days, the recordings were from 6am to 9pm. This leads to a total of 4,935 hours of video. To the best of our knowledge, this is the longest sequence to date to be utilized for multi-object tracking experiments, thus enabling us to evaluate tracking algorithms in realistic long-term tracking scenarios. Caremedia 23d has 65 individuals performing daily activities in the same nursing home as Caremedia 6m. To make the annotation process feasible, ground truth of standing or walking people was annotated every 30 minutes, and annotations were only performed on individuals who have been identified in the first 7 days of the data set. Sitting people were not annotated. Person detections were extracted at every sixth frame. In total, there were 3.1 million recognized faces out of 17.8 million person detections.

### 3.9.2 Baselines

We compared our method with three identity-aware tracking baselines. As discussed in Section 3.1, it is non-trivial to modify a non-identity-aware tracker to incorporate identity information. Therefore, other trackers which did not have the ability to incorporate identity information were not compared.

**Multi-Commodity Network Flow (MCNF)**: The MCNF tracker [109] can be viewed as an extension of the K-Shortest-Path tracker (KSP, [108]) with identity aware capabilities. The KSP is a network flow-based method that utilizes localization information based on POM. Given the POM localizations, a network flow graph is formed. The algorithm will then find the $K$ shortest paths to the graph, which correspond to the $K$ most likely trajectories in the scene. MCNF further duplicates the graph in KSP for every different identity group in the scene. The problem is solved with linear programming plus an additional step of rounding non-integral values.

Following [109], we reimplemented the MCNF tracker. In our experiments, the graph is duplicated $c$ times, because for our setup each individual belongs to its own identity group. Gurobi [165] was used as our linear program solver. Global appearance templates of each person were computed from the appearance of person detections which had recognized faces. Occlusions were computed from a raw probability occupancy map, and
occluded observations were not used to generate templates nor compare color histograms. Following [109], the input of MCNF was taken from the output of POM and KSP to save computation time. The source code of POM and KSP were from the authors [107, 108]. For trajectories which came closer to three grid cells, the cells in between the two trajectories were also activated so that the MCNF had the freedom to switch trajectories if necessary. This setting is referred to as MCNF w/ POM. The base cost of generating a trajectory, which is a parameter that controls the minimum length of the generated tracks, is set to -185 for all MCNF w/ POM experiments.

For the two Caremedia data sets, we also took the person detection (PD) output and generated POM-like localization results which were also provided to MCNF. The POM-like localization results were generated by first creating a grid for the nursing home scene, and then aggregating all person detections falling into each grid at each time instant. This setting is referred to as MCNF w/ PD. For all MCNF w/ PD experiments, the grid size is 40 centimeters, the base cost of generating a trajectory is -60, and detections were aggregated over a time span of 6 frames to prevent broken trajectories. For the Caremedia 8h and Caremedia 23d set, the Gurobi solver was run in 12,000 frame batches to avoid memory issues.

**Lagrangian Relaxation (LR):** [142] utilizes LR to impose mutual exclusion constraints for identity-aware tracking in a network flow framework very similar to MCNF, where each identity has their own identity specific edges. Lagrange multipliers enforce the mutual exclusion constraint over mutual-exclusive edges in the graph. To optimize with LR, dynamic programming first finds trajectories for each identity given the current network weights. Then, the Lagrange multipliers, which are a part of the network weights, are updated to penalize observations that violate the mutual exclusion constraint. This process is repeated again on the updated network weights till convergence. To fairly compare different data association methods, our LR-based tracker utilizes the same appearance information used by all our other trackers, thus the structured learning and densely sampled windows proposed in [142] were not used. Specifically, LR uses the same POM-like input and network as MCNF.

**Non-Negative Discretization (NND):** The Non-Negative Discretization tracker [2] is a primitive version of our proposed tracker. The two main differences are: 1) NND does not have the spatial locality constraint, thus an extra Viterbi trajectory formulation step is necessary, which requires the start and end of trajectories, and 2) a multiplicative update was used to perform non-negative matrix factorization. NND requires the start and end locations of trajectories, which are usually not available in real world scenarios. In our experiments, therefore, no start and end locations were provided to NND, and the final trajectories of NND were formed with the same method used by our proposed
Multi-Object Tracking with Face Recognition

tracker. NND utilizes [155] to build the manifold, but internal experiments have shown that utilizing the method in [155] to build the Laplacian matrix achieves similar tracking performance compared to the standard method [145, 166]. Therefore, to fairly compare the two data association methods, we utilized the same Laplacian matrix computation method for NND and our method. Also the spatial affinity term $K$ was not used in the originally proposed NND, but for fairness the $K$ term was added to NND.

3.9.3 Implementation Details

For person detection, we utilized the person detection model from [4, 167]. The person detection results from different camera views were mapped to a common 3D coordinate system using the camera calibration and ground plane parameters provided. Color histograms for the person detection were computed the same way as in [2]. HSV color histograms were used [105, 168]. We split the bounding box horizontally into regions and computed the color histogram for each region similar to the spatial pyramid matching technique [169]. Given $L$ layers, there are $2^L - 1$ partitions for each template. $L$ was 3 in our experiments. Since the person detector only detects upright people, tracking was not performed on sitting people or residents in wheelchairs. For POM, background subtraction was performed with [170].

For our proposed methods: nonnegative matrix optimization (NMO) and solution path algorithm (SPA), the parameters for all four data sets were as follows. The number of nearest neighbors used for appearance-based manifold construction was $k = 25$. The window to search for appearance-based nearest neighbors was $T = 8$ seconds. The fastest velocity one could walk was $V = 3$ m/s. The similarity threshold for looking for nearest neighbors was $\gamma = 0.85$. The maximum localization error was $\delta = 125$ to take into account camera calibration errors. For modeling spatial affinity, $\hat{D}$ was 20 centimeters, and $\hat{T}$ was 6 frames. When computing the spatial locality constraint, we found that computing the velocity between all pairs of data points will generate too many constraints, thus only conflicting pairs of data points which were less than 6 frames apart were used. The above parameters were also used for NND. For NMO, the initial value of $\tau = 2 \times 10^{-4}$, and the final value was $\tau = 10^{11}$. For SPA, the step size for $p$ was 1.05, i.e. $p^{(m+1)} \leftarrow p^{(m)}/1.05$, and there were a total of $M = 94$ different $p$ values. $p^{(M)} = 0.01$.

For Caremedia 6m, Caremedia 8h, and Caremedia 23d, a preprocessing step to remove incorrect person detection is performed. Person detections with a width/height ratio less than 0.4 (i.e. a “fatter” person detection, which could be only the upper half of a person) were removed.
Face Recognition

Face information is acquired from the PittPatt software\(^2\), which can recognize a face when a person is close enough to the camera. We assumed that the gallery for the persons-of-interest is provided. There are two options to collect this gallery: 1) manually collect faces of the person or 2) perform face clustering over all detected faces and select clusters which consist of faces corresponding to the person-of-interest. Though the selection requires some manual effort, it does not consume a lot of time as the majority of faces have already been clustered. The latter option was utilized to create the Caremedia 6m and Caremedia 8h tracking sequences. As the PittPatt clusters are already very clean, and also the clusters were manually mapped, the face recognition is very accurate. Manual verification on Caremedia 6m showed 98% accuracy on face recognition.

For the Caremedia 23d set, it is too tedious for humans to map clusters into each individual. Therefore, we first manually mapped clusters on 7 days (10/06 to 10/12) of video. Then, based on these mapped clusters, the PittPatt tool is again utilized to perform face recognition on the remaining 16 days of video (10/15 to 10/31, excluding 10/22). Manual verification on the recordings on 2005/10/31 showed that purely automatic face recognition achieves 75\% accuracy, which was still reasonable. More details on face recognition performance and its effect on tracking performance are detailed in Section 4.3.3.

For Caremedia 6m, Caremedia 8h, and Caremedia 23d, a preprocessing step to remove face recognitions that are highly likely to be incorrect is performed. First, for each individual, all its person detections with recognized faces are aggregated. Then, for each person detection, the distance between all other aggregated person detections were computed, and the median distance is extracted. Finally, the person detections with median distances in the top 10\% are filtered out, i.e. the person detections which look most dis-similar with the other person detections belonging to the same individual are filtered out.

3.9.4 Evaluation Metrics

Identity-aware tracking can be evaluated from a multi-object tracking point of view and a classification point of view. From the tracking point of view, the most commonly used multi-object tracking metric is Multiple Object Tracking Accuracy (MOTA\(^3\)) \cite{171,172}. Following the evaluation method used in \cite{2,115}, the association between the tracking

\(^2\)Pittsburgh Pattern Recognition (http://www.pittpatt.com)
\(^3\)Code modified from http://www.micc.unifi.it/lisanti/source-code/.
results and the ground truth is computed in 3D with a hit/miss threshold of 1 meter. MOTA takes into account the number of true positives (TP), false positives (FP), missed detections (false negatives, FN) and identity switches (ID-S). Following the setting in [109] MOTA is computed as follows:

\[ \text{MOTA} = 1 - \frac{\# \text{ FP} + \# \text{ FN} + \log_{10}(\# \text{ ID-S})}{\# \text{ ground truth}}. \] (3.22)

However, the TP count in MOTA does not take into account the identity of a person, which is unreasonable for identity aware tracking. Therefore, we compute identity-aware true positives (I-TP), which means that a detection is only a true positive if 1) it is less than 1 meter from the ground-truth and 2) the identities match. Similarly, we can compute I-FP and I-MD, which enables us to compute classification-based metrics such as micro-precision (\( \text{MP} = \frac{\# \text{ I-TP}}{\# \text{ I-TP} + \# \text{ I-FP}} \)), micro-recall (\( \text{MR} = \frac{\# \text{ I-TP}}{\# \text{ I-TP} + \# \text{ I-FN}} \)) and a comprehensive micro-F1 (\( \frac{2 \times \text{MP} \times \text{MR}}{\text{MP} + \text{MR}} \)) for each tracker. The micro-based performance evaluation takes into account the length (in terms of time) of each person’s trajectory, so a person who appears more often has larger influence to the final scores.

### 3.9.5 Tracking Results

Tracking results for the four data sets are shown in Table 3.1. As we are more interested in identity-aware tracking, we pay more attention to the F1-score from the classification-based metrics, which will only be high if both precision and recall are high. Our proposed methods: NMO and SPA may not always be the best in terms of MOTA scores, but we achieve the best performance in F1-scores across all four data sets. This means that our tracker can not only track a person well, but also accurately identify the individual. Figure 3.9 and Figure 3.10 show some qualitative examples of our tracking result. The performance difference between NMO and SPA is not as large, but SPA often achieves a slightly better F1 score, which reflects that SPA tends to find better local minima than NMO due to its simpler row-wise constraints for mutual exclusion in contrast to the orthogonality constraint use by NMO.

The importance of the spatial locality constraint (SLC) is also shown in Table 3.1a. Without the spatial locality constraint in the optimization step (NND and NMO w/o SLC), performance degrades significantly in *terrace1*, where the tracking scene is extremely crowded and the appearance feature is not very discriminative as most people

---

4There are two common transformation functions (denoted as \( c_s() \) in [172]) for the identity-switch term, either \( \log_{10} \) [109, 172] or the identity function [171]. We have selected the former as this is what was used in MCNF, which is one of our baselines.
<table>
<thead>
<tr>
<th>Method</th>
<th>Micro-Precision</th>
<th>Micro-Recall</th>
<th>Micro-F1</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
<th>ID-S</th>
<th>MOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face only</td>
<td>0.493</td>
<td>0.018</td>
<td>0.035</td>
<td>646</td>
<td>24708</td>
<td>284</td>
<td>5</td>
<td>0.014</td>
</tr>
<tr>
<td>KSP w/ POM</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>22182</td>
<td>2990</td>
<td>767</td>
<td>187</td>
<td>0.852</td>
</tr>
<tr>
<td>MCNF w/ POM</td>
<td>0.503</td>
<td>0.532</td>
<td>0.561</td>
<td>21864</td>
<td>3298</td>
<td>644</td>
<td>197</td>
<td>0.844</td>
</tr>
<tr>
<td>LR w/ POM</td>
<td>0.609</td>
<td>0.478</td>
<td>0.535</td>
<td>19216</td>
<td>5996</td>
<td>521</td>
<td>147</td>
<td>0.743</td>
</tr>
<tr>
<td>NND</td>
<td>0.613</td>
<td>0.238</td>
<td>0.343</td>
<td>8035</td>
<td>17267</td>
<td>1771</td>
<td>57</td>
<td>0.249</td>
</tr>
<tr>
<td>NMO w/o SLC</td>
<td>0.704</td>
<td>0.346</td>
<td>0.464</td>
<td>10642</td>
<td>14655</td>
<td>1745</td>
<td>62</td>
<td>0.353</td>
</tr>
<tr>
<td>NMO</td>
<td>0.692</td>
<td>0.635</td>
<td>0.663</td>
<td>21370</td>
<td>3873</td>
<td>1783</td>
<td>116</td>
<td>0.777</td>
</tr>
<tr>
<td>SPA</td>
<td>0.752</td>
<td>0.705</td>
<td>0.727</td>
<td>22263</td>
<td>2906</td>
<td>1407</td>
<td>100</td>
<td>0.826</td>
</tr>
</tbody>
</table>

(a) Tracking performance on terrace1 sequence.

<table>
<thead>
<tr>
<th>Method</th>
<th>Micro-Precision</th>
<th>Micro-Recall</th>
<th>Micro-F1</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
<th>ID-S</th>
<th>MOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face only</td>
<td>0.942</td>
<td>0.362</td>
<td>0.523</td>
<td>12369</td>
<td>21641</td>
<td>727</td>
<td>9</td>
<td>0.342</td>
</tr>
<tr>
<td>KSP w/ POM</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>21286</td>
<td>11794</td>
<td>36035</td>
<td>939</td>
<td>-0.406</td>
</tr>
<tr>
<td>MCNF w/ POM</td>
<td>0.117</td>
<td>0.238</td>
<td>0.157</td>
<td>23493</td>
<td>9769</td>
<td>44452</td>
<td>757</td>
<td>-0.594</td>
</tr>
<tr>
<td>LR w/ PD</td>
<td>0.746</td>
<td>0.578</td>
<td>0.625</td>
<td>19941</td>
<td>13749</td>
<td>5927</td>
<td>329</td>
<td>0.422</td>
</tr>
<tr>
<td>NND</td>
<td>0.861</td>
<td>0.726</td>
<td>0.787</td>
<td>25628</td>
<td>8364</td>
<td>3100</td>
<td>27</td>
<td>0.663</td>
</tr>
<tr>
<td>NMO w/o SLC</td>
<td>0.889</td>
<td>0.726</td>
<td>0.791</td>
<td>25578</td>
<td>8408</td>
<td>3080</td>
<td>33</td>
<td>0.662</td>
</tr>
<tr>
<td>NMO</td>
<td>0.865</td>
<td>0.755</td>
<td>0.807</td>
<td>26384</td>
<td>7576</td>
<td>3537</td>
<td>59</td>
<td>0.673</td>
</tr>
<tr>
<td>SPA</td>
<td>0.871</td>
<td>0.735</td>
<td>0.809</td>
<td>26531</td>
<td>7458</td>
<td>3004</td>
<td>30</td>
<td>0.692</td>
</tr>
</tbody>
</table>

(b) Tracking performance on Caremedia 6m sequence.

<table>
<thead>
<tr>
<th>Method</th>
<th>Micro-Precision</th>
<th>Micro-Recall</th>
<th>Micro-F1</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
<th>ID-S</th>
<th>MOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face only</td>
<td>0.858</td>
<td>0.256</td>
<td>0.394</td>
<td>164</td>
<td>471</td>
<td>19</td>
<td>2</td>
<td>0.230</td>
</tr>
<tr>
<td>MCNF with PD</td>
<td>0.743</td>
<td>0.418</td>
<td>0.535</td>
<td>265</td>
<td>347</td>
<td>71</td>
<td>25</td>
<td>0.342</td>
</tr>
<tr>
<td>LR with PD</td>
<td>0.787</td>
<td>0.405</td>
<td>0.535</td>
<td>261</td>
<td>360</td>
<td>52</td>
<td>16</td>
<td>0.351</td>
</tr>
<tr>
<td>NND</td>
<td>0.588</td>
<td>0.505</td>
<td>0.543</td>
<td>314</td>
<td>281</td>
<td>174</td>
<td>42</td>
<td>0.283</td>
</tr>
<tr>
<td>NMO w/o SLC</td>
<td>0.638</td>
<td>0.549</td>
<td>0.590</td>
<td>349</td>
<td>257</td>
<td>151</td>
<td>31</td>
<td>0.357</td>
</tr>
<tr>
<td>NMO</td>
<td>0.648</td>
<td>0.571</td>
<td>0.607</td>
<td>370</td>
<td>241</td>
<td>149</td>
<td>26</td>
<td>0.386</td>
</tr>
<tr>
<td>SPA</td>
<td>0.650</td>
<td>0.581</td>
<td>0.614</td>
<td>375</td>
<td>236</td>
<td>152</td>
<td>26</td>
<td>0.389</td>
</tr>
</tbody>
</table>

(c) Tracking performance on Caremedia 8h sequence.

<table>
<thead>
<tr>
<th>Method</th>
<th>Micro-Precision</th>
<th>Micro-Recall</th>
<th>Micro-F1</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
<th>ID-S</th>
<th>MOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face only</td>
<td>0.819</td>
<td>0.199</td>
<td>0.154</td>
<td>127</td>
<td>512</td>
<td>28</td>
<td>2</td>
<td>0.154</td>
</tr>
<tr>
<td>MCNF with PD</td>
<td>0.712</td>
<td>0.355</td>
<td>0.474</td>
<td>227</td>
<td>412</td>
<td>92</td>
<td>22</td>
<td>0.209</td>
</tr>
<tr>
<td>LR with PD</td>
<td>0.663</td>
<td>0.357</td>
<td>0.464</td>
<td>228</td>
<td>411</td>
<td>116</td>
<td>13</td>
<td>0.174</td>
</tr>
<tr>
<td>NMO</td>
<td>0.698</td>
<td>0.532</td>
<td>0.604</td>
<td>340</td>
<td>299</td>
<td>147</td>
<td>14</td>
<td>0.300</td>
</tr>
<tr>
<td>SPA</td>
<td>0.663</td>
<td>0.556</td>
<td>0.605</td>
<td>355</td>
<td>273</td>
<td>158</td>
<td>18</td>
<td>0.326</td>
</tr>
</tbody>
</table>

(d) Tracking performance on Caremedia 23d sequence.

Table 3.1: Tracking performance of each method on 4 data sets. POM: Probabilistic Occupancy Map proposed in [107] as input. PD: Person detection as input. SLC: Spatial locality constraint. “w/” and “w/o” are shorthand for “with” and “without” respectively. We did not perform the MCNF w/ POM on the Caremedia 8h and Caremedia 23h sequences as it was already performing poorly on the shorter sequence.
wear dark clothing. On the other hand, the performance drop on the Caremedia sequences is minimal, which could be because of 1) the Caremedia sequence is not as crowded as terrace1, and 2) people in Caremedia sequences have less similar appearance, thus purely relying on color histograms already provide very good performance. We believe both explanations are valid, and more analysis on the second explanation is in Section 4.3.

The MCNF tracker is also a very strong baseline. For terrace1, KSP and consequently MCNF achieve very good MOTA results with POM person localization. MCNF is slightly worse than KSP on MOTA scores because though MCNF is initialized by KSP, MCNF is no longer solving a problem with a global optimal solution. However, for the Caremedia 6m sequence, results are poor. Through manual analysis, we found that POM, which is used for person localization in KSP and MCNF, has more difficulty in localizing on long corridors where cameras only view the principal direction of the corridor. For example, when there are multiple people on the corridor, their foreground masks tend to merge together into a single blob. Thus there will be many different ways the generative POM model can synthesize the foreground image for each camera view. This leads to ambiguities in person localization, which will significantly hurt tracking performance. Cameras with a side-view on the corridor will significantly alleviate this issue, but this is usually not available in a corridor setting. Also, the indoor scene of Caremedia 6m is more complex than terrace1. Therefore, even though there are 15 cameras in Caremedia 6m, occlusions caused by walls mean that the camera coverage is not as perfect as terrace1, thus causing more ambiguities in POM localization. Lastly, there were other non-person moving objects such as carts and rolling closets in the scene, which would also be detected by POM. These ambiguities cause false positives, leading to poor KSP performance. As MCNF input is based on KSP output, MCNF w/ POM also performs poorly. However, this is unfair, as MCNF performed poorly due to inaccurate localization, which is unrelated to its data association method. Therefore, to fairly compare MCNF, we provided MCNF with the same localization information used by our method and ran the MCNF w/ PD experiment. With the new localization based on person detections, MCNF was able to achieve competitive results. This experiment shows that even if a tracker performs poorly, it may simply be due to a single malfunctioning component in the tracker, and if the component is switched with another effective component, the tracker can still achieve good results. We believe this is a fairer way of comparing different trackers. Nevertheless, with the same person detection as input, our method still outperforms MCNF in all sequences in terms of F1-score.

The performance of Face only clearly shows the contribution of face recognition and tracking. For the Caremedia related sequences, face recognition can already achieve certain performance, but our tracker further improves this performance by roughly 15%
Figure 3.8: Tracking performance on 
terrace1 data set under different step sizes and
$\ell_p$ norm constraints.

Figure 3.9: Snapshots of tracking results from the 4 camera 
terrace1 sequence using
SPA.

absolute, which is still very significant. Furthermore, for terrace1, there are very limited
faces, thus face recognition plays a very minor role in tracking performance. However,
we are still able to achieve reasonable performance, which shows that our tracker is
effective as most of the heavy-lifting is performed by the tracking algorithm.

For SPA, we also tested the performance under different step sizes $s$, i.e. $p^{(m+1)} \leftarrow p^{(m)}/s$
for varying values of $p$ on the terrace1 sequence as shown in Figure 3.8. We can see that
smaller step size leads to better MOTA scores, which shows that directly optimizing
under $\ell_0$ norm constraints (very large step size) may converge to bad local minimum.
Also, there is a big performance drop if one only solved the tracking problem under $\ell_1$
norm constraints, which only achieves MOTA 0.378. However, by using the solution
path algorithm to compute the solution under $\ell_0$ norm constraints, SPA was able to
improve MOTA to 0.83.
Figure 3.10: Snapshots of tracking results from Caremedia 8h data set using NMO. To increase readability, not all arrows are drawn and only 12 out of 15 cameras are shown.
3.9.6 Discussion - Advantages of Tracker

The key advantages of our tracker are as follows:

**Face recognition information is integrated into the framework:** Face recognition serves as a natural way to automatically assign identities to trajectories in long-term tracking scenarios, where manual intervention is prohibitively costly. When the tracker loses track of a person, face recognition can also aid in automatic reinitialization. Also, in long-term scenarios, it is common that people will change clothes, thus drastically changing their appearance. Face recognition will still be able to recognize this person, making our method robust to large appearance changes by a single person.

**Naturally handle appearance changes:** Handling appearance changes is crucial because the appearance of the tracking target can change gradually in different parts of the scene. In our tracker, the appearance templates of the tracked target are implicitly encoded in the manifold structure we learn. Therefore, if the appearance of a tracked object changes smoothly along a manifold, our algorithm can model the change. No fixed global appearance model is required to track each individual, and no threshold is required to decide when to adaptively update the appearance model. If there is a drastic change in appearance for a tracked object, then the appearance manifold will highly likely be broken as the nearest neighbor search will not select the detection where the object’s appearance changed drastically. However, the spatial affinity Laplacian matrix $K$ still can potentially link up these two observations.

**Take into account appearance from multiple neighbors:** Our tracker models appearance by taking into account the appearance information from multiple neighboring points, which enables us to have a more stable model of appearance. Linear programming and network flow-based methods can only either model appearance globally and assume the appearance of a target will not change, or model appearance similarity only over the previous and next detection in the track.

**Handle multiple detections per frame for one individual:** In multi-camera scenes, it is common that at one time instant, multiple detections from different cameras correspond to the same physical person. This phenomenon may be difficult to deal with for single-camera multi-object trackers based on network flow [112, 113], because the spatial locality constraint for these methods are enforced based on the assumption that each individual can only be assigned a single person detection per frame. Therefore, multi-camera network flow-based methods such as [108, 109] utilize a two step process where the POM is first used to aggregate evidences from multiple cameras to perform localization. Then the data association step is used to compute trajectories. The two steps are necessary so that the spatial locality constraint can be enforced for network
flow methods. In our case, our formulation of the spatial locality constraint, which is based on the velocity to travel between two detections being under a certain threshold, can be viewed as a generalization of the aforementioned assumption, and this enables us to incorporate the localization and data association steps into a single optimization framework.

**No discretization of the space required in multi-camera scenarios:** Previous multi-camera network flow methods [108, 109] requires discretization of the tracking space in multi-camera scenarios to make the computation feasible. Finer grids run into memory issues when the tracking sequence is long and covers a wide area, and coarser grids run the risk of losing precision. However, our tracker works directly on person detections, and discretization is not necessary.

### 3.9.7 Discussion - Limitations of Tracker

There are also limitations to our tracker.

**Assumes at least one face recognition per trajectory:** If there is a trajectory where no faces are observed and recognized, then our tracker will completely ignore this trajectory, which is acceptable if we are only interested in identity-aware tracking. Otherwise, one potential solution is to find clusters of unassigned person detections and assign pseudo-identities to them to recover the trajectories.

**Only bounded velocity model employed:** To employ the more sophisticated constant velocity model, we could use pairs of points as the unit of location hypotheses, but this may generate significantly more location hypotheses than the current approach.

**Assumes all cameras are calibrated:** To perform multi-camera tracking, we first map all person detections into a global coordinate system. In order to do so, the intrinsic and extrinsic parameters of the cameras need to be provided. If a camera moves, the updated extrinsic parameters also needs to be provided.

**Face recognition gallery required beforehand:** In order to track persons-of-interest, the gallery is required beforehand. This is the only manual step in our whole system, which could be alleviated when the detected faces are clustered thus making it very efficient for humans to map the face clusters to persons-of-interest. Also, in a nursing home setting, the people we are interested in tracking and observing are fixed, thus this is a one time effort which could be used for days, weeks or even months of recordings.

**Assumes perfect face recognition:** The current framework assumes perfect face recognition, which may not be applicable in all scenarios. Therefore, we also analyzed the effect of face recognition accuracy on tracking performance in Section 4.3.3.
Multi-Object Tracking with Face Recognition

(A) Stable solution path: easy observation. It is clearly ID 2.

(B) Turbulent solution path: more confusing observation. Could be ID 2 or ID 3 as they are close together.

Figure 3.11: Visualization of the decision making process of our tracker. Easy and confusing examples can be identified.

3.9.8 Analyzing the Solution Path

An advantage of the solution path is that it can be utilized to locate uncertainty in the tracker’s prediction, which is not addressed in most multi-object trackers. We emphasize that this uncertainty is different from the confidence of detected people or tracklets used by many trackers [113, 173], which focuses on the confidence of the input data. In our case we are interested in estimating the confidence of the output tracking results. [174] mentioned that the non-integer results acquired from their linear-programming-based multi-object tracker can also be interpreted as uncertainty of tracking output, but no experiments were performed in this direction.

In this section, we demonstrate how to utilize the solution path to locate potential tracking errors. The solution path records the class membership values in the label matrix $F$ for all values of $p$. If the solution path for an observation shows high scores for multiple individuals as shown in Figure 3.11b, this may indicate that the tracker is uncertain, which can be captured with the entropy measure [175]. Specifically, for iteration $m$, we denote $G_i = F_i^{(m)}$ as the score distribution for the $i$-th observation. According to the mutual exclusion constraint, $\|G_i\|_{p(m)} \leq 1$, i.e. $\sum_{j=1}^{c} G_{ij}^{p(m)} \leq 1$. Then, $G_{ij}^{p(m)}$ for all $1 \leq j \leq c$ can be viewed as a probability distribution$^5$ and compute the entropy as follows: $\epsilon_i^{(m)} = -\sum_{j=1}^{c} G_{ij}^{p(m)} \log \left( G_{ij}^{p(m)} \right)$. $\epsilon_i^{(m)}$ captures the spread of the score distribution for observation $i$ at iteration $m$. We compute the uncertainty of an observation by summing the entropy of the observation for iterations where $p \in [0.01, 0.1]$, as most fluctuations were observed in this range, i.e. $\bar{\epsilon}_i = \sum_{p(m) \in [0.01, 0.1]} \epsilon_i^{(m)}$.

$^5$The sum of $\sum_{j=1}^{c} G_{ij}^{p(m)}$ may not always be 1 if the constraint is not tight, but it is usually 1 in most cases.
There are other methods to compute uncertainty, such as computing the residual error for each observation. However, the residual error for a sample is only a single number, but our method provides richer information as the decision process for the whole solution path is taken into account. One may argue that we can also utilize the intermediate results of other optimization methods and treat them as the decision making process. However, these unconverged intermediate results do not have an obvious physical meaning. For our case, the solution path consists of converged solutions from multiple unique optimization problems. Each solution has clear physical meanings: the class membership hypothesis given the current strictness (value of $p$) of the mutual exclusion and spatial locality constraint. In sum, our entropy measure provides deeper insights to the tracking process.

To validate our entropy metric, Figure 3.12 shows the histogram of $\bar{e}_i$ for both correctly and incorrectly assigned observations. Figure 3.12 shows that incorrect observations tend to have larger entropy, thus supporting our claims. In the next section, we demonstrate the usage of the entropy measure for improving multi-object tracking in an active learning scenario.

### 3.9.9 Active Learning based on the Solution Path Algorithm

In challenging scenarios, it is inevitable for trackers to make mistakes, and it would be very useful if the tracking can pinpoint potential errors for human verification and labeling. However, human verification is very expensive, thus one should first present to the human annotators the instances which will improve the tracker the most if the labels of the instances were acquired. The task of automatically pinpointing such instances is called active learning [159]. Active learning for efficient manual refinement of tracking results has been explored in [176, 177], which utilizes a single object tracker to track multiple objects. In our case, our tracker is a multi-object tracker which can aid active learning.
A widely used heuristic in active learning is uncertainty sampling, i.e., selecting the instances of which the classifier is least certain. Uncertainty sampling is a good fit to our entropy measure, which reflects the uncertainty of our tracker’s output. To evaluate our entropy-based sampling method, we performed active learning experiments as follows. The tracker is run iteratively, and after each iteration, the tracker automatically identifies the 5 most confusing observations and requests for their labels. The additional labels are added into $Y$ and the tracker is rerun. This iteration is repeated 10 times. To select the confusing observations, three methods were utilized: sampling based on entropy values, sampling based on time difference from closest labeled instance (baseline), and sampling based on residual error (baseline). The sampling based on entropy values favors the observations with higher entropy, i.e., the probability of the $i$-th observation being selected is $\frac{1}{\text{Rank}(e_i)}$, where the rank instead of the absolute entropy values were used to favor higher ranked observations. Time difference sampling favors observations which are furthest away in terms of time from any labeled instance, thus having higher likelihood of having an identity switch. Residual error sampling favors observations which have high residual error in the final optimization result. High residual error may indicate that this observation is incorrect. During sampling, we also add a simple filter to avoid sampling all 5 observations from the same region (within 1 meter) and time (within 2 seconds). The results shown in Figure 3.13 demonstrates that our entropy-based sampling significantly beats the baselines. Time difference sampling also shows some gains, but residual sampling performs poorly because high residual error mostly occurs near observations with recognized faces. Recognized faces have a fixed score of 1, but scores of neighboring observations will be significantly smaller (e.g., 0.3 or less), thus causing large residual error, but labeling observations near recognized faces is not helpful in improving tracking. In sum, our entropy measure for uncertainty sampling is effective in active learning.
3.10 Summary

In this chapter, we proposed a multi-object tracker which can localize and identify each person at each time instant. The tracker utilizes identity information acquired from external knowledge such as face recognition to not only enhance multi-person tracking performance, but also assign a real-world identity to each tracked target. The spatial-temporal smoothness constraint was utilized for modeling appearance/spatial affinity and formulating the spatial locality constraint. We proposed two optimization methods: Nonnegative Matrix Optimization (NMO) and the Solution Path Algorithm (SPA) to solve our tracker’s loss function. Experiments show that our two methods outperform the state-of-the-art, and in general SPA outperforms NMO as it more concisely models the mutual exclusion constraint and the spatial locality constraint.

Our tracker is fully automatic except for the acquisition of the face recognition gallery, which does not require too much effort assuming the people we are observing (in the nursing home) do not change often. We further demonstrated the scalability of our method by applying our tracker on 4,935 hours of surveillance video, where our tracker is able to locate a person-of-interest around 55% of the time with 65% precision. However, in this chapter we have only explored different optimization methods to enhance tracking. The appearance features utilized are still hand-crafted color histograms. In the next chapter, we explore the possibility of utilizing deep appearance features to replace color histograms and further enhance tracking.
Chapter 4

Deep Person Re-Identification for Multi-Object Tracking

Deep learning has not only shown to be very effective in multiple vision tasks such as object recognition [5], pose estimation [178] and single object tracking [94], but it has also been shown to be effective in person re-identification (ReID) [179–181]. The task of person ReID is to classify whether a pair of person detections are of the same individual or not. This becomes very challenging when the pair of person detections were detected by different cameras, which leads to difference in viewpoints of the person and a potential mismatch in color distributions.

On the other hand, previous work have leveraged person ReID techniques to enhance multi-object tracking [121]. Handcrafted features such as HOG [128] and color histograms were utilized to compute the affinity of the appearance of person detections. However, recent work [179–181] have shown that deep features perform significantly better than handcrafted features in person ReID, which motivates us to revisit the task of utilizing person ReID techniques to enhance multi-object tracking. Therefore, in this chapter, we explore the fusion of deep person ReID methods with our proposed multi-object tracker to enhance tracking performance.

A crucial prerequisite of deep learning is to acquire enough training data. To train person ReID networks, one needs a large number of person detection pairs. Though there are public person ReID data sets [180] available, these data sets tend to 1) be limited in size and 2) have domain differences with the current data set of interest. Therefore, we propose an unsupervised method to collect large numbers of person detection pairs directly from the current data set.
To overcome the domain difference and lack of training data, we propose to utilize the spatial-temporal smoothness constraint, which is an internal constraint in video, to automatically collect person ReID training data directly from unlabeled multi-camera surveillance environment footage. The intuition is that a person cannot be at two places at the same time, thus if two separate persons were detected in a single frame, then they cannot be the same person and is a negative training example. To collect positive examples, the multi-camera setup is utilized. In a small time window, if two cameras both detect a person at the same location, then it is highly likely that the two detections correspond to the same individual thus the pair of detections can be treated as a positive sample. Based on the two aforementioned rules which could be utilized in an unsupervised fashion, we were able to utilize thousands of hours of multi-camera surveillance environment footage and collect millions of diverse yet accurate cross-view in-domain person ReID training data to train our networks, which is the key novelty of our method. Experiments show that deep person ReID combined with multi-object tracking further improves tracking performance.

In sum, the contribution of this chapter are as follows.

1. We propose an unsupervised method to collect large amounts of same-view and cross-view person ReID training data. As the method is unsupervised and can be applied directly to the scene of interest, our method is able to overcome issues such as lack of training data and domain mismatch.

2. We present experiments testing multiple deep ReID network architectures on both the person ReID task and the multi-object tracking task. Results show that deep ReID networks learned based on our collected training examples can significantly improve tracking performance.

In the following sections, Section 4.1 gives an overview of existing deep ReID networks. Section 4.2 describes our proposed training data collection method. Section 4.3 presents experimental results on multi-object tracking and Section 4.4 concludes this chapter.

### 4.1 Review of Deep Person Re-Identification Networks

To utilize deep ReID for multi-object tracking, we first survey and explore existing deep ReID architectures. There are two popular deep person ReID architectures: the siamese network [179] and the “mixed network” [180, 181] as shown in Figure 4.1. Both network architectures take a pair of person detections as input and learns a model to differentiate whether these two person detections are of the same individual or not.
Deep Person Re-Identification for Multi-Object Tracking

Figure 4.1: Diagrams of common deep person ReID architectures during training.

For the siamese network [179], the pair of images goes through the same network (Network A), which generates a vector representation for each image. The loss used is the contrastive loss [182], which tries to learn parameters such that positive samples have vector representations which are very close in Euclidean distance, and negative samples are far in Euclidean distance. Negative samples further than a threshold are not penalized.

For the mixed network, the first half of the network is also siamese, where both images go through Network B. However, the mixed network will then merge the output of the two Network B’s and pass through another Network C to get the final output. The final output is a two class classification prediction which predicts whether the current pair is a positive or negative pair. Therefore, the softmax loss is used to optimize the mixed network. The main novelty of the mixed network is in Network C, where previous work have designed specialized layers to compare the similarity of the two person detections. In [180], multiple layers such as the “patch matching” and “maxout-grouping” layers were proposed to handle geometric and photometric differences of the person detections. In [181], the “cross input neighborhood difference”, “patch summary features” and “across patch features” layers were proposed to enhance person ReID.

The biggest difference between the two architectures is that siamese networks learn a vector representation for each input image, whereas mixed networks do not. Therefore, the siamese network is ultimately limited to using the Euclidean distance to measure the similarity of two images’ vector representation. On the other hand, the mixed network can learn more sophisticated distance metrics than the siamese network, because the output of Network B goes through Network C, which can be viewed as a highly non-linear distance metric to compare the output of the two Network Bs’. This enables mixed networks to learn more sophisticated relations between person detection pairs. However, the siamese network is advantageous because learning a vector representation which is optimized under Euclidean distance enables approximate nearest neighbor techniques to be used for k nearest neighbors search. This is very useful in our tracker, as to build the manifold, we need to find the k nearest neighbors of each person detection. In order to find nearest neighbors for the mixed network, one can no longer utilize approximate nearest neighbor search and is forced to compute all pair-wise person detection
comparisons, which is in the order of $O(n^2)$ and very time consuming.

**Experiments on CUHK03**

To evaluate the effectiveness of different network architectures, experiments were ran on the current largest person ReID data set CUHK03 [180]. CUHK03 has 13,164 images from 1360 pedestrians. For each pedestrian, images from two camera views were provided. For each camera view, there were up to 5 images for the pedestrian. Following [180, 181], person ReID was evaluated as a retrieval task. 20 standardized test sets with 100 people each were defined. For a test set, each of the 100 people were used as queries. For each person/query, the model was given a single image of the person from one view, and asked to retrieve the same person in the 100 images of people from the second view. A model performed better if the queried person was found higher up in the ranked list. We aggregated the retrieval results from each of the 100 queries over 20 test sets and drew the standard Cumulative Matching Characteristic (CMC) curve, which plotted the following: for a given rank, the number of queries who had their answers found above this rank.

The network architectures and features tested were as follows:

1. HSV Color histograms: the features used for our tracker in Chapter 3.
2. Ahmed et. al. CVPR15: A reimplementation of [181]. There were two differences: 1) hard negative mining was performed by [181] but not in this experiment, and 2) data augmentation by random translation was performed by [181] but not in this experiment. These may have caused a slight drop in performance. The meta-parameters used all follow [181], where the stochastic gradient descent with initial learning rate $\eta^{(0)} = 0.01$ was used. The inverse policy: $\eta^{(i)} = \eta^{(0)}(1 + \gamma \times i)^{-p}$ with $\gamma = 10^{-4}$ and $p = 0.75$ was used to update the learning rate. $i$ denotes the current mini-batch iteration. The momentum was $\mu = 0.9$ and weight decay was $\lambda = 5 \times 10^{-4}$. The maximum number of iterations was 210K.
3. Ahmed et. al. CVPR15 with pose: An experiment testing whether pose detection [8] results will help in person ReID. An example is shown in Figure 4.2. The hypothesis is that if pose estimation can locate the body parts of the person, this could potentially help match body parts of different person detections and improve person ReID.

4. Alexnet siamese: Our self-implemented siamese network, which is based on a simplified Alexnet [5]. The network has three 5x5 convolutional layers with 96, 256 and 256 filters respectively. Each convolution layer is followed by a ReLU and a max pooling to reduce the input size by a factor of 2. This is followed by two fully connected layers with size 512. The last layer is a contrastive loss layer. For training, stochastic gradient descent with initial learning rate 0.002 was used for the first 12K iterations. Then the learning rate was dropped to 0.0004 and the network was trained for another 12K iterations. The momentum was $\mu = 0.9$ and weight decay was $\lambda = 5 \times 10^{-4}$.

5. Alexnet 6 channels: This network has the exact same network architecture as the “Alexnet siamese” network, but instead of a siamese network, this network is a mixed network without a Network B. The two 3 channel RGB input images were directly concatenated to create a 6 channel output, which is fed into our simplified Alexnet. The meta-parameters used to train the model were the same as “Alexnet siamese”.

All the models were implemented with Caffe [183]. To train each network, we generated around 30K positives and 600K negatives. The 30K positives included horizontal reflection as data augmentation. The positive negative ratio during the actual training process was 1:3.

Results are shown in Figure 4.3. Clearly, deep models performed significantly than hand-crafted HSV color histograms used by our tracking in Chapter 3. Deep models also performed more or less in the same space. “Alexnet 6 channels” is the best performer, which shows that a network with no specialized layers for person ReID could also perform very well. However, “Alexnet 6 channels” had 8 times more parameters than “Ahmed et. al. CVPR15”, which may also be a reason of its good performance. “Ahmed et. al. CVPR15 Reproduce” performed slightly worse than “Ahmed et. al. CVPR15 Original”. This may be because we did not do hard negative mining in our implementation, which was shown to be very useful in [181]. Interestingly, adding pose information did not really improve person ReID. Finally, “Alexnet siamese” performed slightly worse than “Alexnet 6 channels”, which shows that mixed networks were able to model more complex distance metrics.
In sum, our experiments show that the models we trained are on par with the state-of-the-art [181]. Also, we find that a generic network with more parameters can still perform well or sometimes even better than networks with specialized layers for person ReID. Our next step is to apply these models to multi-object tracking, but before training models, one needs to first collect in-domain person ReID training data for each data set. Therefore, in the next section we will detail our unsupervised method to collect person ReID training data.

4.2 Unsupervised Collection of Person Re-Identification Training Data

Though person ReID data sets already exist, domain discrepancies between different data sets may cause person ReID performance to drop when an out-of-domain model is utilized. Thus it would be ideal to train person ReID networks on in-domain data. However, it would be very tedious if manual annotation of training data is required for all new data sets. Therefore, we propose an unsupervised method to collect in-domain person ReID training data from multi-camera surveillance scenarios.

Based on the spatial-temporal smoothness constraint, the fundamental assumption made is that a person cannot be at multiple places at the same time. If two person detections
are very close in space and time, then it is highly likely they correspond to the same person. This criteria is used when building the spatial affinity Laplacian matrix for our tracker, but in this chapter we instead utilize the criteria to collect positive person ReID training samples. All person detection pairs which qualify the constraints of Equation 3.6 were selected as positive samples. On the other hand, if moving between two person detections exceeds the maximum speed a person can reasonably walk, then it is highly likely they are not the same individual. As this criteria is also used in our tracker, we reused the velocity equation from Equation 3.3 and the velocity constraints from Equation 3.10 to find negative samples. Intuitive examples are shown in Figure 4.4.

The key novelty of our method is extending these two assumptions to the multi-camera setting. The idea of utilizing these two assumptions are not new and have been utilized in other single-view multi-object tracking papers [120, 184]. However, to the best of our knowledge, we are the first work to apply this idea on very large amounts of multi-camera data. The multi-camera scenario enables us to automatically collect large amounts of training pairs with very large viewpoint changes, thus providing abundant data for deep networks to learn an effective cross-camera representation. Examples of collected training data are shown in Figure 4.5.

There are two limitations to our method of unsupervised collection of training data. First, our assumption: a person cannot be at multiple places at the same time, can actually fail when the surveillance cameras are not perfectly time-synchronized. In this case, incorrect negative training data, i.e. person detections of the same person but treated as negative data, could be collected. The second limitation is that we assume the surveillance cameras have significant view overlap. If this view overlap does not exist or is too small, then the system will not be able to harvest large amounts of cross-view positive examples.
4.3 Multi-Object Tracking with Appearance Features from Deep Person Re-Identification Models

In this section, we detail the tracking experiments performed to evaluate the effectiveness of the networks trained on the data collected in an unsupervised manner.

4.3.1 Experiment Setup

**Collecting positive and negative samples:** We ran the unsupervised data collection process over 3 video sets: 1) *terrace1*, 2) *Caremedia 8h* and 3) a subset of *Caremedia 23d*, which included data sampled from 12 days (10/07 - 10/18) of videos. Networks were trained on the collected data and applied on to the respective data sets. *Caremedia 6m* utilized the model trained from *Caremedia 8h*.

For each data set, at least 500K positives and 1M negatives were used to train the network. Specifically, we were able to collect 474K (same view) and 109K (different view) positives pairs from *Caremedia 8h*, which has 116.25 hours of video. Note that these numbers are before data augmentation. On the other hand, CUHK03 only has 30K positives after data augmentation such as horizontal reflection. This clearly demonstrates the power of unsupervised collection of training examples, and if we further run our algorithm over the 5000 hours of surveillance videos, tens of millions of positive pairs could be collected.

Examples of collected training data from *Caremedia 8h* are shown in Figure 4.5, which shows that intra-camera positives look very similar and not very informative. However, inter-camera positives can look very different, yet they all clearly belong to the same person. Therefore, the diverse training data potentially enables the person ReID model to really learn to perform cross-view person ReID.

**Network training:** During training, 3 different network architectures were utilized: “Ahmed et. al. CVPR15”, “Alexnet siamese”, and “Alexnet 6 channels”. The same training parameters as Section 4.1 were used to train the networks. The only minor change was that for the tracking experiments, the final layer for “Alexnet siamese” only had 252 dimensions instead of 512. This was to match the number of dimensions in the HSV color histogram, so that the comparison was fair.

**Baseline runs:** We had two baselines. The first baseline was the HSV color histogram run, which was based on a handcrafted feature and does not require any training data. The second baseline was the tracking run based on the deep ReID network trained on CUHK03, which was an out-of-domain data set with respect to Caremedia.
Figure 4.5: Example of positive and negative samples collected by our method on Caremedia 8h.
Tracking parameters: As our deep network features already utilized spatial affinity information, we further shrunk the spatial affinity parameters to $\tilde{D} = 5$ centimeters, and $\tilde{T} = 3$ frames. Also, the appearance similarity threshold $\gamma = 0.99$ to reflect the different scale of confidence output of our network. All other parameters were the same.

4.3.2 Tracking Results and Discussion

Tracking results are shown in Table 4.1. Overall, for Caremedia-based sequences, deep features provided a slight boost in performance. However, on terrace1, tracking based on deep features was able to achieve near perfect performance. This demonstrates the effectiveness of our collected training data and deep networks. More detailed discussion are provided in the next few paragraphs.

Why was there a big improvement on terrace1 but not on Caremedia 6m?

The huge improvement on terrace1 based on deep feature was mainly due to the lack of discriminative power of the HSV color histogram features for this data set. From Figure 3.9, we can see that most people in the tracking sequence wear dark clothing, which was very challenging for HSV color histogram features. However, there were still enough discriminative cues available to distinguish each person, and the deep networks were able to latch onto these cues based on the training data collected. Note that both HSV color histograms and the “Alexnet siamese” network output are 252 dimensional vector representations, but the “Alexnet siamese” network was able to enhance tracking to near perfect performance. This demonstrates that these deep features are both discriminative and compact. A further qualitative analysis of nearest neighbors found by different features are shown in Figure 4.6. We can see that deep features had the ability to generalize across cameras whereas the color histograms only found nearest neighbors from the same camera. Though cross-camera color differences could be alleviated through the spatial affinity Laplacian matrix, an appearance feature which has the ability to generalize across cameras is still significantly more powerful.

On the other hand, for the Caremedia sequences the improvement based on deep features were not as substantial. One key reason is because people in Caremedia sequences, as shown in Figure 3.10, were easily distinguished based on their color, thus deep features were not able to significantly surpass HSV color histogram features on this data set.

We further calculated the error rate of the nearest neighbors (NN) found in the appearance affinity computation step, and results are shown in Table 4.2, which shows that color histograms make significantly more error on the terrace1 sequence compared to deep features, thus the large improvement in tracking performance. The similar error rates on Caremedia 6m also support the minimal tracking improvement of deep features.
<table>
<thead>
<tr>
<th>Method</th>
<th>Micro-Precision</th>
<th>Micro-Recall</th>
<th>Micro-F1</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
<th>ID-S</th>
<th>MOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face only</td>
<td>0.493</td>
<td>0.018</td>
<td>0.035</td>
<td>646</td>
<td>24708</td>
<td>284</td>
<td>5</td>
<td>0.014</td>
</tr>
<tr>
<td>SPA w/ HSV</td>
<td>0.777</td>
<td>0.722</td>
<td>0.749</td>
<td>22587</td>
<td>2700</td>
<td>916</td>
<td>72</td>
<td>0.857</td>
</tr>
<tr>
<td>SPA w/ CUHK03</td>
<td>0.263</td>
<td>0.271</td>
<td>0.267</td>
<td>16910</td>
<td>7902</td>
<td>8678</td>
<td>547</td>
<td>0.346</td>
</tr>
<tr>
<td>Alexnet siamese</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPA w/ Ahmed et. al. CVPR15</td>
<td>0.977</td>
<td>0.962</td>
<td>0.969</td>
<td>24477</td>
<td>852</td>
<td>462</td>
<td>30</td>
<td>0.948</td>
</tr>
<tr>
<td>SPA w/ Caremedia</td>
<td>0.967</td>
<td>0.947</td>
<td>0.957</td>
<td>24250</td>
<td>1085</td>
<td>575</td>
<td>24</td>
<td>0.934</td>
</tr>
<tr>
<td>Alexnet 6 channels</td>
<td>0.982</td>
<td>0.972</td>
<td>0.977</td>
<td>24644</td>
<td>697</td>
<td>440</td>
<td>18</td>
<td>0.955</td>
</tr>
<tr>
<td>SPA w/ Caremedia</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alexnet siamese</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) Tracking performance on terrace1 sequence.

<table>
<thead>
<tr>
<th>Method</th>
<th>Micro-Precision</th>
<th>Micro-Recall</th>
<th>Micro-F1</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
<th>ID-S</th>
<th>MOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face only</td>
<td>0.942</td>
<td>0.362</td>
<td>0.523</td>
<td>12369</td>
<td>21641</td>
<td>727</td>
<td>9</td>
<td>0.342</td>
</tr>
<tr>
<td>SPA w/ HSV</td>
<td>0.880</td>
<td>0.746</td>
<td>0.808</td>
<td>26198</td>
<td>7798</td>
<td>2666</td>
<td>23</td>
<td>0.692</td>
</tr>
<tr>
<td>SPA w/ CUHK03 siamese</td>
<td>0.876</td>
<td>0.646</td>
<td>0.744</td>
<td>22429</td>
<td>11566</td>
<td>2611</td>
<td>24</td>
<td>0.583</td>
</tr>
<tr>
<td>SPA w/ Ahmed et. al. CVPR15</td>
<td>0.866</td>
<td>0.788</td>
<td>0.825</td>
<td>27040</td>
<td>6952</td>
<td>4508</td>
<td>27</td>
<td>0.663</td>
</tr>
<tr>
<td>SPA w/ Caremedia</td>
<td>0.862</td>
<td>0.789</td>
<td>0.824</td>
<td>27120</td>
<td>6880</td>
<td>4044</td>
<td>19</td>
<td>0.679</td>
</tr>
<tr>
<td>Alexnet 6 channels</td>
<td>0.914</td>
<td>0.745</td>
<td>0.821</td>
<td>25363</td>
<td>8637</td>
<td>2455</td>
<td>19</td>
<td>0.674</td>
</tr>
<tr>
<td>SPA w/ Alexnet siamese</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Tracking performance on Caremedia 6m sequence.

<table>
<thead>
<tr>
<th>Method</th>
<th>Micro-Precision</th>
<th>Micro-Recall</th>
<th>Micro-F1</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
<th>ID-S</th>
<th>MOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face only</td>
<td>0.858</td>
<td>0.256</td>
<td>0.394</td>
<td>164</td>
<td>471</td>
<td>19</td>
<td>2</td>
<td>0.230</td>
</tr>
<tr>
<td>SPA w/ HSV</td>
<td>0.650</td>
<td>0.581</td>
<td>0.614</td>
<td>375</td>
<td>236</td>
<td>152</td>
<td>26</td>
<td>0.389</td>
</tr>
<tr>
<td>SPA w/ CUHK03 siamese</td>
<td>0.648</td>
<td>0.513</td>
<td>0.573</td>
<td>334</td>
<td>287</td>
<td>143</td>
<td>16</td>
<td>0.323</td>
</tr>
<tr>
<td>SPA w/ Ahmed et. al. CVPR15</td>
<td>0.694</td>
<td>0.606</td>
<td>0.64</td>
<td>391</td>
<td>237</td>
<td>135</td>
<td>9</td>
<td>0.415</td>
</tr>
<tr>
<td>SPA w/ Alexnet 6 channels</td>
<td>0.695</td>
<td>0.606</td>
<td>0.648</td>
<td>389</td>
<td>239</td>
<td>131</td>
<td>9</td>
<td>0.418</td>
</tr>
<tr>
<td>SPA w/ Caremedia</td>
<td>0.713</td>
<td>0.589</td>
<td>0.645</td>
<td>382</td>
<td>251</td>
<td>122</td>
<td>4</td>
<td>0.414</td>
</tr>
<tr>
<td>Alexnet siamese</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) Tracking performance on Caremedia 8h sequence.

<table>
<thead>
<tr>
<th>Method</th>
<th>Micro-Precision</th>
<th>Micro-Recall</th>
<th>Micro-F1</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
<th>ID-S</th>
<th>MOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face only</td>
<td>0.819</td>
<td>0.199</td>
<td>0.154</td>
<td>127</td>
<td>512</td>
<td>28</td>
<td>2</td>
<td>0.154</td>
</tr>
<tr>
<td>SPA w/ HSV</td>
<td>0.663</td>
<td>0.556</td>
<td>0.605</td>
<td>355</td>
<td>284</td>
<td>180</td>
<td>14</td>
<td>0.272</td>
</tr>
<tr>
<td>SPA w/ Caremedia</td>
<td>0.698</td>
<td>0.573</td>
<td>0.629</td>
<td>366</td>
<td>273</td>
<td>158</td>
<td>18</td>
<td>0.324</td>
</tr>
<tr>
<td>Alexnet siamese</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(d) Tracking performance on Caremedia 23d sequence.

Table 4.1: Tracking performance with hand-crafted HSV color histograms versus deep models. For brevity, only selected trackers from Table 3.1 are shown here.
over color histograms on Caremedia 6m. Nevertheless, these results show that the deep network still has learned discriminative features to disambiguate each person in the scene based on the training data collected in an unsupervised manner.

Why is the NN error rate of deep features on Caremedia 6m higher than terrace1? Table 4.2 shows that the error rate of deep features is similar to color histograms for Caremedia 6m, and we hypothesize that this was caused by the deep network not receiving enough training data for 1) each individual person and 2) each region / camera view of the nursing home.

To validate the first hypothesis, we first computed the number of ReID training pairs harvested for each individual in Caremedia 6m and terrace1. Then the number of training pairs and the tracking F1-score of the individual were correlated as shown in Figure 4.7a. We can clearly see that 1) individuals with many training pairs tend to perform better and 2) individuals in terrace1 have more pairs than people Caremedia 6m. These observations support our first hypothesis.
To validate the second hypothesis, we first discretized the nursing home into many regions, and then computed the tracking error rate per region. The error was computed by dividing the number of incorrectly classified person detections with the total number of ground-truth in the region. The number of training pairs per region was also computed, and the statistics of each region is shown on a nursing home map in Figure 4.8. Figure 4.8 indicates that regions with higher tracking errors tend to have less training pairs. Therefore, we further correlated the number of training pairs per region with the error rate per region as shown in Figure 4.7b. Results show that regions with more training pairs tend to have lower tracking error rates. The correlation is -0.56.

A qualitative analysis to analyze the cross-camera generalization ability of the feature was further performed. The appearance features corresponding to the same individual were extracted from different camera views, and the pair-wise similarity matrices of the features are shown in Figure 4.9. We can clearly see that cross-camera generalization was successful for Figure 4.9a, which was a region (intersection of the corridor and living room) with many training pairs. However, cross-camera generalization failed completely for Figure 4.9b, which was a region (end of the corridor) with less training pairs. The above observations support our second hypothesis.

In sum, deep features were not as effective on Caremedia sequences because the scene had significantly more variability than terrace1. There were more people walking around in a larger area which was covered by more cameras. Therefore, the amount of data required to learn all the variations in the scene for all camera pairs was significantly larger. To make matters worse, the amount of training data per person is less than terrace1. Therefore, if one is to tackle these challenges, two key points are 1) make sure all regions have enough training pairs and 2) collect training pairs from as many different individuals as possible.

Cross-data set generalization capability of deep networks is limited. As shown in Table 4.1, when we applied an out-of-domain deep network trained on CUHK03 to Caremedia and terrace1, performance dropped significantly. Performance on terrace1 dropped to near random. These results act as a warning that deep networks can easily overfit to the training data, and if the testing data is not in-domain, then features acquired from deep networks can hurt performance. Therefore, this makes our unsupervised training data collection method an even more ideal candidate in providing in-domain samples to train deep networks.

No clear winner in deep network architectures. If we compared the performance of different deep network architectures, there is no clear winner. Despite the fact that siamese networks were inferior to other network architectures in the person ReID task, siamese networks performed as well as other networks on all tracking data sets. These
results are very encouraging in that siamese networks have the nice property of generating a vector representation for each person detection, thus enabling the utilization of approximate nearest neighbor techniques for efficient nearest neighbor search. This property is crucial because finding nearest neighbors is a fundamental component of our tracker (Section 3.4.1), and if we want to apply our tracker to large-scale or time critical scenarios, one needs to compute nearest neighbor search efficiently. For mixed networks such as “Ahmed et. al. CVPR15” and “Alexnet 6 channels”, comparing two person detections require a feedforward pass through the network, which is extremely time consuming and significantly slower than computing the pairwise distance of two vectors. Therefore, this is the main reason why the other two architectures: “Ahmed et. al. CVPR15” and “Alexnet 6 channels” were not run on the Caremedia 23d sequence.

**Already achieves reasonable long-term tracking performance.** Examining the
Deep Person Re-Identification for Multi-Object Tracking

Figure 4.9: Pair-wise similarity of deep features across multiple cameras. Detections from different cameras have been grouped into consecutive IDs, and if there are clear “squares” along the diagonal, then this means that the intra-camera similarity is significantly larger than inter-camera similarity, which implies that cross-camera generalization is not successful.

4.3.3 Analysis of Tracking Errors

Though our tracker was able to achieve near perfect tracking performance on terrace1, there is still large room for improvement on the Caremedia sequences. In this section, we analyze the failure cases of our tracker, which may account for the remaining gap between the current performance and perfect performance.

Detection errors/misses: This is one of the main causes of tracking errors. As our tracking algorithm is based on tracking-by-detection, a person which was not detected by the detection algorithm will not be tracked. We performed an analysis on the false negatives of Caremedia 6m, which showed that 72.5% were caused by the person detector not being able to detect the person. If the ground-truth instances that have no
corresponding detection were removed, the tracking F1-score on Caremedia 6m increases from 0.821 to 0.899, which is getting close to near perfect performance. Qualitatively, Figure 4.10 shows some examples of difficult person detection scenarios. These scenarios include difficult poses and occlusion. A possible solution to increase recall is to utilize deep models such as the Faster RNN [185], which has shown to significantly out-perform the Deformable Part-based Models we used.

**Face Recognition:** As described in Section 3.9.3, manual verification on Caremedia 6m shows that 98% of the face recognitions were correct. The high accuracy shows that face recognition errors were not the main cause of the remaining performance gap for Caremedia 6m. However, as our purely automatic face recognition performance on Caremedia 23d achieves only around 75% accuracy (Section 3.9.3), we analyzed the effect of incorrect face recognition on tracking performance. Face recognition errors at different error rates were randomly generated on the Caremedia 6m set. Errors were generated by randomly changing an already recognized face to another person’s face. Results are shown in Figure 4.11, which show that performance drops steadily as face recognition errors increase, and a 25% error (75% accuracy) lead to around 10% drop in F1, which coincides with the results on Caremedia 23d. For the first 7 days in Caremedia 23 where the face clusters were manually mapped into each person, the average F1 score was 0.699. For the remaining 16 days, where face recognition was computed completely automatically, the F1 score dropped to 0.591, which is a 10% drop.

**Camera Synchronization:** This is the other main cause of tracking errors. Due to hardware issues, the Caremedia videos were not time synchronized. To make matters worse, the video encoding process was not completely error free, and many recordings have missing frames which last up to a few seconds. Though the location of the encoding errors could be automatically found, and synchronization with audio was sometimes effective, there were many cases when the videos were still not synchronized. In the worst case, synchronization errors were up to 20 seconds, which will confuse the tracker as the same person could be at more than one place at the same time. Figure 4.12 shows an example. We hypothesize that the drop in performance between Caremedia 6m and Caremedia 8h is mainly due to this issue, because the videos in Caremedia 6m
Figure 4.11: Analyzing the effect of face recognition errors on tracking performance on Caremedia 6m. At each error rate, the experiment was repeated 3 times and the 95% confidence interval is shown.

Figure 4.12: Example of synchronization error in the Caremedia data set where two sets of cameras provide contradicting information on the location of a person.

have been manually synchronized, and the videos in Caremedia 8h were automatically synchronized with audio.

Camera Calibration: Camera calibration errors were also a source of error. As the nursing home scene no longer exists, the calibration of Caremedia cameras were performed directly on the video footage, and only rough estimates of the intrinsic and extrinsic camera parameters could be acquired. This caused issues in the localization of people. An example is shown in Figure 4.13, where the localization of the yellow bounding box in the lower right camera view was accurate, but the yellow bounding box in the upper right camera view was not accurate. However, as camera calibration errors were only severe at some locations, we believe it does not cause as much error as missed detections, face recognition errors and synchronization issues.
Figure 4.13: Example of calibration error in the Caremedia data set. The localization of the yellow bounding box in the upper right camera view was not accurate.

4.4 Summary

In this chapter, we demonstrated the effectiveness of utilizing an internal constraint: spatial-temporal smoothness, to automatically collect large amounts of person ReID training data for training deep ReID models. As the collected samples were accurate, diverse and in-domain, the features acquired from the deep models trained on the collected samples were able to further improve tracking performance. On the terrace1 sequence, our tracker was able to achieve near perfect performance thanks to the very effective training data collected and the ability of deep networks to learn powerful features to distinguish each person.

Combined with the advances from the current chapter and Chapter 3, our tracker was able to locate a person with 70% precision and 57% recall in 4,935 hours of surveillance video. These promising results motivate and enable us to perform surveillance video summarization, which is detailed in Chapter 6.
Chapter 5

Unsupervised Adaptation of Image-based Pose Detectors to Video

Pose detection focuses on finding the location of each joint of a person, which would be very useful information when trying to understand the action of a person. For example, in the nursing home environment, through pose estimation of elderly people sitting in a dining room, we can automatically compute the eating speed (frequency of food to mouth) of each elderly person, which would be very useful in assessing the person’s state of health.

Pose estimation algorithms have been developed for analyzing static images [8, 186–188] and video [6, 33, 189–191]. The standard procedure for evaluating these methods is to train and test a pose detector on different splits of the same data set, which implies that both sets are from the same domain. The pose detectors trained based on this standard procedure face one big problem if the detectors were to be applied to a new (surveillance) video data set. There is highly likely to be domain difference between the training set and the new video data set, thus leading to degradation in pose detection performance. An example is shown in Figure 5.1, which shows that the appearance, configuration and angle of poses are different for each data set.

A straightforward method to overcome this domain difference is to manually label training data for each new domain. However, manually labeling each new domain may be too tedious to perform. Therefore, we propose to utilize constrained self-training to automatically harvest in-domain training data directly from unlabeled surveillance videos. The collected training data is then combined with the existing out-of-domain training data to build a more effective pose estimator for the new domain.
Starting from a static-image based pose detector, we propose to perform constrained self-training to gradually adapt the pose detector to video data. Self-training [30, 31] is the iterative process of adding highly confident predicted testing instances into the training set to enhance the current model. However, as highly confident predicted instances may not always be correct, we constrain the self-training process by only selecting highly confident poses which follow the internal spatial-temporal smoothness constraints. The assumption is that pose estimation results in neighboring frames should not vary too much. If the estimated pose for two neighboring frames vary greatly, then it is highly likely that one of the detected poses are incorrect despite the fact that they both may have high confidence scores. In this case, we will ignore these two pairs of frames. The advantage of our method is that since it is fully unsupervised, it has the ability to harvest training examples from large amounts of unlabeled in-domain testing videos. Therefore, we can afford to be very conservative and only select instances in which we are very confident, i.e. it is acceptable if we throw away 99.99% of the data as long as if we still have a 0.01% harvest rate on a very large unlabeled set.

We emphasize that our main goal is not to utilize the spatial-temporal smoothness constraint to enhance pose estimation in video, but to utilize this constraint as a consistency check for better self-training of pose estimation models, which leads to better domain adaptation from images to video. This is a different motivation and method of utilizing the spatial-temporal smoothness constraints for video pose estimation compared to [6, 190], which directly learns a video-based pose detector. Video-based pose detectors utilize spatial-temporal smoothness constraints themselves to acquire a pose detection result, so utilizing the same constraint to verify whether a pose detection is correct or not is unreasonable.
The advantage of our algorithm is two folds. First, our method is not constrained to which pose estimation model is used. Any method which outputs a confidence score corresponding to the detected pose can be used. Second, our method is fully automatic, thus it has the potential to adapt to large varieties of video domains without any supervision. In sum, our contributions are as follows:

1. We propose to automatically adapt image pose detectors to video by utilizing the spatial-temporal smoothness constraint to perform robust self-training.
2. Experiments performed on the Caremedia video data set with two pose detectors: [8, 186] demonstrates the effectiveness of our method.

5.1 Related Work - Pose Estimation

Pose estimation in images [186–188] and video [6, 33, 189–191] have been intensively studied but still remains to be a challenging topic.

One main challenge of pose estimation is developing effective and efficient models to parse human joints. As humans limbs form a tree structure, tree structured models (also called pictorial structures) [6, 186, 192] have been widely used for pose estimation as they are computationally efficient and effective. However, as these models do not take into account the relation between the two hands and legs, double counting is a big problem, e.g. the pose estimation prediction associates both the left and right hand to the same physical hand. Self-occlusions is also a big problem, as different parts may articulate and occlude other parts. To model the interacting joints, many loopy graph-based models have been introduced [189, 193, 194], but these models are usually harder to optimize and significantly more time consuming to run. [188] replaced the loops in the graph by introducing multiple layers of simple classifiers, i.e. a sequential prediction model, which lead to the efficient and effective pose machine algorithm. Many video-based pose estimation algorithms have also been proposed [6, 189–191], but they also face the same trade-off of faster but coarser tree-models versus slower but more fine-grained graph models. Furthermore, video-based methods are more difficult to train because annotation of video training data is required, which is significantly more tedious than annotating a single image.

In the advent of deep learning, multiple deep convolutional pose estimators were proposed. In terms of how the output and loss of the deep network were formulated, there are roughly two families. One kind of network directly outputs the \((x, y)\) Cartesian coordinates of each body joint [178, 195], i.e. the network directly regresses over the coordinates and the network is optimized with the Euclidean loss between the predicted
and ground-truth locations. The second kind of network outputs a heat-map to the locations of each joint [8, 196], and the network is optimized by minimizing the Euclidean loss between the predicted and ground-truth heat-maps. The main advantage of utilizing the heat-map is that uncertainties in joint locations are preserved, whereas only outputting the \((x, y)\) locations provide no information in terms of confidence. Therefore, when the entire heat-map is fed into the next model in a sequential prediction framework, the next model can utilize the uncertainties to generate better pose estimation predictions [8].

On the other hand, regardless of how the loss of a deep pose estimator was formulated, both [195] and [8] were formulated under the sequential prediction model. [195] iteratively predicted the necessary offset required to move the current joint prediction to the correct pose location. [8] had multiple stages where each stage outputs a heat-map for each joint. The heat-map from the previous stage was taken as input and refined by the next stage. More details of [8] are in Section 5.3.

Another main challenge of pose estimation comes from the large configuration of possible human poses, which is difficult enumerate even with the large data sets currently available. Current popular pose estimation data sets include PARSE [197] with 305 annotated full-body poses, BUFFY [198] with 748 upper body poses, VideoPose2.0 [6] with 1286 upper body frames, FLIC [187] with 5K upper body frames, extended LEEDS Sports [199] with 12K poses, and MPII human pose dataset [200] with 40K poses. Even though coverage increases as the data sets become larger, but as the coverage of these data sets are not perfect, in-domain data will still be very useful in enhancing pose estimation. This motivates us to utilize constraint self-training to automatically adapt an image-based pose detector to any video data set.

Finally, our work can be viewed as an extension of a classic pose detector [192]. [192] proposes to track each limb of a person by first finding highly confident pose detections of the person using a pre-trained lateral person pose detector and then directly learning an appearance model for each limb of the person, thus involving two models. In our work, we merge these two models by updating the pose detector directly based on the highly confident pose detections. Also, our work utilizes spatial-temporal constraints to select highly confident poses, which was not used in [192].
5.2 Constrained Self-Training for Unsupervised Domain Adaptation

We propose to automatically adapt image-based detectors to video based on constrained self-training (CST). As shown in Figure 5.2, the input to our algorithm is an initial image-based pose detector and videos on which we would like to perform pose estimation. The output will be a pose estimator adapted to the video-domain at hand. The main steps of our method is as follows:

1. Perform pose estimation on input videos with current model.
2. Locate future pose estimation training data by selecting results that not only have high confidence, but also violate the continuity and tracking constraint the least.
3. Update/retrain pose estimation model based on newly selected data. Go to step 1 till maximum iterations reached.

The continuity and the tracking constraints are the key to prevent the self-training procedure from utilizing incorrect pose estimation examples for training. In the following sections, we will describe the two constraints in detail.

5.2.1 Continuity Constraint

The continuity constraint captures the fact that limbs should not move too far in a short period of time. We model this by finding a smoothed trajectory of body part locations over time which also fits the current observations. Let $P$ be the number of body parts, and $F$ be the total number of frames in a video. Let $p_i^f$, $1 \leq i \leq P$, $1 \leq f \leq F$ be the location of part $i$ detected by the pose detector in frame $f$. Let $\tilde{p}_i^f$ be the location of the body parts for the smoothed trajectories. We find the smoothed trajectories by minimizing the following equation.

\[
\min_{\tilde{p}_i^f} \sum_{f=1}^{F} \sum_{i=1}^{P} \left( \| \tilde{p}_i^f - p_i^f \|_2^2 + \alpha \| \tilde{p}_i^{f+1} - \tilde{p}_i^f \|_2^2 \right). \tag{5.1}
\]
The first term makes sure the smoothed trajectory is consistent with the detections. The second term minimizes the distance between detections in adjacent frames, which forces the trajectory to be smooth. $\alpha$ controls the relative weight between these two terms.

### 5.2.2 Tracking Constraint

The tracking constraint assumes that the body parts in different frames should be consistent with local optical flow based tracking results. The tracking procedure is performed as follows. For each pose estimation in each frame, we track each body part for 1 second. Each body part is tracked with dense trajectories [201] based on the KLT tracker [202]. Dense trajectory keypoints are extracted from the bounding box of the body part and tracked. As the body parts are tracked for 1 second, we can compute a location hypothesis for the tracked body part for each frame in the 1 second window. The body part location hypothesis for a frame is the average location of the tracked keypoints in that frame.

For a body part in a single frame, it will have multiple location hypotheses from neighboring frames. We merge these location hypotheses to compute the most likely location of the body part by performing a weighted sum over all the location hypotheses. The weight of each location hypothesis corresponds to the detection confidence of the detected pose which provided the specific hypothesis. Let the results of the weighted sum be $T_f$. We update Equation 5.1 such that the smoothed trajectories also need to be close to the tracking hypotheses.

$$
\min_{\tilde{p}_f} \sum_{f=1}^{F} \sum_{i=1}^{P} \left( \left\| \tilde{p}_f - p_f \right\|_2^2 + \alpha \left\| \tilde{p}_{f+1} - \tilde{p}_f \right\|_2^2 + \beta \left\| \tilde{p}_f - T_f \right\|_2^2 \right).
$$

(5.2)

$\beta$ controls the relative weighting of the tracking constraint.

### 5.2.3 Selecting Positive Examples

We now utilize the continuity and tracking constraint to add pose detections which are highly likely to be correct into the self-training process. Once we have computed $\tilde{p}_f$ by solving Equation 5.2, we can compute the pose detections which are not only confidently detected by the pose detector, but also violates the constraints the least. Let $c_f$ be the confidence of the pose detection in frame $f$. Let $s_f$ denote the negated error in the two
Unsupervised Adaptation of Image-based Pose Detectors to Video

5.3 Experiments

We evaluated our constrained self-training pose estimation approach with the following setup.

**Pose Estimators**: To demonstrate that our method is agnostic to the pose estimator used, we performed pose estimation experiments with the Flexible Mixture-of-Parts (FMP, [186]) and Convolutional Pose Machines (CPM, [8]).

FMP utilizes a tree structure to model a person, and each body part is represented by a mixture of configurations. HOG [128] was used as the low-level feature. The code was acquired directly from the authors [186], and the optimal parameters: 26 joints and 6 clusters were used to train FMP. There is no clear way to fine-tune FMP with in-domain samples, so a new model was retrained whenever a new set of training instances were provided.

CPM is a deep pose estimator with multiple stages as shown in Figure 5.3. There are two main parts: the low-level network and the high-level network. The low-level network consists of 4 convolutional layers with filter size $5 \times 5 \times 128$ for all layers and max-pooling in the first three layers which reduces the size of the original image by a factor of 8. This captures the lower level details of the image. Then the high-level network will take as input the output of the low-level network and the output of the previous high-level network. For the first high-level network, there are 3 convolutional layers with filter size $9 \times 9 \times 512$, $1 \times 1 \times 512$ and $1 \times 1 \times 15$. For the other high-level networks, there are 5 convolutional layers with filter size $9 \times 9 \times 64$, $9 \times 9 \times 64$, $9 \times 9 \times 128$, $1 \times 1 \times 128$ and $1 \times 1 \times 15$. All outputs of the high-level network are connected to a Euclidean loss of the
Figure 5.3: Illustration of the CPM pose estimator with 3 high-level network stages.

The advantage of the multi-layer design is that double counting can be alleviated, i.e. if the left and right wrists were all predicted on the same physical wrist, the next stage can change the location of one of the wrists to the other physical wrist. In our experiments there were 3 high-level networks. The CPM implementation\(^1\) was based on Caffe [183]. If a CPM was trained from scratch, the base learning rate was \(2 \times 10^{-5}\), and the learning rate was divided by 3 every 66,000 iterations. Batch size was set to 12. For fine-tuning, the base learning rate was \(5 \times 10^{-6}\), and the number of iterations utilized was \(2.5 \times\) number of fine-tuning instances. In the prediction phase, the locations with the highest scores were selected as the locations of each body part, and the confidence of the pose was computed by averaging the highest scores of each body part.

Data Sets: The FMP pose estimators were trained on PARSE [197], which had 305 full-body poses, and the negative set was from the INRIA Person database. Data augmentation by mirroring and rotating the images by -15, -7.5, 7, 7.5 degrees were performed. One limitation of FMP is that the method assumes all joints are visible, thus when training over the Caremedia pose data set described below, only the poses where all joints were visible were used.

The CPM pose estimators were trained on LEEDS Sports [199, 203]. There were 11K training images, and each image was resized to 304-by-304. Data augmentation by mirroring and rotating the images from -150 to 180 degrees in 30 degree intervals were performed. 213,000 iterations were run to train the CPM network on LEEDS Sports.

The target domain was the Caremedia data set, where 3,193 poses were annotated. 14 joints were annotated if the joints were visible. Some examples are shown in Figure 5.4. To compare the effectiveness of our self-training method, we also evaluated the performance of models which used in-domain ground-truth instances for fine-tuning. To perform fine-tuning experiments, the Caremedia data set was split into two folds, where one fold was used for fine-tuning and the other fold was used for testing. The prediction results from the two folds were combined and an overall score was computed. The

\(^1\)Code modified from Shih-En Wei’s implementation of CPM.
folds were split so that ground-truth instances of a single individual would all be in the same fold. This was to prevent the pose estimator from “memorizing” a single individual. Data augmentation by mirroring and rotating the images by -15, -7.5, 7.5 and 15 degrees were performed. Also, to understand the effect of training data size versus pose estimation performance, we randomly sampled from the training fold 50, 200, 584 and 800 instances 5 times each and utilized each sampled training set to fine-tuning the CPM. For FMP, we sampled 50 and 227 instances per fold.

**Parameters:** The self-training methods were presented 10 second video clips which encompassed each of the 3,193 annotated poses. For each video clip, at most one pose estimation which had a score larger than a threshold was selected. For FMP, the minimum confidence thresholds used for self-training was $c^f \in \{0.9, 1.0\}$ and CST was $c^f \in \{0.4, 0.5\}$, $s^f \in \{-600, -1000\}$. For CPM, the minimum confidence thresholds used for self-training was $c^f \in \{0.5, 0.6, 0.7\}$ and CST was $c^f \in \{0.5, 0.6, 0.7\}$, $s^f = -350$. We ran the self-training for one iteration. Internal tests have shown that running on one iteration already saturates performance, which we believe was due to our unlabeled set not being big enough. We set $\alpha = 5$ and $\beta = 0$ in our experiments. $\beta = 0$ because we found in our preliminary experiments that KLT keypoint tracking cannot find many keypoints to track in low resolution Caremedia video, which leads to severe body-part tracking error. Therefore the tracking continuity term was not utilized in our experiments. This issue could be alleviated by utilizing a patch-based tracker instead of a keypoint-based tracker.
Evaluation Metrics: We utilized the widely used Percent of Correct Keypoints (PCK, specifically PCK@0.2) [187] as our evaluation metric. For PCK@0.2, a keypoint is counted as correct if the distance between the keypoint and the ground-truth is less than $0.2 \times d$, where $d$ is the diameter of the torso, i.e. distance between the left hip and the right shoulder. In this way, the accuracy for each of the 14 keypoints can be computed.

Results and Discussion

The performance of constrained self-training to adapt an image-based detector trained on PARSE or LEEDS Sports to Caremedia is summarized in Table 5.1 and Table 5.2 respectively. As the CPM is the state-of-the-art pose detector, a more detailed analysis of the CPM trained with different numbers of LEEDS Sports training examples are shown in Figure 5.5a. We can see that both self-training and CST were effective in enhancing both FMP and CPM pose estimators. For FMP, CST performed slightly better than self-training. However, there was no clear difference between the performance of self-training and CST for CPMs. This may be because a very conservative self-training $c_f$ threshold collected around the same number of training examples as CST, which could afford a lower $c_f$ threshold but would end up only harvesting the same number of training examples because many examples were filtered out by the smoothness threshold $s_f$. Nevertheless, CST still enabled us to automatically collect in-domain data, and from Figure 5.5a we can see that a LEEDS Sports model with 2,000 training instances and enhanced with CST performed as well as a LEEDS Sports model with 11,000 instances. This shows that CST was equivalent to 9,000 out-of-domain instances, thus demonstrating the importance of acquiring in-domain data. Compared with the fine-tuning scenario, CST was equivalent to around 50 manually collected in-domain training samples, which takes around 1 hour of manual human effort to acquire. Also, in general the margin of improvement of both CST and fine-tuning decreased as more out-of-domain training samples were used for training.

Digging deeper into the figures, we can see clearly from Figure 5.5b that the quality of training instances collected manually versus through CST was different. For example, with 584 manually collected training instances, the PCK was 88.5, whereas for CST it was only 84.2. There are 3 possible reasons of this gap: 1) the automatically collected training instances were inaccurate, 2) the collected instances were biased to a specific set and lacked variety, and 3) the collected instances were too easy thus not informative to the classifier. In order to understand the reason of this gap in performance, we manually corrected all the 584 pose estimation results collected by CST when $c_f = 0.6$ and $s_f = -350$. Then we utilized these manually corrected instances to fine-tune the
Unsupervised Adaptation of Image-based Pose Detectors to Video

<table>
<thead>
<tr>
<th>Method</th>
<th>PCK</th>
<th>In-domain (pseudo-)training instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARSE</td>
<td>64.6</td>
<td>0</td>
</tr>
<tr>
<td>CST, $c^f = 0.5, s^f = -600$</td>
<td>67.6</td>
<td>196</td>
</tr>
<tr>
<td>CST, $c^f = 0.4, s^f = -1000$</td>
<td>69.1</td>
<td>347</td>
</tr>
<tr>
<td>Self-training, $c^f = 1.2$</td>
<td>66.6</td>
<td>208</td>
</tr>
<tr>
<td>Self-training, $c^f = 1.0$</td>
<td>67.5</td>
<td>340</td>
</tr>
<tr>
<td>Self-training, $c^f = 0.9$</td>
<td>67.5</td>
<td>435</td>
</tr>
<tr>
<td>50 fine-tuning instances</td>
<td>68.5</td>
<td>50</td>
</tr>
<tr>
<td>227 fine-tuning instances</td>
<td>70.6</td>
<td>227</td>
</tr>
</tbody>
</table>

Table 5.1: PCK performance for self-training, CST and fine-tuning based on PARSE data set with the FMP model.

<table>
<thead>
<tr>
<th>Method</th>
<th>PCK</th>
<th>In-domain (pseudo-)training instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEEDS Sports 11K</td>
<td>79.5</td>
<td>0</td>
</tr>
<tr>
<td>CST, $c^f = 0.7, s^f = -350$</td>
<td>82.8</td>
<td>189</td>
</tr>
<tr>
<td>CST, $c^f = 0.6, s^f = -350$</td>
<td>84.2</td>
<td>584</td>
</tr>
<tr>
<td>CST, $c^f = 0.5, s^f = -350$</td>
<td>84.2</td>
<td>1220</td>
</tr>
<tr>
<td>CST, $c^f = 0.4, s^f = -350$</td>
<td>81.7</td>
<td>1708</td>
</tr>
<tr>
<td>Self-training, $c^f = 0.7$</td>
<td>84.5</td>
<td>675</td>
</tr>
<tr>
<td>Self-training, $c^f = 0.6$</td>
<td>83.9</td>
<td>1695</td>
</tr>
<tr>
<td>Self-training, $c^f = 0.5$</td>
<td>84.5</td>
<td>2592</td>
</tr>
<tr>
<td>50 fine-tuning instances</td>
<td>85.8</td>
<td>50</td>
</tr>
<tr>
<td>200 fine-tuning instances</td>
<td>87.4</td>
<td>200</td>
</tr>
<tr>
<td>584 fine-tuning instances</td>
<td>88.5</td>
<td>584</td>
</tr>
<tr>
<td>800 fine-tuning instances</td>
<td>89.8</td>
<td>800</td>
</tr>
<tr>
<td>1600 fine-tuning instances</td>
<td>90.5</td>
<td>1600</td>
</tr>
</tbody>
</table>

Table 5.2: PCK performance for self-training, CST and fine-tuning based on the LEEDS Sports 11K CPM model.

CPM, and results are shown in Figure 5.6a. Results showed that our CST actually achieved the same level of performance as “manually corrected CST” and “584 randomly sampled fine-tuning instances” for the upper body, which includes the head, shoulder, elbow and wrist. However, CST performed significantly worse on the hip, knee and ankles. Digging deeper into why CST was not as helpful on the hip, knee and ankle, we found that the labeling accuracy of these joints by CST were significantly lower than the other joints as shown in Figure 5.6b. There is a very clear positive correlation between the body joint labeling accuracy of CST versus the relative PCK performance of CST when compared to the “manually corrected CST” run. We believe this can largely explain the performance gap between CST and fine-tuning with manually labeled samples. This leads us to conclude that: 1) the collected instance were not too easy for the classifier because we see as much gain in performance on the upper body joints, 2) the collected instances have just as much variety as the randomly sampled manual
Unsupervised Adaptation of Image-based Pose Detectors to Video

Figure 5.5: Results of self-training compared to “fine-tuning on manually labeled instances” with the CPM model.

(a) PCK performance of self-training and “fine-tuning with manually labeled instances” when utilizing different numbers of out-of-domain LEEDS Sports training examples.

(b) Results of self-training and fine-tuning with 11,000 out-of-domain training examples. The 95% confidence interval is shown for the “fine-tuning with manually labeled instances”.

instances, because the average PCK of the “manually corrected CST” and “584 randomly sampled fine-tuning instances” runs have similar performance, and 3) the accuracy of the automatically labeled instances played a key role in affecting CST performance.

Qualitative analysis of CST pose estimation was also performed. As CST directly harvests training examples from the testing data, CST can potentially “memorize” a person, thus enhancing the pose estimation performance for that specific person. Figure 5.7 shows the poses which were incorrectly predicted by the out-of-domain model but correctly predicted by the CST model. A CST collected instance which could have likely helped in correcting this pose is also shown. Examples of pose estimation failure cases are shown in Figure 5.8. Failure cases includes double counting for the first three images, confusing hairstyles for the second image, and challenging upper-body poses with occlusions and another person.

5.4 Summary

We present constrained self-training, which utilizes the spatial-temporal smoothness constraint to adapt static-image trained pose detectors to video data. The key advantage of our method is that the whole process is unsupervised, thus enabling us to potentially generalize to a large variety of video. Experiments showed that CST was effective in adapting two image pose detectors: FMP and CPM to the Caremedia video domain in an unsupervised fashion, thus supporting that CST is agnostic to the pose detector used. For CPM, a pose estimator trained on 2,000 out-of-domain training examples combined with CST achieved equivalent performance to a model trained with 9,000
(A) PCK performance for different body parts under different conditions.

(B) Relation between CST labeling accuracy and PCK performance relative to the manually corrected CST run with the CPM model.

Figure 5.6: Analysis of CST performance drop.

Figure 5.7: Examples of pose estimation errors which were fixed through CST. Green and red lines correspond to the correct and incorrect limbs respectively. The image connected by a dotted line are training examples harvested from CST.
more (total of 11,000) out-of-domain training examples. Also, the effect of CST was equivalent to around 50 manually labeled in-domain instances, thus saving around 1 hour of manual effort. Detailed analysis of the training samples collected by CST showed that the samples collected were actually diverse and not simply focused on the easy instances. Also, for the upper body joints, the performance of models fine-tuned with CST collected instances achieved as good a performance compared to models fine-tuned on manually collected instances. However, CST labeling errors on the lower body heavily affected the pose estimation performance on the corresponding joints, which was the main cause of CST not performing as well as fine-tuning on manually collected instances. Another negative outcome is that self-training and CST performed very similarly, thus showing that the constraint may not be very cost-effective in collecting accurate training examples.

A future work is to apply CST to very large amounts and large varieties of video to automatically collect training data to improve a generic pose estimator. This process could be run iteratively to potentially perform never-ending pose estimation learning. However, the biggest bottleneck in our experiments were the speed of pose estimation on videos, which was the limiting factor for the number of videos used in our CST experiments. Therefore, developing efficient pose estimators for video will potentially enable us to unleash the full power of CST.
Chapter 6

Long-Term Surveillance Video Analysis

According to reports\textsuperscript{1}, there were 245 million surveillance cameras operating in 2014. This means that every hour, 245 million hours of surveillance video are generated. Though many surveillance cameras have lower frame-rates which will slightly alleviate the amount of data to be analyzed, the amount of data generated is still too much to analyze manually. Therefore, we demonstrate in this chapter two applications of automated long-term surveillance video analysis based on the identity-aware multi-object tracker and pose detector we developed. The first use case is video summarization and long-term statistics computation based on tracking output. The second use case is eating detection based on pose estimation.

6.1 Visual Diary Generation with Multi-Object Tracking

To demonstrate the usefulness of our tracking output, video summarization experiments were performed. We propose to summarize surveillance video using visual diaries, specifically in the context of monitoring elderly residents in a nursing home. Automatic visual diary generation for elderly nursing home residents enables doctors and staff to quickly understand the activities of a senior person throughout a day to facilitate the diagnosis of the elderly person’s state of health. The visual diary for a specific person consists of two parts: 1) snippets which contain snapshots and textual descriptions of activities-of-interest performed by the person, and 2) activity-related statistics accumulated over time. The textual descriptions of the detected events enables efficient indexing of what

\textsuperscript{1}https://technology.ihs.com/532501/245-million-video-surveillance-cameras-installed-globally-in-2014
a person did at different times. The statistics for the activities detected can be accumulated over many days to discover long-term patterns. An example is shown in Figure 6.1, where the visual diary of a nursing home resident is shown.

We propose to generate visual diaries with a summarization-by-tracking framework. Using the trajectories acquired from our tracking algorithm, we extract motion patterns from the trajectories to detect certain activities performed by each person in the scene. The motion patterns are defined in a simple rule-based manner. Even though more complex methods such as variants of Hidden Markov Models [204] to detect interactions could also be used, this is not the main focus of our work. Also, our experiments showed that with the rules we defined, reasonable interaction detection performance was achieved. The activities we detect are as follows:

- **Room change:** Given the tracking output, we can detect when someone enters or leaves a room.
- **Sit down / stand up:** We trained a sitting detector [167] which detects whether someone is sitting. Our algorithm looks for tracks which end/begin near a seat and check if someone sat down/stand up at around the same time.
- **Static interaction:** if two people stand closer than distance $D'$ for duration $T'$, then it is likely that they are interacting.
- **Dynamic interaction:** if two people are moving with distance less than $D'$ apart for a duration longer than $T'$, and if they are moving faster than 20 cm/s, then it is highly likely that they are walking together.
According to [205], if people are travelling in a group, then they should be at most 7 feet apart. Therefore, we set the maximum distance $D'$ between two people for there to be interaction to 7 feet. The minimum duration of interaction $T'$ was set to 8 seconds in our experiments.

Given the time and location of all the detected activities, we can sort the activities according to time and generate the visual diary. The visual diary for a given individual consists of the following:

- **Snippets**: snapshots and textual descriptions of the activity. Snapshots are extracted from video frames during the interaction and textual descriptions are generated using natural language templates.
- **Room/state timing estimates**: time spent sitting or standing/walking in each room.
- **Total interaction time**: time spent in social interactions.
- **Interacted targets**: people with whom the person interacted.

Our proposed method of using tracking output for activity detection can be easily combined with traditional activity recognition techniques using low-level features such as Improved Dense Trajectories [206] with Fisher Vectors [207] to achieve better activity detection performance and detect more complex actions, but extending activity recognition to activity detection is out of the scope of this work.

### 6.1.1 Visual Diary Generation Results

We performed long-term surveillance video summarization experiments by generating visual diaries for the Caremedia 8h sequence based on trajectories acquired with our Solution Path Algorithm tracker and deep appearance features. To acquire ground truth for activity detection experiments, we manually labeled the activities of 3 residents throughout the sequence. The 3 nursing home residents were selected because they are the people who we would like to focus on for the automatic analysis of health status.

We evaluated the different aspects of the visual diary, including “room/state timing estimates”, “interaction timing estimates”, “interacted target prediction” and “snippet generation”.

The evaluation of “room/state timing estimates”, i.e. predicted room location and state (sitting or upright), of a person was done on the video frame level. A frame was counted as true positive if the predicted state for a given video frame agreed with the ground truth. False positives and false negatives were computed similarly.

To evaluate “interaction timing estimates”, i.e. how much time a person spent in interactions, a frame was only counted as true positive if both the prediction result and
Long-Term Surveillance Video Analysis

<table>
<thead>
<tr>
<th>Visual diary components</th>
<th>Micro-Precision</th>
<th>Micro-Recall</th>
<th>Micro-F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snippet generation</td>
<td>0.382</td>
<td>0.522</td>
<td>0.441</td>
</tr>
<tr>
<td>Room/state timing estimates</td>
<td>0.809</td>
<td>0.511</td>
<td>0.626</td>
</tr>
<tr>
<td>Interaction timing estimates</td>
<td>0.285</td>
<td>0.341</td>
<td>0.311</td>
</tr>
<tr>
<td>Interacting target prediction</td>
<td>0.533</td>
<td>0.762</td>
<td>0.627</td>
</tr>
</tbody>
</table>

Table 6.1: Evaluation of the generated visual diary.

<table>
<thead>
<tr>
<th>Patient</th>
<th>Time spent in ...</th>
<th># interacted people</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sit in living room</td>
<td>stand/walk in living room</td>
</tr>
<tr>
<td>Patient 3</td>
<td>00:26:28</td>
<td>00:19:05</td>
</tr>
<tr>
<td>Patient 6</td>
<td>01:18:14</td>
<td>00:09:02</td>
</tr>
<tr>
<td>Patient 11</td>
<td>00:20:10</td>
<td>00:01:25</td>
</tr>
</tbody>
</table>

Table 6.2: Summary of important statistics for 3 nursing home residents in video with 7 hours 45 minutes wall time. Timing is formatted as hh:mm:ss. For example, patient 6 spent 1 hours 18 minutes and 14 seconds sitting in the living room.

ground truth result agreed that there was interaction and also the ID of the interacted targets matched. False positives and false negatives were computed similarly. For “interacted target prediction”, i.e. who interacted with whom, a true positive was counted when the predicted and ground truth output both agreed that the resident interacted with a given person. False positives and false negatives were computed similarly.

The evaluation of “snippet generation” accuracy was done as follows. For snippet related to sit down, stand up and room change activities, a snippet was correct if the predicted result and ground truth result had less than 5 second time difference. For social interaction related snippets, a snippet was correct if 50% of the predicted snippet contained a matching ground truth interaction. Also, if a ground truth interaction was predicted as three separate interactions, then only one interaction would be counted as true positive while the other two would be counted as false positives. This prevented double counting of a single ground-truth interaction.

Based on the tracking output, we performed activity detection and visual diary generation on the three residents. 184 ground-truth snippets were annotated. The performance of visual diary generation is summarized in Table 6.1. From the table, 38% of the generated snippets were correct, and we have successfully retrieved 52% of the activities-of-interest. For “room/state timing estimates”, a 51.1% recall shows that we know the state and room location of a person more than 50% of the time. The lower performance for “interaction timing estimates” is mainly caused by tracking failures, as both persons need to be tracked correctly for interactions to be correctly detected and timings to be accurate. However, if we only want to know the interaction targets, we still can achieve 63% F1-score. These numbers are not high, but given that our method
is fully automatic other than the collection of the face gallery, this is already a good first cut at generating visual diaries for the elderly by summarizing hundreds or even thousands of hours of surveillance video.

As our visual diary generation is heavily based on tracking, we analyzed the effect of tracking performance on visual diary generation accuracy. We computed the snippet generation F1-score for multiple tracking runs with varying tracking performance. These runs include our baseline runs and also runs where we randomly corrupted face recognition labels to decrease tracking performance. Results are shown in Figure 6.3, which shows that as tracking performance increases, snippet generation F1 also increases with a trend which could be fitted by a second-order polynomial.

Figure 6.1 and Figure 6.2 show example visual diary snippets for the three residents. From the generated snippets, we can clearly see what each resident was doing at each time of the day. Long term statistics were also compiled as shown in Table 6.2, which clearly shows the amount of time spent by each person in each room and in social interactions.

6.1.2 Long-term (23 Day) Statistics

We also showed how the visual diary could be extended to utilize statistics over 23 days. The amount of time interacting with each person, the amount of time spent in interactions each day and the distance walked everyday were automatically computed and aggregated into a single report as shown in Figure 6.4. Based on our manual observation, ID 34 is resident 11’s partner, which is reflected in our pie chart as ID 34 is ranked second in all the people with whom resident 11 interacts. These long-term statistics, which are too tedious to collect manually, have the potential aid doctors and staff in assessing the health status of nursing home residents. Though there is not a very clear trend in Figure 6.4 for interaction timing and walking distance, we believe this is mainly due to our observation being only limited to 23 days. If our analysis was performed over a period of half an year or a whole year, then it is likely some hidden trends will be visible from the automatically computed statistics.

6.2 Eating Detection with Pose Estimation

We explore the usefulness of our pose estimator for eating detection. As a proof-of-concept, our main goal is to automatically detect when a person “takes a bite”, which could be useful in the analysis of eating speed and eating behavior. We propose to utilize
Figure 6.2: Selected visual diary snippets for each resident.
the distance between the resident’s wrist and his/her head to detect whether the person took a bite.

3 residents were selected and for each resident, we annotated 15 minutes of eating footage for 7 out of the 23 days, thus leading to 105 minutes of footage per person. We manually selected the 15 minutes of video which contains footage of the resident of interest eating. The annotation protocol was that whenever a person touches his/her head with either a spoon, fork, cup, hand or napkin, then it is a true positive.

The pose estimator was applied on all frames of the video. To make our computation faster, we also manually cropped the bounding box to indicate the region of interest. This step could still be automatically performed by utilizing face recognition to locate the resident of interest. As wrist joints were often occluded or hard to distinguish, we
only retained wrist joint detections which had a prediction score larger than 0.5. Then we smoothed the wrist joint detections over time to create a trajectory of the wrist. Finally, the distance between the wrist and the head was computed, and if the distance was less than a certain threshold, a “taking a bite” action had taken place.

We evaluated our performance with frame-based precision, recall and F1. A video frame was true positive if both the prediction and ground-truth indicate that there was an eating action in that frame. False positives and false negatives were computed accordingly. The statistics from different people were aggregated to compute the final micro-F1 score, which takes into account the number of positive instances per person. Computing the distance threshold for eating was difficult because the threshold was greatly affected by the size of the person in the video and the angle of how the camera viewed the person. Therefore, to circumvent this issue, we utilized the ground-truth data to find the break-even point (the threshold where precision equals recall) and report the F1-score there. In this way, our performance was not affected by the quality of the selected threshold.

Pose detectors which used different training data were tested. 5 pose detectors used 50, 200, 800, 1600, and 3193 manually labeled in-domain instances respectively for fine-tuning. These in-domain instances included samples of the same individual eating, so it is expected that the performance will be higher. Our constrained self-training (CST) based pose estimator was also used. Results are shown in Table 6.3. We can see an improvement of CST and other fine-tuned pose detectors over the detector trained only on LEEDS Sports 11K, thus showing that there was significant domain difference. However, our CST-based pose detector was significantly lower than the detector fine-tuned over 50 manually labeled instances, which was different from our observation in Chapter 5. One main reason was that CST collected more full-body poses and had less eating related poses, thus leading to poor performance on eating poses.

On the other hand, the performance on resident 3 was significantly better for all detectors. Our observation was that performing pose estimation for eating activities from a slightly profile view was easier than the frontal view. For the frontal case, the elbow was often occluded by the table and not visible, and this may make the pose estimation for the wrist joint less accurate. However, for the side view case, all joints were clearly visible, thus leading to significantly better pose estimation. Figure 6.5 shows snapshots of pose estimation for the 3 eating residents.

In sum, we show that we were able to detect a “taking a bite” activity with 26% F1 if we used our CST pose estimator. This shows that there is still a big domain difference between LEEDS Sports and Caremedia, and the pose estimator enhanced with CST could not fully overcome this difference. However, a detector fine-tuned with manually labeled samples still achieved around 50% F1, which is already useful in automatically
aggregating the eating footage of a resident of interest over long periods of time. The footage can potentially aid doctors and staff in discovering subtle changes of the person’s eating behavior, such as slower movement or shakier hands, thus providing more information in better diagnosing the nursing home resident.

### 6.3 Summary

In this chapter, we have demonstrated how our developed multi-object tracker and pose estimator could be utilized to analyze very large amounts of surveillance video and detect certain events of interest with reasonable accuracy. Visual diary summarization achieved a precision of 38% and recall of 52%. Individual-specific interaction timing analysis also showed trends which were in line with manual observations. Also, our “taking a bite” activity detector based on CST achieves 26% F1. Though our numbers are not high, compared to tedious manual analysis of thousands of hours of surveillance video, our
methods are a strong alternative, and it potentially opens the door to automatic analysis of the ocean of surveillance video recorded everyday.
Chapter 7

Conclusions and Future Work

In this thesis, we proposed to utilize external knowledge and internal constraints in video to perform large-scale surveillance video analysis in an unsupervised fashion. We have presented improvements in identity-aware multi-object tracking and pose estimation. For multi-object tracking, we first demonstrated how to utilize external knowledge: face recognition combined with an internal constraint: spatial-temporal smoothness to localize and identify each person at each time instant. Then the same spatial-temporal smoothness constraint was further utilized to automatically collect large amounts of person re-identification training data, which was used to learn deep appearance features to further enhance tracking. For pose estimation, spatial-temporal smoothness was utilized to perform constrained self-training for unsupervised domain adaptation. Finally, our methods were applied to analyze 23 days of multi-camera surveillance footage. Tracking results showed that our system was able to localize a person 57% of the time with 70% precision. Summarization results showed that we were not only able to localize events of interest with 38% precision and 52% recall, but also accumulate useful long-term statistics for each individual. Overall, these results demonstrate the effectiveness of our methods, which potentially opens the door to the automatic analysis of thousands or tens-of-thousands of hours of surveillance video.

Future work in terms of long-term nursing home analysis and also surveillance video analysis in general would be as follows:

1. Automatically training a robust detector for a specific object or person pose: In specific surveillance scenes there will be different objects that one would like to detect and track. However, it is often the case that a robust detector is not available for those objects. Therefore, in order to be able to utilize tracking-by-detection, being able to automatically train an in-domain object detector would be crucial. In the nursing home case, it would be very useful if a general purpose
sitting person detector and wheelchair detector could be trained, so that we are not only limited to tracking pedestrian-like people. One direction could be to utilize unsupervised co-localization [59] or object discovery [60] mentioned in Chapter 2. The high-level idea is that given the large amount of unlabeled videos, objects that are of interest for tracking should form a large enough cluster which could be automatically discovered. Another direction for automatically collecting training data of sitting people could be to jointly utilize face recognition, pose detection and a chair detector which are all external resources we already have. With face recognition, we would know there is a person there. With pose estimation and a chair detector, we can predict whether the person is sitting. If we have multi-camera information, we could take this one step further and collect images/video of the same sitting person from multiple views.

2. Adapting pose detectors to deal with heavy occlusions: In indoor surveillance scenes, joints are easily occluded by static objects such as tables and chairs. Therefore, to perform robust pose estimation it is crucial that the pose estimator has the ability to deal with occlusions. Automatically collecting such training data is hard, and clever assumption/constraints are required. One potential assumption that could be utilized is: a person’s face and their hands have similar color. Based on this assumption, one would have a very strong prior for the location of a wrist, which could potentially be useful for collecting pose estimation training data.

3. Activity detection in surveillance videos: Being able to detect arbitrary activities of interest is also a crucial step for analyzing surveillance videos. The classic way of training activity detectors is also based on supervised learning, which requires manual annotation of training data. Motivated by our work on automatically collecting action examples from instructional videos [101], one could potentially collect activity training data from a surveillance scene based on cues from the audio channel and automatic speech recognition. These cues provide us indirect information on the activities currently occurring in the scene.

The above enhancements, if successful, will enable us to acquire more information for each surveillance scene with less manual effort. In the nursing home case, we will be able to track significantly more elderly residents and harvest even more information from nursing home surveillance videos.

Taking a step back, the two information sources we used: external knowledge and internal constraints, are not just useful in surveillance video analysis. In general, high-precision external knowledge sources can provide the initial labels for an otherwise completely unlabeled testing set, thus giving us a starting point to tackle the task. For example, in order to localize and identify each person in the scene, we leveraged face
recognition to give us an initial set of labels, on which we performed constrained optimization. On the other hand, internal constraints have the ability to constrain an unlabeled data set such that useful information can still be found. The internal constraints can be very strict, but since the constraint can be operated on the near infinite amount of unlabeled video data available, we can still collect a significant amount of useful information. For example, the spatial-temporal smoothness constraint enabled us to collect person re-identification data for any calibrated multi-camera network scenario. The exploitation of the massive amount and variety of unlabeled videos available potentially enables a machine learning algorithm to learn more generalizable models than models trained on manually labeled data sets. Therefore, when facing a new task on a new (unlabeled) data set, one could first try to look for suitable external knowledge or available internal constraints to acquire useful information and alleviate manual annotation.

Taking a step further back, we believe an even higher-level principle which encompasses the formulation of the two information sources we have proposed is: how can one be smart with the labeled and unlabeled data we have? In the era of big data, data is our best friend, and the people who have the creativity and ability to harness the infinite power locked in big data will be the ones who have the potential to make a large impact on society.
Appendix A

Details of Iterative Projection Algorithm for Tracking with Solution Path Algorithm

In Section 3.7.2, we wanted to solve the following optimization problem:

\[
\min_{\mathbf{G}} f(\mathbf{a}') = \left\| \mathbf{G} - \mathbf{a}' \right\|^2_2 \quad \text{s.t.} \quad \left\| \mathbf{G} \right\|_p \leq 1, \tag{A.1}
\]

where \( \mathbf{a}', \mathbf{G} \in \mathbb{R}^c \), \( \mathbf{a}' = (a_1, a_2, \cdots, a_c)^T \) is the known reference point and \( \mathbf{G} \) is the queried projection point on \( \ell_p \) ball \( \left\| \mathbf{G} \right\|_p \leq 1 \).

The details of our iterative projection algorithm for solving (A.1) are as follows:

1: If \( \left\| \mathbf{a}' \right\|_p \leq 1 \), then output \( \mathbf{G} = \mathbf{a}' \), and terminate the entire procedure. Otherwise, record \( \mathbf{s} = \text{sign}(\mathbf{a}') \) and reformulate \( \mathbf{a}' = \mathbf{s} \odot \mathbf{a}' \) to make all its elements non-negative (i.e. to let \( \mathbf{a}' \) located at the first quadrant). Initialize \( \mathbf{G}^{(1)} = (g_1^{(1)}, g_2^{(1)}, \cdots, g_c^{(1)})^T \) as the intersection of line segment connecting \( \mathbf{a}' \) and the origin and the boundary of the \( \ell_p \) ball \( \left\| \mathbf{G} \right\|_p = 1 \). This intersection can be found efficiently by the binary search strategy. Set \( l = 1 \) and \( \varepsilon \) as a small threshold value.

2: Repeat:

3: Compute the tangent plane of the \( \ell_p \) ball boundary curve \( \left\| \mathbf{G} \right\|_p = 1 \) at \( \mathbf{G}^{(l)} \) as:

\[
\pi^{(l)} = \{v | \mathbf{w}^{(l)}(\mathbf{G} - \mathbf{G}^{(l)}) = 0\},
\]
Details of Iterative Projection Algorithm for Tracking with Solution Path Algorithm

where

\[ w^{(l)} = \left( \nabla \| G \|_p \right) G^{(l)} \]
\[ = \left( p(g_1^{(l)})^{p-1}, p(g_2^{(l)})^{p-1}, \ldots, p(g_c^{(l)})^{p-1} \right)^T, \]

where \( g_i^{(l)} \) is the \( i \)th element of \( G^{(l)} \). Calculate the projection point of \( a' \) to \( \pi^{(l)} \) as

\[ x^{(l)} = a' - \frac{w^{(l)T} a' - w^{(l)T} G^{(l)}}{\|w^{(l)}\|_2^2} w^{(l)}. \]

4. If \( x^{(l)} \) is located in the first quadrant (i.e., \( x^{(l)} \geq 0 \)), then draw a line segment \( z(t) \) between \( a' \) and \( x^{(l)} \) as

\[ z(t) = (x^{(l)} - a')t + a', 0 \leq t \leq 1, \]

and compute its intersection point \( G^{(l+1)} \) with the \( \ell_p \) ball boundary curve \( \|G\|_p = 1 \) using binary search. Then let \( l = l + 1 \), and go to the next iteration.

5. If \( x^{(l)} \) is located outside the first quadrant, then calculate

\[ t^* = \min_i (t_i^*), t_i^* = \frac{a_i}{a_i - x_i}, \]

where \( a_i \) and \( x_i \) are the \( i \)th elements of \( a' \) and \( x^{(l)} \), respectively. If \( z(t^*) = (x^{(l)} - a')t^* + a' \) satisfies that \( \|z(t^*)\|_p \leq 1 \), then use the similar binary search strategy as step 4 to calculate the projection point \( G^{(l+1)} \). Then let \( l = l + 1 \), and go to the next iteration.

6. If \( \|z(t^*)\|_p > 1 \), this means that there is no intersection of the line segment \( z(t) \) (0 \( \leq t \leq 1 \)) and the \( \ell_p \) ball boundary curve \( \|G\|_p = 1 \). Calculate the critical point \( y(s^*) \) where

\[ y(s) = (x^{(l)} - G^{(l)})s + G^{(l)}, 0 \leq s \leq 1, \]

and

\[ s^* = \min_i (s_i^*), s_i^* = \frac{g_i^{(l)}}{g_i^{(l)} - x_i}. \]

And then draw a line segment \( z(t) \) between \( a' \) and \( y(s^*) \) and compute its intersection point \( G^{(l+1)} \) with the \( \ell_p \) ball boundary curve \( \|G\|_p = 1 \) using the binary search strategy. Then let \( l = l + 1 \), and go to the next iteration.

7. **End Repeat** when \( \|G^{(l)} - G^{(l-1)}\| < \varepsilon \)

8. Output the projection point \( G = s \odot G^{(l)} \).
A.1 Theoretical Principle

We use step-by-step remarks to explain the theoretical principle underlying our algorithm in detail.

A.1.1 Remarks for steps 1 and 8

**Remark 1:** When $\|a'\|_p > 1$, it is easy to prove that its projection on the $\ell_p$ ball $\|G\|_p \leq 1$ (the solution of (A.1)) is located on its boundary $\|G\|_p = 1$. Furthermore, its projection lies on the same quadrant with $a'$ [208].

**Remark 2:** Due to the symmetry property of the $\ell_2$ objective and the $\ell_p$ constraint of (A.1), we can equivalently solve this optimization problem by getting the solution $G$ for $f(|a'|)$, where $|a'| = s \odot a'$ and $s = sign(a')$ (step 1), and then transfer $G$ (with all positive elements according to Remark 1) back to $G = s \odot G$ (step 8). Here $\odot$ is the Hadamard product meaning the element-wise multiplication between two vectors.

**Remark 3:** When $\|a'\|_p > 1$, since $\|0\|_p < 1$, the intersection of line segment connecting $a'$ and the origin $0$ and the unit $\ell_p$ ball boundary $\|G\|_p = 1$ can definitely be found.

A.1.2 Remark for step 3

**Remark 4:** In the first quadrant, it is evident that the unit $\ell_p$ ball boundary curve $\|G\|_p = 1$ is convex. This means that the tangent plane of this curve at $G^{(l)}$ is below it. See Figure A.1 for better understanding.

![Figure A.1: Principle illustration for Remark 4.](image-url)
A.1.3 Remark for step 4

**Remark 5:** Since \( \|a'|_p > 1 \) and \( \|x(l)\|_p \leq 1 \) (based on Remark 4), the intersection \( G^{(l+1)} \) of the line segment \( z(t) \) (\( 0 \leq t \leq 1 \)) and \( \|G\|_p = 1 \) definitely exists. Since \( x(l) \) is the projection of \( a' \) on \( \pi(l) \) and \( G(l) \) is located on \( \pi(l) \), we have \( \|x(l) - a'|_2 \leq \|G(l) - a'|_2 \). Besides, since \( G(l+1) \) is obtained at \( z(t') \) for certain \( 0 \leq t' \leq 1 \) and \( x(l) \) and \( a' \) are the two end points of \( z(t) \), we have \( \|G(l+1) - a'|_2 \leq \|G(l) - a'|_2 \). It thus holds that \( \|G(l+1) - a'|_2 \leq \|G(l) - a'|_2 \). See Figure A.2 for better understanding.

![Figure A.2: Principle illustration for Remark 5.](image)

A.1.4 Remark for steps 5 and 6:

**Remark 6:** Along the line segment \( z(t) \) (\( 0 \leq t \leq 1 \)) connecting \( a' \) and \( x(l) \), the critical point of its \( i^{th} \) element varying from positive to negative can be calculated by:

\[
(x_i - a_i)t^*_i + a_i = 0 \implies t^*_i = \frac{a_i}{a_i - x_i}.
\]

Then it is evident that the critical point of \( z(t) \) varying out from the first quadrant at \( t^* = \min_{i} (t^*_i) \).

We then have:

(i) When \( \|z(t^*)\|_p \leq 1 \), since \( \|a'|_p > 1 \), the intersection of \( z(t) \) (\( 0 \leq t \leq 1 \)) and \( \|G\|_p = 1 \) exists in the first quadrant. Thus we can use binary search to find this intersection point. Based on the similar proof as Remark 5, we have \( \|G(l+1) - a'|_2 \leq \|G(l) - a'|_2 \). Please see Figure A.3 for better understanding.

(ii) When \( \|z(t^*)\|_p > 1 \), we know that \( z(t^*) \) is not inside the \( \ell_p \) ball \( \Omega = \{G | \|G\|_p \leq 1 \} \). Since \( \Omega \) in the first quadrant is convex (equivalent to that its boundary curve \( \|G\|_p = 1 \) in the first quadrant is convex) and \( a' \in \Omega \), it holds that the entire line segment \( z(t) \)
(0 ≤ t ≤ 1) is in $\overline{\Omega}$ and has no intersection with the curve $\|G\|_p = 1$ in the first quadrant. We thus utilize the following strategy to find the next iteration point.

By connecting the last iteration point $G(l)$ and the projection point $x(l)$, we can formulate a line segment $y(t)$ (0 ≤ t ≤ 1). Using the similar strategy like (i), we can find the critical point $y(s^*)$ at which $y(t)$ goes out from the first quadrant, where

$$s^* = \min_i (s_i^*), s_i^* = \frac{g_i^{(1)}}{g^{(1)} - x_i},$$

where $g_i^{(1)}$ and $x_i$ are the $i$th element of $G(l)$ and $x(l)$, respectively. Since both $y(s^*)$ and $G(l)$ are on the tangent plane $\pi(l)$, and $y(s^*)$ is closer to the projection point $x(l)$ of $a'$ than $G(l)$, we have that $\|y(s^*) - a'\|_2^2 \leq \|G(l) - a'\|_2^2$.

Since $\pi(l)$ is below the curve $\|G\|_p = 1$ based on Remark 4, we know that $\|y(s^*)\|_p \leq 1$. Then together with $\|a'\|_p > 1$, it holds that the intersection $G(l+1)$ of the line segment connecting $a'$ and $y(s^*)$ and $\|G\|_p = 1$ definitely exists in the first quadrant, and $\|G(l+1) - a'\|_2^2 \leq \|y(s^*) - a'\|_2^2$. We thus have $\|G(l+1) - a'\|_2^2 \leq \|G(l) - a'\|_2^2$. The aforementioned can be easily understood by observing Figure A.4.

Based on the aforementioned Remarks 4, 5 and 6, we know that during the iterative process of our algorithm, the objective $\|G(l) - a'\|_2^2$ is monotonically decreasing with
respect to the iteration number $l$ under the constraint $\|G^{(l)}\|_p \leq 1$. Our algorithm is thus convergent and expected to get a rational local minimum of the original problem.
Bibliography


Bibliography


[161] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyunghae Lee, Larry Davis,


