

**VISUAL DETECTION AND
TRACKING OF
UNKNOWN NUMBER OF OBJECTS
WITH
NONPARAMETRIC CLUSTERING
METHODS**

by

İbrahim Saygın Topkaya

Submitted to
the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

SABANCI UNIVERSITY

June 2016

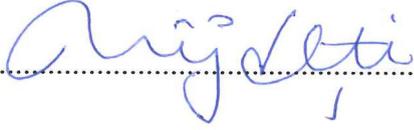
VISUAL DETECTION AND TRACKING OF UNKNOWN NUMBER OF
OBJECTS WITH NONPARAMETRIC CLUSTERING METHODS

APPROVED BY:

Assoc. Prof. Dr. Hakan Erdoğan
(Thesis Supervisor)



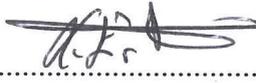
Assoc. Prof. Dr. Müjdat Çetin



Prof. Dr. Berrin Yanıkoğlu



Assoc. Prof. Dr. Erchan Aptoula



Assoc. Prof. Dr. Metin Sezgin



DATE OF APPROVAL:

02 / 06 / 2016

©İbrahim Saygın Topkaya 2016
All Rights Reserved

To my grandmother, Şükran Coşkuntuna...

Acknowledgements

I would like to express my sincere gratitude to my advisor Assoc. Prof. Dr. Hakan Erdoğan and co-advisor Prof. Dr. Fatih Porikli for their endless guidance, support, advices and encouragement throughout my thesis.

I am also grateful to my thesis committee members; Assoc. Prof. Dr. Müjdat Çetin, Prof. Dr. Berrin Yanıkođlu, Assoc. Prof. Dr. Erchan Aptoula and Assoc. Prof. Dr. Metin Sezgin for their valuable advices and time.

My deepest gratitude goes to my family for their endless support; my wife Ceren Aydın Topkaya, my mother Nilufer Topkaya and my brother Çađın Topkaya. This work would be impossible without them.

I would also like to thank all members of VPA Laboratory; I am very happy and proud to be a part of this great family.

Finally, special thanks go to my friend Dr. Til Bartel for encouraging me during my study.

VISUAL DETECTION AND TRACKING OF UNKNOWN NUMBER OF OBJECTS
WITH NONPARAMETRIC CLUSTERING METHODS

İBRAHİM SAYGIN TOPKAYA

EE, Ph.D. Thesis, 2016

Thesis Supervisor: Hakan Erdoğan

Keywords: Nonparametric clustering, Dirichlet process mixture models, Chinese restaurant process, multiple object tracking, people counting.

Abstract

Clustering methods that do not expect the number of clusters to be known a priori and infer the number of clusters are known as nonparametric clustering methods in the literature. In this thesis we propose novel approaches to common computer vision applications using nonparametric clustering. We attack the problems of multiple object tracking and people counting. Our main motivation is to approach those as data association tasks where we define the data association problem specific to the nature of the application and benefit from the nonparametric nature of the clustering model. We first propose a detection free tracking method which tracks an unknown number of objects by clustering superpixels. We define the clusters as targets with spatial and visual features and track their changes through time by sequential clustering. The clusters yield tracked targets through time. We also propose a method for clustering short track segments into unknown number of tracks. The clustering similarity is defined using the spatio-temporal features of the short track segments. The clustering process yields robust tracks of objects through time. We use this approach also to improve the tracking results of the detection free tracking proposed before. Finally we cluster raw person detector outputs to obtain groups of people in a scene and estimate the number of people inside a cluster using the features already extracted for clustering with a proposed metric which is invariant to perspective distortion.

PARAMETRİK OLMAYAN KÜMELEME YÖNTEMLERİ İLE
BİLİNMEYEN SAYIDA NESNENİN GÖRSEL TESPİT VE TAKİBİ

İBRAHİM SAYGIN TOPKAYA

EE, Doktora Tezi, 2016

Tez Danışmanı: Hakan Erdoğan

Anahtar Kelimeler: Parametrik olmayan kümeleme, Dirichlet süreç karışım modelleri, Çin restoranı süreci, çoklu nesne takibi, insan sayımı

Özet

Kümeleme işlemi sonucunda elde edilecek kümelerin sayısının girdi olarak verilmesini gerektirmeyen ve küme sayısını da tahmin edebilen yöntemler literatürde parametrik olmayan kümeleme yöntemleri olarak anılmaktadır. Bu tezde yaygın bilgisayarla görü problemlerine parametrik olmayan kümeleme yöntemlerinden faydalanan çözümler sunuyoruz. Özellikle çoklu nesne takibi ve insan sayımı uygulamalarına çözüm getirmeye çalışıyoruz. Bu konudaki ana motivasyonumuz, bahsi geçen konuları veri ilişkilendirme işlemi olarak ele almak ve veri ilişkilendirme problemini üzerinde çalıştığımız konunun doğasına özgü bir şekilde tanımlayarak ilgili kümeleme yönteminin parametrik olmayan yapısından faydalananmaya çalışmaktır. İlk olarak tespit temelli olmayan ve bilinmeyen sayıda nesneyi süpernokta kümeleyerek takip eden bir nesne takibi yöntemi sunuyoruz. Uzaysal ve görsel özniteliklerini tanımladığımız hedef kümelerin, zaman içerisinde değişen özniteliklerini ardışık olarak kümeleyerek takip ediyoruz. Elde edilen kümeler zaman içerisinde takip edilen hedefleri vermektedir. Ek olarak kısa takip izlerini bilinmeyen sayıda ize kümeleyen bir yöntem sunuyoruz. Kümeleme benzerliği kısa izlerin uzay-zamansal öznitelikleri kullanılarak tanımlanmaktadır. Kümeleme süreci nesnelerin zamanla değişen izlerini gürbüz bir şekilde vermektedir. Bu yöntemden aynı zamanda tespit temelli olmayan takip sonuçlarını iyileştirmek için de faydalanıyoruz. Son olarak da ham insan tespiti sonuçlarını kullanarak insan grupları elde ediyoruz ve her grup içerisindeki insan sayısını kümeleme için kullandığımız hazır öznitelikleri kullanarak, perspektiften bağımsız bir ölçü kullanarak tahmin ediyoruz.

Contents

Acknowledgements	v
Abstract	vi
Özet	vii
List of Figures	xi
List of Tables	xiv
Abbreviations	xv
1 Introduction	1
1.1 Multiple Object Tracking	1
1.2 People Counting	5
1.3 Motivation	7
1.4 Contributions of This Thesis	8
1.4.1 Multiple Object Tracking With Clustering	8
1.4.2 People Counting As Detection Clustering	9
1.5 Organization of This Thesis	10
2 Background and Preliminaries	12
2.1 Introduction	12
2.2 Nonparametric Clustering	12
2.2.1 Dirichlet Process Mixture Models	13
2.2.1.1 Finite Mixture Models	13
2.2.1.2 Infinite Mixture Models	14
2.2.1.3 Chinese Restaurant Process	15
2.2.1.4 Clustering with DPMM	18
2.2.1.5 Gibbs Sampling	18
2.2.1.6 Sampling for Finite Mixture Models	18
2.2.1.7 Sampling for DPMM	19
2.2.1.8 A Synthetic Data Example	20
2.2.2 Distance Dependent Chinese Restaurant Process	22
2.2.2.1 Similarity and Assignment	23

2.2.2.2	Cluster Likelihood	24
2.2.2.3	Clustering with ddCRP	26
2.2.2.4	A Synthetic Data Example	26
2.3	Multiple Visual Object Tracking	28
2.3.1	Tracking by Detection	28
2.3.2	Detection Free Tracking	30
3	Multiple Object Tracking by Superpixel Clustering	32
3.1	Introduction	32
3.2	Related Work and Motivation	33
3.3	Foreground Superpixels as Observations	34
3.3.1	Superpixel Extraction	34
3.3.2	Background Modeling	35
3.4	Observation and Target Models	38
3.5	Target Assignment and Tracking with Clustering	39
3.5.1	Tracking Hypotheses	39
3.5.2	Transition Probabilities	41
3.5.3	Hypothesis Pruning	42
3.6	Post-Processing and Output Representation	42
3.6.1	Grouping Targets into Objects	42
3.6.2	Refining Object Boundaries	45
3.7	Experiments and Results	47
3.7.1	Visual Tracking Results	48
3.7.2	Quantitative Tracking Results	49
3.7.3	Sensitivity Analysis Results	52
3.8	Discussions and Future Work	54
4	Robust Multiple Object Tracking by Tracklet Clustering	55
4.1	Introduction	55
4.2	Related Work and Motivation	56
4.3	Tracklet Generation	57
4.3.1	Detections and Tracklet Models	57
4.3.2	Assigning Object Detections to Tracklets	57
4.3.3	Filtering False Object Detections and Outlier Tracklets	59
4.3.3.1	False Object Detections	59
4.3.3.2	Outlier Tracklets	60
4.4	Tracklet Clustering With Distance Dependent CRP	62
4.4.1	Extracting Tracklet Features for Clustering	64
4.4.2	Tracklet Similarity Function	64
4.4.3	Cluster Likelihood	65
4.4.4	Sampling Tracklet Assignments	66
4.4.5	Output Representation of Tracks	68
4.5	Experiments and Results	68
4.5.1	Visual Results	69
4.5.2	Output Representation	71
4.5.3	Quantitative Results	71
4.5.4	Sensitivity Analysis	72

4.5.5	Running Speed	74
4.6	Improving Detection Free Superpixel Tracking	74
4.6.1	Motivation	74
4.6.2	Extracting The Tracklets	75
4.6.3	Clustering To Obtain Complete Tracks	76
4.6.4	Improved Detection Free Tracking Results	76
4.7	Discussions and Future Work	77
5	People Counting by Clustering Person Detector Outputs	79
5.1	Introduction	79
5.2	Related Work and Motivation	80
5.3	Extracting Person Detector Outputs For Observations	80
5.3.1	HOG Person Detector	80
5.3.2	Aggregating Detections from Consecutive Frames	81
5.3.3	Filtering False Detections	82
5.4	Observation And Cluster Models	83
5.4.1	Color and Spatial Features	83
5.4.2	Temporal Features	84
5.4.3	Cluster Model	85
5.4.4	Cluster Likelihood	86
5.5	Sampling Target Assignments and Clustering	87
5.6	Inferring the Number of People from Clusters	87
5.7	Learning The α Parameter	89
5.8	Experiments and Results	93
5.8.1	Visual Results	94
5.8.2	Quantitative Results	95
5.8.3	Running Time	96
5.9	Discussions and Future Work	97
6	Conclusion and Future Work	98
6.1	Future Work	99

List of Figures

2.1	Graphical model of DPMM.	15
2.2	An example of CRP with 3 existing tables with 5, 8 and 3 customers respectively and 17th customer arriving.	17
2.3	Example synthetic data (a) and clusters (b)-(d) for different values of the α parameter, where a cluster is represented by the isocontour ellipse of its spatial components for $\sigma = 3$	22
2.4	An example of ddCRP with 2 existing clusters (a), and how they change after assignment of 3rd observation (b) and 6th observation(c).	25
2.5	Example synthetic data (a) and clusters (b) obtained with ddCRP.	26
2.6	Three sample frames from a video sequence with outputs of person detections denoted with red rectangles (upper row) and track associations with trajectories denoted with blue traces (lower row).	29
2.7	Symbolic representation of tracking using tracklet extraction and grouping where detections (a), tracklets (b), groups of tracklets (c), and full tracks (d) are presented.	30
2.8	A sample frame (a), optical flow vectors (b), foreground blobs (c), and tracking results (d).	31
3.1	An example frame (a), superpixel borders for the bottom left part of the frame (b), the foreground probability map for each pixel (c) and centers of superpixels that contain foreground pixels (d).	36
3.2	Original targets (a) and grouped targets (b) for an example frame where a target is represented by the isocontour ellipse of its spatial components for $\sigma = 1$	44
3.3	A target represented by the isocontour ellipse of its spatial components for $\sigma = 1$ (a), superpixel clustering results represented by superpixel borders (b), MRF labeling results represented by α -shapes boundaries of the pixels labeled as the target where distance condition in Equation (3.18) is not imposed (c) and imposed (d).	45
3.4	Example frames from PETS 2001 dataset where targets are not grouped and represented by the isocontour ellipse of their spatial components for $\sigma = 1$	47
3.5	Example frames from PETS 2009 dataset where targets are not grouped and represented by the isocontour ellipse of their spatial components for $\sigma = 1$	48
3.6	Example frames from PETS 2001 dataset where targets are grouped as presented in Section 3.6.1 and represented by the isocontour ellipse of their spatial components for $\sigma = 1$	49

3.7	Example frames from PETS 2009 dataset where targets are grouped as presented in Section 3.6.1 and represented by the isocontour ellipse of their spatial components for $\sigma = 1$	50
3.8	Example frames from PETS 2001 dataset where targets are grouped as presented in Section 3.6.1 and represented by α -shapes boundaries of the pixels refined and labeled as the target with MRF as presented in Section 3.6.2.	51
3.9	Example frames from PETS 2009 dataset where targets are grouped as presented in Section 3.6.1 and represented by α -shapes boundaries of the pixels refined and labeled as the target with MRF as presented in Section 3.6.2.	52
4.1	Three detections depicted with blue rectangles that appear only for a single frame. After three such false positives, the detections from this area are immediately filtered out; where two such example detections are depicted with red rectangles.	59
4.2	An example tracklet that lasts four frames. The detections that constitute the tracklet are shown in blue and each one is much larger than its neighbors which are drawn in red (every 50th detection is drawn).	60
4.3	Five example frames belonging to a tracklet. The detections that constitute the tracklet are shown in blue and each one has much less number of neighbors (which are drawn in red) than the total number of frames so the tracklet is filtered out.	61
4.4	An example tracklet, the detections constitute which are shown with red rectangles. Each detection intersects with a larger detection so the <i>smaller</i> tracklet is eventually filtered out.	62
4.5	Example frames from PETS 2009 dataset with tracklet clustering results, i.e., each trajectory on the frame corresponds to a <i>cluster</i> of tracklets.	69
4.6	Example frames from PETS 2009 dataset without any clustering, i.e., each trajectory on the frame corresponds to a tracklet.	70
4.7	Example frames from SPEVI Frontal dataset with tracklet clustering results, i.e., each trajectory on the frame corresponds to a <i>cluster</i> of tracklets.	70
4.8	Example frames from SPEVI Frontal dataset without any clustering, i.e., each trajectory on the frame corresponds to a tracklet.	71
4.9	Graphical user interface that summarizes the tracking results per object together with entry/exit times.	71
4.10	Sensitivity analysis results for the α parameter by analyzing the change of MOTA with different parameter values.	74
4.11	A tracking sequence without (a-c) and with (d-f) track termination.	76
5.1	HOG detections with different scales separately (a)-(i) and all detections for a frame with different scales together (j).	81
5.2	Detections from an example frame itself (a), from the previous frame (b), next frame (c) and detections from the frame itself and shifted detections from the previous and next frames together (d).	82
5.3	Foreground map of a scene (a) and HOG based person detections (b), where detections filtered out because of size are depicted with yellow and because of foreground probability are depicted with red borders. Blue represents the final remaining detections.	83

5.4	Two HOG detection areas with their optical flow vectors (a) and (b); their corresponding HOOF features with four bins (c) and (d).	85
5.5	Detections from an example frame with the keypoints inside the detection area (a)-(g) and a single cluster with all of the keypoints inside the cluster (h).	88
5.6	Example frames with HOG detections (first row) and clusters with estimated number of people (second row) for PETS 2009 dataset.	92
5.7	Example frames with HOG detections (first row) and clusters with estimated number of people (second row) for PETS 2009 dataset.	93
5.8	Example frames with HOG detections (first row) and clusters with estimated number of people (second row) for BEHAVE dataset.	94
5.9	Example frames with HOG detections (first row) and clusters with estimated number of people (second row) for Peds2 dataset.	95

List of Tables

3.1	Comparative results for tracking by superpixel clustering on PETS 2001 and PETS 2009 datasets.	51
3.2	Tracking results for tracking by superpixel clustering on PETS 2001 and PETS 2009 datasets with different superpixel sizes.	53
3.3	Tracking results for tracking by superpixel clustering on PETS 2001 and PETS 2009 datasets with and without noise.	53
4.1	Comparative results for tracking by tracklet clustering on PETS 2009, TownCentre, TUD Stadmitte, ETH and SPEVI datasets.	73
4.2	Comparative results for tracking by hierarchical superpixel clustering on PETS 2009 and TUD Campus datasets.	77
5.1	Comparative people counting results with MAE and MRE values for PETS 2009 dataset.	96
5.2	Comparative people counting results with MAE and MRE values for UCSD Peds2 dataset.	97

Abbreviations

CRP	C hinese R estaurant P rocess
ddCRP	d istance d ependent C hinese R estaurant P rocess
DPMM	D irichlet P rocess M ixture M odel
GMM	G aussian M ixture M odel
HOG	H istogram of O riented G radients
HOOF	H istogram of O riented O ptical F lows
MCMC	M arkov C hain M onte C arlo
MRF	M arkov R andom F ields
SMC	S equential M onte C arlo

Chapter 1

Introduction

Being an important set of methods in data analysis, clustering methods use the similarities of data elements and group them into meaningful subsets called clusters in the context of the application. Nonparametric clustering methods are the type of clustering methods that do not expect the number of clusters to be known a priori and infer the number of clusters as a part of the clustering process. In this work we employ nonparametric clustering methods to propose novel approaches to common computer vision applications, specifically *multiple object tracking* and *people counting*.

For instance, we propose a detection free tracking method which tracks an unknown number of objects by clustering superpixels where the clusters are defined as targets with spatial and visual features which change (especially spatially) in small steps through time. By our definition sequential clustering actually yields tracked targets through time. We also propose a tracker which clusters short track segments into unknown number of tracks where the clustering similarity is defined using spatio-temporal features of short track segments. The similarity measures and constraints that we define on the clustering process yields clusters which are actually robust tracks of objects through time.

1.1 Multiple Object Tracking

In this section we start with a review of basic Bayesian approaches to single object tracking and further review multiple object tracking problem and existing methods for solving it. We focus specifically on visual object tracking with single camera setups.

Tracking a single target emitting a single observation at each time step can be considered as one of the most basic scenarios in object tracking. In this type of tracking, the uncertainty in the tracking arises from the implicit noise in the target movement and the noise in the observation acquisition process as well as false and missed detections.

Denoting the observation at time t with y_t and the real and unknown state of the object at time t with x_t , in the most general sense the object tracking problem is to estimate the most probable set of states at each time, i.e., $\hat{X} = \{\hat{x}_1 \dots \hat{x}_t \dots\}$, given the set of observations, i.e., $Y = \{y_1 \dots y_t \dots\}$, which can be written as:

$$\hat{X} = \operatorname{argmax}_X p(X|Y). \quad (1.1)$$

A fundamental approach in this type of situation is employing the Kalman filter [1] which relies on the assumption that the problem can be modeled as a linear state-space system with independent Gaussian noise. Under this assumption, the Kalman filter gives the optimal estimate of the target with least mean square error.

Strict assumptions of the Kalman filter can be relaxed by extended Kalman filter [2] which assumes the states and observations can be modeled with differentiable functions rather than linear models and unscented Kalman filter [3] which samples state estimates around the mean and propagates them with nonlinear functions. Particle filters [4] handle the problem in the most relaxed case, where there are no prior assumptions on the state and observation models and they are sampled at each time step with a sequential Monte Carlo (SMC) sampler.

An improvement for particle filters is the so called Rao-Blackwellization [5] process which yields Rao-Blackwellized particle filters. In this type of filters, if the problem components can partly be defined as a linear model, it is proposed to solve those parts of the problem with regular Kalman filter equations, but perform particle filters in nonlinear parts.

Apart from the observation and target scheme introduced above, multiple object tracking aims to track multiple targets emitting multiple observations, which introduces the additional problem of data association; the process of deciding which observation belongs to which target. If the simple case of single observation per target is considered, data association problem can be handled as an additional step of assigning observations to targets where the number of observations need not be equal to the number of targets

due to occlusion and clutter. Revising Equation (1.1) for multiple targets yields the same likelihood where each element of $\hat{\mathbf{X}}$ at each time, i.e., $\hat{\mathbf{x}}_t$ and each element of \mathbf{Y} at each time, i.e., \mathbf{y}_t denotes a set of observations and states at that time—as opposed to single values. Specifically $\hat{\mathbf{x}}_t = \{\hat{\mathbf{x}}_t^1, \hat{\mathbf{x}}_t^2 \dots\}$ and $\mathbf{y}_t = \{y_t^1, y_t^2 \dots\}$.

Under the assumption that targets emit single observations, there is at most one observation per target and an observation is generated either from a target or from the clutter. The solutions, known as multiple hypothesis trackers [6][7], follow this assumption and update the states of targets using multiple parallel hypotheses using only the observation associations of the relevant association hypotheses. Another set of trackers, known as joint probabilistic data association [8] based trackers, also follow the same assumption however update the states considering all valid associations. So states of all targets are updated using all observations and the association probability of an observation to an object is actually the sum of all probabilities of that association in all valid observation sets. One significant difference between the two is that multiple hypothesis trackers can handle unknown number of objects naturally since there can be many different hypotheses where the number of objects may vary.

An alternative approach to track the target states is the probability hypothesis density filtering [9] which handles the finite set of all targets. Methods employing probability hypothesis density filters work on the intensity function rather than the states of individual targets where the intensity function is defined on the single target state space. The integral of the intensity function on a subset of the state space yields the expected number of targets in that subset [10] and consecutively local peaks of the intensity function hint the likelihood of target appearance at that part of the state space.

In realistic applications, some of the previously tracked objects may go out of scene temporarily or permanently and new objects that haven't been seen before can enter the scene. Usually so called *birth* and *death* events are handled with defined probabilities and some observations are assigned to new objects with respect to the birth probability, or no observations are assigned to some objects with respect to the death probability. In [11], associations are modeled as an m th order process, so association at each time depends on m previous associations. In the same work, when a new object is born, its life time is associated with a probability and the probability of death of a target at a

time is calculated by using the time elapsed since the last time an observation has been associated to the target and the death probability.

A visual object tracking application of the same approach is presented in [12] where the authors perform people tracking by extracting observations with a person detector and solve the association problem with the approach of [11], as well as perform dynamic selection of motion parameters with the help of Dirichlet process mixture model (DPMM), where they cluster motion parameters.

A possible categorization [13] of multiple object tracking is the choice of initialization. In *tracking by detection* methods, observations are obtained by object detectors specific to the application and then assigned to tracks over time. [14] and [15] are two recent works that can be given as examples, both of which solve the track association problem in a conditional random fields [16] framework. *Detection free tracking* methods, on the contrary, do not rely on individual detections and try to *search* for objects between frames. [17] is an example of this approach where the authors try to cluster foreground pixels at each frame into trajectories using DPMM. [18] and [19] extract and cluster point tracks using optical flow [20] and build a graph to handle the tracking problem in a graph partitioning framework. Although not being completely detection free, [21] is also a very recent work that tries to overcome the artifacts of missed detections by extracting object neutral superpixels and assigning them to tracks sequentially.

Merging short but highly confident sequences of tracks (i.e., tracklets [22]) into longer and more complete tracks is also a common tracking method. In [23] the authors associate tracks to previous ones using an online learned linking model and in [24] and [25] the authors create a similarity matrix between the short tracks and cluster them using k-means [26]. In [27] the authors calculate confidence values for tracklets and assign the ones with low confidence values to the ones with high confidence values.

Network flow based solutions are also usually employed in recent work as in [28] which defines the assignment problem as a cost flow network and tries to find the optimum solution with minimum cost and [29] which, in addition, enforces a spatial constraint to disambiguate nearby targets with similar appearance.

Other recent and interesting work in the multiple object tracking literature include; [30] where the authors handle the observation to target assignment problem in a tensor

optimization framework, [31] which presents a work that employs several binary classifiers that work on detections from consecutive frames, [32] about compensating missed detections caused by occlusions using motion evolution information, [33] on discovering groups of objects and tracking them together, [34] applying clustering to confidence map outputs of object detectors and [35] which handles occlusions caused by colliding people by training detectors for multiple people.

1.2 People Counting

People counting is one of the fundamental yet challenging computer vision tasks that has many applications in a diverse set of fields from video surveillance [36] to business intelligence for retail space management [37]. Different sources of information are used to count people including stereoscopic camera systems [38], infrared cameras [39], optical cameras [40] and even radio signals in Wi-Fi networks [41].

In general, people counting methods that use regular optical cameras and do not employ any holistic approaches like tracking can be categorized into two groups that are based on object detection outputs (i.e., *detection based methods*) or *regression based methods*.

1. Detection based methods infer the number of people in the scene from region classifiers that are designed to locate people, like Histogram of Oriented Gradients (HOG) detector [42], or body parts, like Haar-like features based face detector [43]. For instance, [44] uses a head detector to determine the number of people by applying a classifier that is trained with color and orientation of gradients features around a set of chosen interest points.
2. Regression based methods learn a function of linear or nonlinear correspondences between the image features and the number of people in the training data, and then employ the learned function to estimate the number of people in the test data. For instance, [40], computes a fixed ratio between the number of extracted foreground corners [45] and the number of people. An improved work [46] clusters interest points and trains a regressor on the number of interest points and the number of people in the cluster. A more recent work [47] tries to minimize a cost function to find the optimal correspondence between the features extracted around difference pixels of consecutive frames and the number of people.

The main disadvantage of detection based methods is their sensitivity to occlusions and artifacts caused by imperfect detector responses. Regression based approaches on the other hand, although being quite common in the literature, require different sets of training data with different setups, thus their generality is limited and the training step usually requires high amount of manual input and processing. Additionally, perspective difference in the scene introduces additional challenges to the regression task, since most of the features used in regression (e.g., corners) are not perspective invariant.

To overcome the problems introduced by perspectivity, manual distortion correction can be applied to the scene as done in [48] or the counting process itself can be adapted to handle the issue. For instance, [49] also uses a similar approach to [40], but to take perspective difference into account, partitions the scene into horizontal bands size of which are determined by a specific training procedure that requires the annotation of appearance heights of a single person captured in different parts of the scene. [50] trains the regressor using Gaussian processes [51] on features that are based on perspectively normalized interest points.

Other than interest points, foreground segments are also used as inputs to regressors, like [52] and [53] extracting feature vectors of length 29 from foreground segments and employing regressors based on Gaussian processes and Poisson regression [54] respectively, where the latter is more suitable for regression on integer values.

The rising interest in deep learning also found its place in the people counting literature. A very recent work [55] handles the counting problem in a deep learning framework and proposes to use clusters of convolutional neural network [56] responses to count objects in a scene. [57] also extracts features using convolutional neural networks and trains a sparse classifier [58].

In addition to the methods that work on individual frames, which are reviewed in two groups above, tracking based methods work on the sequence as a whole and estimate the number of people by grouping similar trajectory segments. [59] is an example of such a work, where a model based tracker is used to generate short trajectories, which are grouped into unique tracks per person using spatial and temporal consistency heuristics. [60] is another example work that employs a Lucas-Kanade tracker [61] to extract short trajectories. These methods inherit errors caused by tracking and try to solve a broader

class of problems while performing occlusion free tracking or keeping person identities consistent across frames.

Also it should be noted that, although indirectly, virtually every tracking or content retrieval solution gives an estimate about the number of objects in the video sequence, but the literature of people counting is usually interested in framewise (or frame neighborhood) people counting.

1.3 Motivation

Our main motivation is to approach mentioned problems in computer vision applications as data association tasks that are defined according to the nature of the relevant application and benefit from the nonparametric nature of the employed clustering models. We believe the problems can be handled in a clustering framework because of the association between the entities in computer vision applications and the grouping capability of clustering tasks. We specifically investigate nonparametric clustering methods because the number of clusters, by definition of our applications, are unknown and actually estimating the number of clusters itself is also a crucial part of the applications.

We employ DPMM and distance dependent Chinese Restaurant Process (ddCRP); for their implicit nonparametric nature and advantage of allowing to determine the clusters even when their number is unknown a priori. Although we cover both in the next chapter, at this point we can briefly distinguish the two such that; DPMM tries to infer clusters and assigns observations to those clusters, whereas ddCRP models similarities between the observations and assigns each of them to others and obtains clusters as a byproduct of that process.

We prefer DPMM specifically over other clustering methods such as DBSCAN [62], which also does not require the number of clusters, because instead of requiring a similarity metric between feature vectors, DPMM models the data such that the probabilities of cluster assignments are defined with a mixture model. We go for such a probabilistic mixture model for clusters that can be defined *around* some points in the relevant feature space (e.g., color or spatial). We employ DPMMs in tracking by superpixel clustering because it is suitable to model the superpixels as clusters in spatial feature space and inherit cluster parameters temporally. The reason that we employ DPMMs in people

counting by clustering detector outputs also depends on the fact that spatial and visual features of the multiple detections of the same person are close to the spatial and visual features of the target person and hence close to each other.

The advantage of ddCRP to other nonparametric clustering methods that do not require complex cluster models is its neutrality to the observations; for instance unlike DBSCAN, ddCRP does not define any *core* or *border* points and relies solely on pairwise similarities. We employ ddCRP because of the complex similarity between tracklets make it hard to define a cluster model that can be defined easily. Hence, we exploit pairwise tracklet assignments and obtain clusters as a byproduct of this process.

In addition, a practical advantage of clustering with both DPMMs and ddCRPs is that the overall performance of the clustering can be controlled with a single parameter. This allows us to build overall systems that are robust to the control parameter and do not require complex training or user annotation steps.

1.4 Contributions of This Thesis

We attack two important computer vision problems of object tracking and people counting within nonparametric clustering framework.

1.4.1 Multiple Object Tracking With Clustering

We present completely two different tracking approaches where we employ nonparametric clustering for multiple object tracking.

In the first approach, we perform sequential clustering in image space without using any object detectors. We extract foreground superpixels as observations and cluster them using the set of clusters inherited from the previous frame. The clusters inherited between frames actually denote the temporal sequence of object parts. Our contributions are:

- We use superpixels as atomic observations for clustering; which reduce the number of observations and clustering time.

- We employ a robust GMM based background model and use only foreground superpixels for tracking.
- We integrate historical motion explicitly to the clustering process by initially estimating spatial parameters of clusters before clustering and calculating transition probabilities after clustering to update hypothesis likelihoods.
- We explore the whole association space and track only effective hypotheses by pruning associations or transitions with low probability.
- We refine the object boundaries with MRFs and compensate border artifacts.
- We group clusters to combine different parts of tracked objects into one.

In the second approach, we aim to cluster tracklets and overcome discontinuities caused by occlusion or target ambiguity. To extract tracklets, we employ object detectors, however the detectors are not specific to a specific object class and we demonstrate our results with outputs of different object detectors. Eventually, we obtain a robust and fast object tracker, suitable for stationary single camera setups. Our contributions are:

- We employ ddCRPs to cluster tracklets and need to calculate only pairwise tracklet similarities.
- We do not model explicit cluster models, so we can define complex tracklet features.
- Our method does not require any training and can only be controlled by a single ddCRP parameter.
- We show that tracklet clustering can improve detection free tracking by applying clustering to detection free track segments.

1.4.2 People Counting As Detection Clustering

As reviewed in Section 1.2, there are two main approaches based on object detection or regression for people counting. We present a hybrid system for people counting, that starts with dense outputs of a person detector and clusters them using DPMM. We go for a probabilistic mixture model for assignments of the detections to the clusters, which represent groups of people, because of the nature of the detection process. For example,

spatially, detections for a single person naturally group around the correct location of the person, and other visual features, e.g., color, depict a similar behavior as varying around an average value. In addition, higher values for the single control parameter in a DPMM generate a larger number of clusters, which is preferable for more crowded scenes.

After clusters (i.e., groups of people) are obtained, we try to estimate the number of people in each cluster. Indeed, ideally the expected outcome of the clustering process would be one person in each cluster and obtaining a perfect segmentation as well as counting; however even in semi crowded scenes people interact with each other and it is not easy to distinguish them with crude detectors. Thus, we propose a metric to estimate the number of people in clusters and like most of the regression based methods, it relies on extracted interest points. The estimation is done locally (i.e., for clusters of a few people) so perspective invariance is implicitly preserved.

Our contributions are:

- We use raw detector outputs for people counting.
- We aggregate detections from neighboring frames to compensate misses and occlusions.
- We employ DPMMs to cluster detector outputs and model spatio-temporal features.
- We integrate temporal information into clustering by employing HOOF features.
- We learn the optimal clustering parameter with a practically simple training process.
- We propose a metric to infer the number of people in each cluster.

1.5 Organization of This Thesis

We begin with the mathematical background of DPMM and ddCRP in Chapter 2, where we review the derivation of mathematical models and present their capabilities with synthetic data. We also briefly summarize multiple visual object tracking and the

relationship of the data association problem with it. Next, in Chapters 3 and 4 we present our work on detection free tracking with sequential superpixel clustering and robust tracking by detection with tracklet clustering. In Chapter 5 we present our work on people counting with clustering of detector outputs. Each chapter is presented as a separate work on its own, in which the motivations and conclusions for each work are presented in detail. Finally, in Chapter 6, we present our conclusions and discuss possible future work.

Chapter 2

Background and Preliminaries

2.1 Introduction

In this chapter we review the mathematical models of the family of nonparametric Bayesian clustering models that we use in the presented work; specifically DPMM and closely related Chinese restaurant process (CRP), as well as ddCRP. Following that, we review the definition of visual object tracking where we elaborate the details of detection free and tracking by detection approaches and the task of data association in visual object tracking.

2.2 Nonparametric Clustering

Clustering is the generic name given to the class of data analysis tasks of grouping the elements of a sample of observations into meaningful subsets (i.e., *clusters*), where the elements of each subset is *similar* to each other than the others, according to an application specific definition of *similarity*.

Putting aside all other differences, there are two major groups of clustering methods; where the first group of methods (k-means [26] being the most famous example) assume that the number of clusters is known a priori and expect it to be given as an input and the second group of methods, referred as *nonparametric* clustering methods, do not expect the number of clusters to be known a priori and infer it as a part of the clustering process as well.

In the following chapters we present computer vision applications where we employ nonparametric clustering methods as the essence of our proposed solutions where the nonparametric nature of the employed methods allows us to model the data without explicitly considering the number of underlying clusters.

2.2.1 Dirichlet Process Mixture Models

The idea behind DPMM [63] [64] is to provide a way to model a set of observation data as a mixture of unknown number of distributions.

2.2.1.1 Finite Mixture Models

Starting [65] with the definition of mixture models with known number (K) of mixtures, the model for data $\{x_i | i = 1 \dots N\}$ is given as:

$$p(x_i | \Theta) = \sum_{k=1}^K p(c_i = k) p(x_i | c_i = k, \theta), \quad (2.1)$$

where $p(x_i | c_i = k, \theta)$ is the probability density function for a single mixture component defined by parameters $\theta = \{\theta_1 \dots \theta_K\}$. Introducing the *indicator/assignment* variables, $c_i = k$, denoting $x_i \in k$ i.e., i th observation is assigned to k th component, the model can be detailed as below:

$$\begin{aligned} x_i | c_i, \Theta &\sim p(x_i | c_i = k, \theta), \\ \theta_k | G_0, \\ c_i | w &= (w_1 \dots w_K) \sim \text{Discrete}(w_1 \dots w_K), \\ w | \alpha &\sim \text{Dirichlet}(\alpha_1 \dots \alpha_K), \end{aligned} \quad (2.2)$$

where G_0 is the prior distribution for the parameters of the mixture components and called the *base distribution*, from which the component parameters ($\Theta = \{\theta_1 \dots \theta_K\}$) are sampled. *Dirichlet* is the *Dirichlet distribution*, the conjugate prior of the categorical distribution *Discrete* and probability density function of which is defined for K categories as:

$$\text{Dirichlet}(x_1 \dots x_K | \alpha_1 \dots \alpha_K) = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} \prod_{i=1}^K x_i^{\alpha_i - 1} \quad (2.3)$$

where $\alpha = (\alpha_1 \dots \alpha_K)$ is the parameter vector of the distribution. In practice when Dirichlet distribution is used as a prior for a categorical distribution without any prior knowledge, a symmetric distribution is employed [65] where all α_i values are equal, thus the whole distribution is defined with a single α value.

Using Equation (2.2) and Equation (2.3), the prior for indicator variables can be rewritten in terms of the symmetric Dirichlet distribution as [65]:

$$p(c_1 \dots c_N | \alpha) = \frac{\Gamma(\alpha)}{\Gamma(N + \alpha)} \prod_{j=1}^K \frac{\Gamma(n_j + \alpha/K)}{\Gamma(\alpha/K)} \quad (2.4)$$

and fixing only for one c_i yields [65] [66]:

$$p(c_i = j | c_{-i}, \alpha) = \frac{n_{-i,j} + \alpha/K}{N - 1 + \alpha}, \quad (2.5)$$

where c_{-i} is the set of all indicator variables other than i , $n_{-i,j}$ is the number of observations assigned to the j th component before assignment of i and N is the total number of observations.

2.2.1.2 Infinite Mixture Models

Taking the limits when K goes to ∞ , Equation (2.5) yields the following for components which have observations assigned to them:

$$\lim_{K \rightarrow \infty} p(c_i = j | c_{-i}, \alpha) = \frac{n_j}{N - 1 + \alpha} \quad (2.6)$$

and the sum of all other remaining (of infinite) components is:

$$\lim_{K \rightarrow \infty} \sum_{\forall n_j=0} p(c_i = j | c_{-i}, \alpha) = \frac{\alpha}{N - 1 + \alpha}. \quad (2.7)$$

Equation (2.6) and Equation (2.7) differ only on n_j and α , that is, the more observations are assigned to a cluster, the more probable a new observation will be assigned to it. This generalization of Dirichlet distribution to infinite number of components is called *Dirichlet process* and the yielded model is called Dirichlet process mixture model (DPMM), where the mixture parameters are defined by the base distribution prior G_0 and the Dirichlet process parameter α , thus usually denoted as $DP(\alpha, G_0)$.

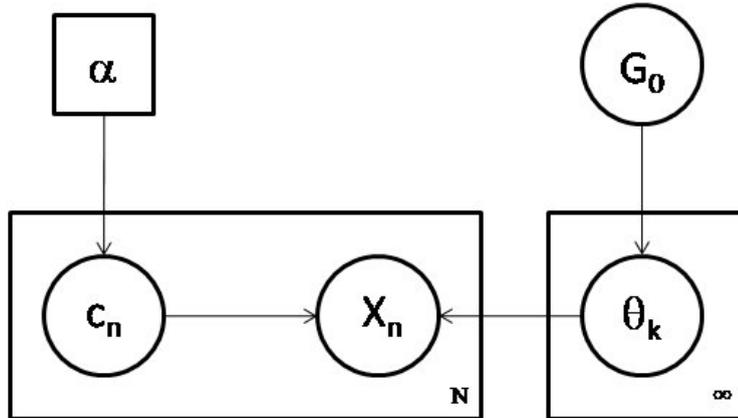


FIGURE 2.1: Graphical model of DPMM.

In practice, infinite number of mixture components allow to model data with an unknown number of mixture components where only a small subset with a finite number of elements have components with data assigned to them. This flexibility is the key idea about nonparametric clustering within a probabilistic framework, where each observation (x_i) depends on one of the infinite number of components (θ_k) through the indicator variable (c_i) which depends on the Dirichlet process parameter α . The graphical model of DPMM is presented in Figure 2.1.

The α parameter in Equation (2.6) and Equation (2.7) controls the probability that an observation will be assigned to an existing cluster with other observations or will be assigned to a new cluster. The probability that the observation will be assigned to an existing cluster is proportional to the number of observations already assigned to that cluster ($\propto n_j$) and the probability that the observation will be assigned to a new cluster is proportional to a fixed value ($\propto \alpha$). Increasing α results in more clusters with fewer observations whereas decreasing α results in fewer clusters with more observations.

2.2.1.3 Chinese Restaurant Process

The *infinite number of components* assumption bears the *Chinese Restaurant Processes* analogy [67], where a Chinese Restaurant with an infinite number of tables without any capacity limit is considered. Each new customer (i.e., N th customer), chooses to sit with uniform probability at a designated chair next to one of the $N - 1$ existing customers or at a new empty table. The new customer is assigned to the table of the existing

customer whom he sat next to. We can denote the choice of the new customer with m_N , which takes values in $\{1 \dots N\}$. $m_N = N$ means that the customer will sit at a new table and $m_N \in \{1 \dots N - 1\}$ means that the customer will sit at the chair next to an existing customer. The uniform probability of the case that the new customer will sit next to an existing customer is:

$$p(m_N \in \{1 \dots N - 1\}) = \frac{1}{N - 1 + \alpha}, \quad (2.8)$$

and the probability of the case that the new customer will sit at a new table is:

$$p(m_N = N) = \frac{\alpha}{N - 1 + \alpha}, \quad (2.9)$$

which is same as Equation (2.7)). The sum of the probabilities for existing $N - 1$ customers and a new table is:

$$\begin{aligned} p(m_N = N) + p(m_N \in \{1 \dots N - 1\}) &= \frac{\alpha}{N - 1 + \alpha} + \sum_{i=1}^{N-1} \frac{1}{N - 1 + \alpha} \\ &= \frac{\alpha}{N - 1 + \alpha} + \frac{N - 1}{N - 1 + \alpha} \\ &= 1, \end{aligned} \quad (2.10)$$

and the total probability that the N th customer sitting at a particular table j , with n_j number of customers already sitting at that table, is

$$\sum_{i=1}^{n_j} \frac{1}{N - 1 + \alpha} = \frac{n_j}{N - 1 + \alpha}, \quad (2.11)$$

which is same as Equation (2.6).

In summary, the probability that the new customer will sit at an already occupied table is proportional to the number of customers already sitting at that table ($\propto n_j$) and the probability that the new customer will sit at a new table is proportional to a fixed value ($\propto \alpha$). Increasing α results in more occupied tables with fewer customers or few tables with more customers vice versa. Since each customer can sit at one table, the customers are *partitioned* across tables (clusters).

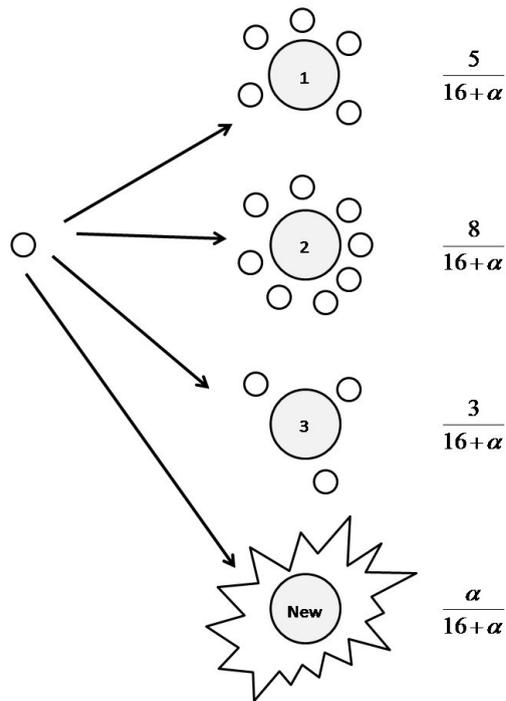


FIGURE 2.2: An example of CRP with 3 existing tables with 5, 8 and 3 customers respectively and 17th customer arriving.

In Figure 2.2, we present an example for CRP where there are 3 tables with 5, 8 and 3 customers already sitting at them. 17th customer arrives to the restaurant and chooses to sit at one of those 3 tables or a new table with presented probabilities. The value of the α parameter determines the probabilities at this point; a high α value like 100 yields the result that the new customer will sit at a new table with a probability value of 0.86 whereas lower α values like 10 and 1 will yield the result that the new customer will sit at a new table with probability values of 0.38 and 0.06 respectively. Note that, these are prior probability of cluster assignments not considering observations x_i . Posterior probability of assignments will be influenced with the likelihood of x_i being assigned to clusters.

We will revisit the Chinese restaurant analogy in Section 2.2.2 for a different nonparametric clustering model.

2.2.1.4 Clustering with DPMM

Let $\{x_i | i = 1 \dots N\}$ be the data to be clustered using DPMM, and assumed to be distributed according to the model in Equation (2.1) for $K = \infty$. As reviewed in Section 2.2.1.2, DPMM model assumes an infinite number of mixture components exist, but with only a finite subset of these components having data assigned to them. Thus the task of clustering with DPMM involves finding the parameters of those finite and unknown number of mixture components. [66] reviews Markov chain Monte Carlo (MCMC) methods [68] such as Gibbs sampling [69] to iteratively sample assignment probabilities of observations to the unknown number of underlying mixture components, as well as sample the mixture probabilities simultaneously.

2.2.1.5 Gibbs Sampling

The idea in Gibbs sampling is to sample variables with a joint probability distribution (e.g., $(x_1 \dots x_N) \propto p(x_1 \dots x_N)$) by sampling from the marginal distribution of variables (e.g., $x_i \propto p(x_i | x_1 \dots x_{i-1}, x_{i+1} \dots x_N)$) rather than sampling from the joint distribution.

The sampling algorithm begins with initial or random values of marginals and samples marginals for each variable one by one iteratively. The initial samples are usually discarded and after a certain number of iterations (referred as *burn-in period*) the distribution is assumed to reach equilibrium and samples, believed to be proportional to the real probabilities, are obtained. In Algorithm 1 we review Gibbs sampling algorithm -regardless of DPMM- briefly.

2.2.1.6 Sampling for Finite Mixture Models

We review the sampling method [70] for finite mixture models, derivation of which is presented also in [71]. We are searching the marginal probability of a single assignment variable c_i for a given single observation x_i , the assignments for other observations c_{-i} , the mixture component parameters Θ and the Dirichlet model parameter α :

$$p(c_i = j | x_i, c_{-i}, \Theta, \alpha) \propto p(c_i = j | c_{-i}, \Theta, \alpha) \times p(x_i | c_i = j, c_{-i}, \Theta, \alpha), \quad (2.12)$$

Input: $p(x_i|x_1 \dots x_{k-i}, x_{i+1} \dots x_N) \forall i$

Result: Samples of $x_1 \dots x_N$

```
// Optionally permute  $x_1 \dots x_N$  randomly
// Optionally assign initial or random values to marginals

// Repeat for #Gibbs iterations
for  $k \in \{1 \dots \#Gibbs\}$  do
  for  $i \in \{1 \dots N\}$  do
    // Sample  $i$ th element in  $k$ th iteration
    Sample  $x_i^k \propto p(x_i|x_1^k \dots x_{k-i}^k, x_{i+1}^{k-1} \dots x_N^{k-1})$ 
  end
end
end
```

Algorithm 1: Gibbs sampling algorithm

since the posterior (i.e., $p(c_i = j|x_i \dots)$) is proportional to the multiplication of the prior (i.e., $p(c_i = j|\dots)$) and the likelihood (i.e., $p(x_i|c_i = j \dots)$). Furthermore, the likelihood term $p(x_i|c_i = j, c_{-i}, \Theta, \alpha)$ depends only on the parameters of the j th mixture component (i.e., θ_j), and the assignment prior for c_i when the others (i.e., c_{-i}) are fixed depends only on the parameter α which is already given in Equation (2.5). Thus we can write Equation (2.12) as:

$$p(c_i = j|x_i, c_{-i}, \Theta, \alpha) \propto \frac{n_j + \alpha/K}{N - 1 + \alpha} \times p(x_i|\theta_j). \quad (2.13)$$

2.2.1.7 Sampling for DPMM

If we consider [70][72][73] infinite number of mixture components (i.e., $K \rightarrow \infty$), the assignment prior takes the values in Equation (2.6) and Equation (2.7):

$$p(c_i = j|c_{-i}, \alpha) = \begin{cases} \frac{n_j}{N-1+\alpha} & \text{for existing } j \\ \frac{\alpha}{N-1+\alpha} & \text{for new } j+1 \end{cases}, \quad (2.14)$$

and the likelihood for the existing mixture components (i.e., the ones which already have observations assigned to them) is $p(x_i|\theta_j)$ whereas the sum of all other remaining infinite components is calculated by integrating over all mixture component parameters,

yielding:

$$p(x_i | c_i = j, c_{-i}, \Theta, \alpha) = \begin{cases} p(x_i | \theta_j) & \text{for existing } j \\ \int_{\theta} p(x_i | \theta) dG_0(\theta) & \text{for new } j + 1 \end{cases} \quad (2.15)$$

Combining Equation (2.14) and Equation (2.15) as in Equation (2.12) yields the following sampling probabilities:

$$p(c_i = j | x_i, c_{-i}, \Theta, \alpha) \propto \begin{cases} \frac{n_j}{N-1+\alpha} \times p(x_i | \theta_j) & \text{for existing } j \\ \frac{\alpha}{N-1+\alpha} \times \int_{\theta} p(x_i | \theta) dG_0(\theta) & \text{for new } j + 1 \end{cases}, \quad (2.16)$$

which is also presented as *Algorithm 2* in [66]. Again, the α parameter in Equation (2.16) controls the probability that an observation will be assigned to an existing cluster with other observations or will be assigned to a new cluster. Therefore α parameter can be used to control ultimately the number of clusters.

Here we would like to briefly discuss the calculation of the integral in Equation (2.16), which integrates over possible samplings of the parameter set Θ from the base distribution G_0 . A common case where DPMM is employed is the Gaussian mixture model (GMM) with unknown number of Gaussian mixture components. Each mixture component is defined by two parameters; the mean vector and the covariance matrix, i.e., $\theta : (\mu, \Sigma)$. The prior for Gaussian parameters is the normal-inverse Wishart distribution and integrating over it results in a Student-t distribution [74]. However [74] also shows that the Student-t distribution can be replaced with a Gaussian distribution with proper parameters.

The overall Gibbs sampling and clustering algorithm with DPMM is presented in Algorithm 2.

2.2.1.8 A Synthetic Data Example

We would like to end reviewing of DPMM by presenting an example on clustering of visual data. We generate a random dataset of point observations in 2D space using a generative process such that observations are concentrated *around* 10 underlying clusters

```

Input:  $X = \{x_1 \dots x_N\}$ 
Input: DPMM parameter  $\alpha$ 
Result: Clusters  $\Theta = \{\theta_1 \dots \theta_K\}$ 
Result: Assignments  $c = \{c_1 \dots c_N\}$ 

// Optionally permute  $x_1 \dots x_N$  randomly
// Set initial assignments, e.g.,  $c_i = i \ \forall i$ 

// Repeat for #Gibbs iterations
for  $g \in \{1 \dots \#Gibbs\}$  do
  for  $i \in \{1 \dots N\}$  do
    //  $k$  is current number of clusters

    // Remove  $x_i$  from  $c_i$  and update parameters of  $\theta_{c_i}$ 
     $c_i \leftarrow c_i - x_i$ 
     $\theta_{c_i} \leftarrow \theta_{c_i} - x_i$ 

    Sample  $c_i \propto p(c_i = j | x_i, c_{-i}, \Theta, \alpha)$  // Using Equation (2.16)
    if  $c_i = k + 1$  then
      // To a new cluster
      Init  $\theta_{k+1}$  with  $x_i$ 
    else
      // To an existing cluster
      Update  $\theta_{c_i}$ 
    end
  end
end

```

Algorithm 2: Gibbs sampling algorithm for DPMM

with random deviations in position and color. The generated random data is presented in Figure 2.3(a).

We run the clustering algorithm for DPMM presented in Algorithm 2 with 50 Gibbs iterations using different values for the α parameter in Equation (2.16). We model the cluster likelihood with multidimensional Gaussians for position (x and y) and color (r , g , b) components, specifically:

$$\begin{aligned}
 p(x_i | \theta_j) = & \mathcal{N}(x_x | \mu_x^j, \sigma_x^j) \times \mathcal{N}(x_y | \mu_y^j, \sigma_y^j) \times \\
 & \mathcal{N}(x_r | \mu_r^j, \sigma_r^j) \times \mathcal{N}(x_g | \mu_g^j, \sigma_g^j) \times \mathcal{N}(x_b | \mu_b^j, \sigma_b^j).
 \end{aligned} \tag{2.17}$$

In Figure 2.3(b) $\alpha = 1$ results in 10 underlying clusters, where a smaller value ($\alpha = 1e-5$) yields less clusters in Figure 2.3(c) and a larger value ($\alpha = 1e2$) yields more clusters in Figure 2.3(d).

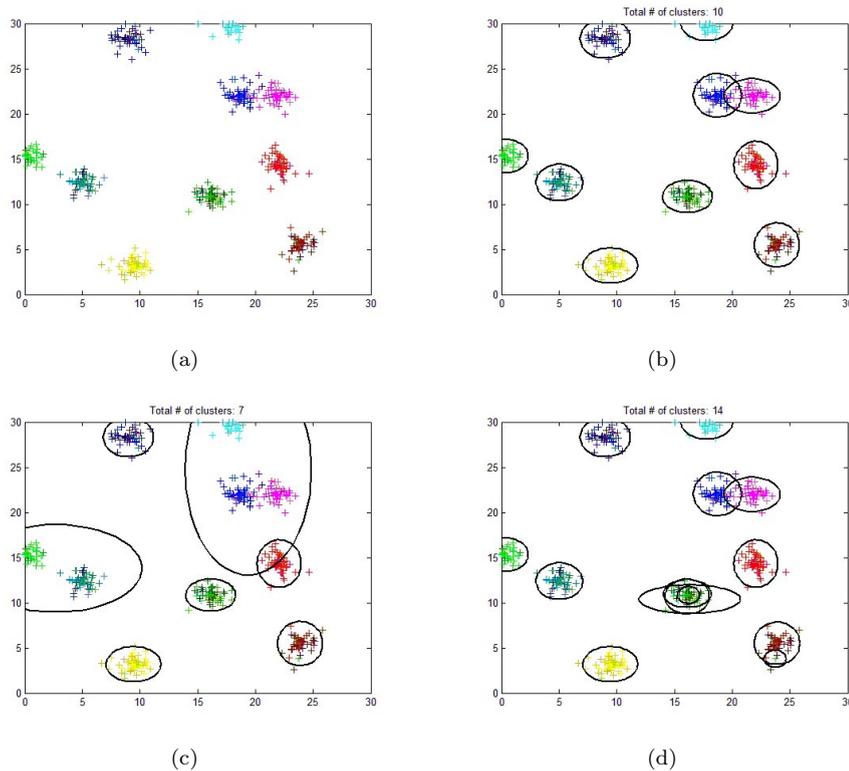


FIGURE 2.3: Example synthetic data (a) and clusters (b)-(d) for different values of the α parameter, where a cluster is represented by the isocontour ellipse of its spatial components for $\sigma = 3$.

2.2.2 Distance Dependent Chinese Restaurant Process

Continuing with the Chinese restaurant analogy presented in Section 2.2.1.3, we now review ddCRP [75]. In CRP, the similarity between customers are not taken into consideration and the probability that a new customer sitting down with another customer is uniform (i.e., $1/(N - 1 + \alpha)$). In fact, eventually, CRP is interested in the probabilities between customers and tables (Equation (2.11)) rather than between individual customers. This is compatible with DPMM (e.g., Equation (2.14)) where assignment of observations to clusters is considered and CRP acts as an assignment prior accompanying the likelihood (as in Equation (2.16)). This, of course, requires a cluster model to be defined (and updated after each assignment) as well as the cluster likelihood function.

ddCRP, on the other hand, seeks assignments between customers only. In ddCRP analogy, a new arriving customer chooses to sit down with an existing customer (consecutively at the same table) with a *nonuniform* probability or by itself at a new table. Thus, customers that choose to sit down together, either directly or indirectly through

another customer, constitute a table. In clustering context, observations are assigned to each other and the ones that are assigned to each other directly or indirectly through others, constitute a cluster.

2.2.2.1 Similarity and Assignment

ddCRP was proposed in [75] raising the issue of *exchangeability*. In DPMM and CRP model, it does not matter in which order the observations are handled; since assignment prior of one observation to another is uniform and one observation to a cluster is proportional only to the number of observations assigned to that cluster before, there is no way to integrate the relationship between the observations to the clustering process. In other words, in CRP, the observations are exchangeable, meaning the order of the processing of the observations does not matter.

In ddCRP model, the relationship between the observations can be integrated into the clustering process and, for instance the ordering of the observations can be emphasized. An example presented in [75] is processing of temporal observations and giving high similarities to observations temporally close to each other. Our motivation for employing ddCRP is the ease of modeling similarities between the observations and integrating it into the clustering process without modeling complex cluster models. In Chapter 4, we employ ddCRP to cluster tracklets with complex similarities which would not be possible with a mixture model such as CRP or DPMM that covers such a diverse range of features.

Instead of placing a prior on the assignments of observations to clusters (i.e., Equation (2.14)), ddCRP replaces the prior on assignments of observations to each other. For instance assignment prior of observation i to observation j is denoted as:

$$p(c_i = j | c_{-i}, \alpha, F) \propto \begin{cases} F(i, j) & i \neq j \\ \alpha & i = j \end{cases}, \quad (2.18)$$

where $c_i = j$ denotes that x_i is *linked* to (thus *assigned* to the same cluster with) x_j and the assignment prior is proportional to $F(i, j)$, which is the similarity measure between these two observation indices. Proportional to α , the observation is not assigned to any other observation but to itself. Note that the assignment prior does not depend on

other assignments but only on similarities between observation indices. The similarity function $F(i, j)$ is defined between observation indices. Often a decay function of an exponential form [75], which is applied on the distance, is employed.¹ Obviously, this is totally application specific.

The clusters are formed as a byproduct of assignments of observations to others. Observations assigned to each other directly or indirectly constitute a cluster and a single change in those assignments can change the overall cluster structure through directly or indirectly assigned observations.

Consider the example in Figure 2.4 where on the left 6 observations with assignments and on the right cluster interpretations of them are presented. The observations form 2 clusters in Figure 2.4(a) per the assignments. We now review the example cases where assignments for observation 3 and 6 are sampled.

Assume that $p(c_3)$ is sampled as 3, i.e., 3rd observation is assigned to itself. Since there was no other observation assigned to 3rd observation it is removed from the cluster it was at before and alone constitutes a new cluster. The resulting cluster structure is presented in Figure 2.4(b).

After $p(c_3) = 3$ had been sampled, assume that $p(c_6)$ is sampled as 2, so that 6th observation is assigned to 2nd observation. The result of the new assignment is that 6th observation is removed from the cluster that it was at before and moves to the same cluster with 2nd observation. In addition to that, since 5th observation had been assigned to 6th observation, it is also considered in the same cluster with 2nd observation from now on. The resulting cluster structure is presented in Figure 2.4(c).

Again, we must note again that cluster structure is only a byproduct interpretation of observation assignments and observations are not *moved* between clusters actually.

2.2.2.2 Cluster Likelihood

The likelihood of assignment for observation x_i is given by:

$$p(x_i | c_i = j, c_{-i}) = p(X | z(\{c_i \cup c_{-i}\})), \quad (2.19)$$

¹Even though the formulation does not explicitly allow for using the observations x_i in calculating the above probability, in our experiments we assumed that we can calculate $F(i, j)$ based on x_i and x_j since we can always assume an oracle giving this information to us before starting clustering.

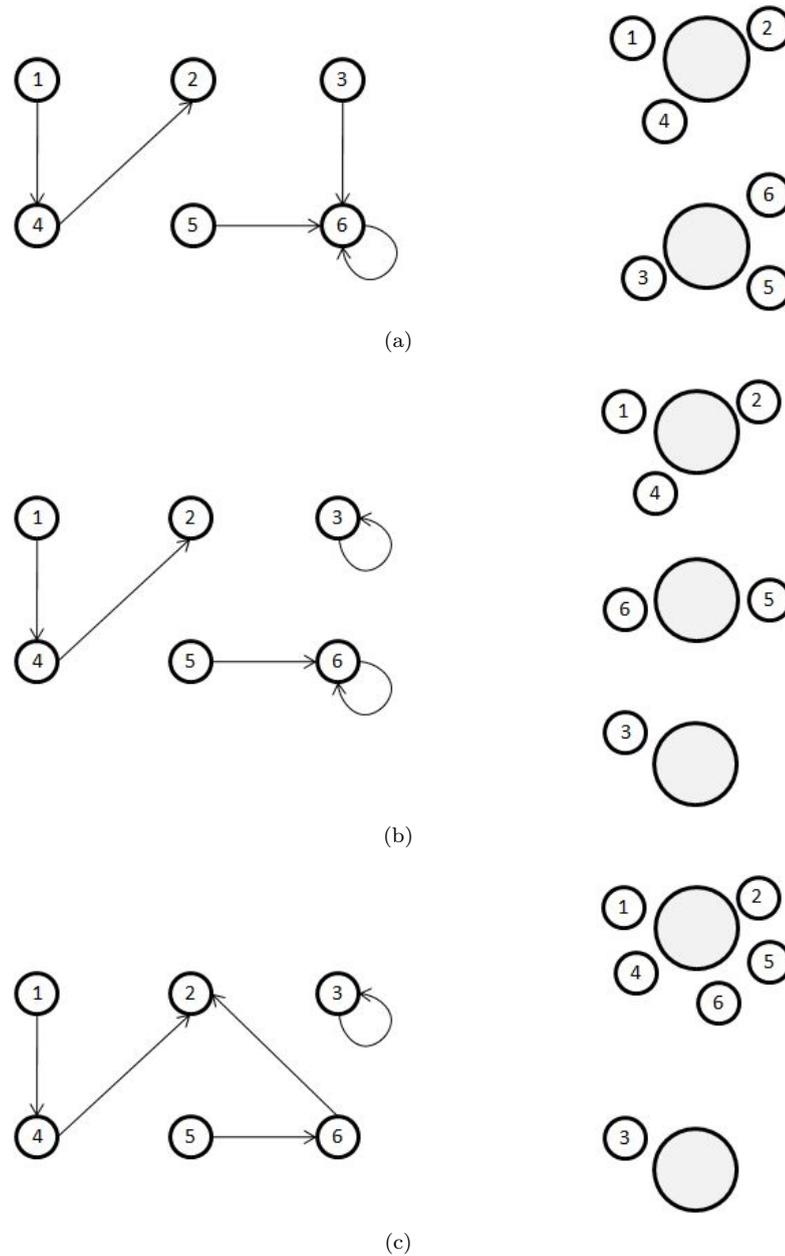


FIGURE 2.4: An example of ddCRP with 2 existing clusters (a), and how they change after assignment of 3rd observation (b) and 6th observation(c).

which is the overall likelihood of all observations (i.e., $X = \{x_1 \dots x_N\}$) under the new set of clusters obtained after sampling the assignment of x_i , i.e., c_i , denoted by $z(\{c_i \cup c_{-i}\})$.

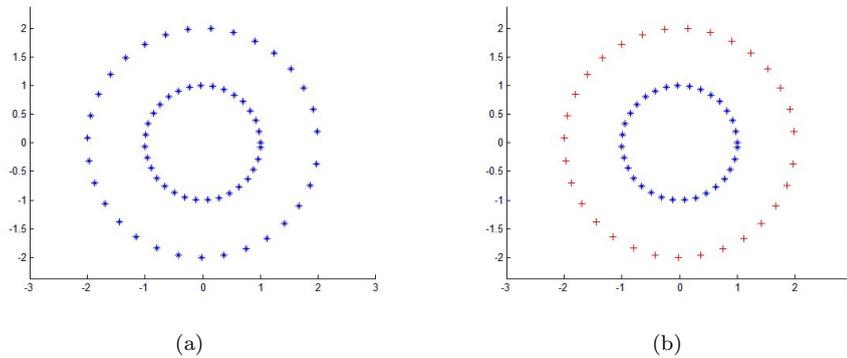


FIGURE 2.5: Example synthetic data (a) and clusters (b) obtained with ddCRP.

2.2.2.3 Clustering with ddCRP

Let $\{x_i | i = 1 \dots N\}$ be the data to be clustered using ddCRP, the task of clustering with DPMM involves finding the unknown number clusters by sampling assignments of observations to other observations. The Gibbs sampling can be employed to sample those assignments for each observation and to define the sampling probability, as Equation (2.16) combines Equation (2.14) and Equation (2.15), we can combine Equation (2.18) and Equation (2.19) as:

$$p(c_i = j | X, c_{-i}, \alpha, F) \propto \begin{cases} F(i, j) & i \neq j \\ \alpha & i = j \end{cases} \times p(X | z(\{c_i \cup c_{-i}\})) \quad (2.20)$$

where, notice that no set of explicit cluster parameters are employed like Θ as in Equation (2.16).

The modified Gibbs sampling (per Equation (2.20)) and clustering algorithm with ddCRP is presented in Algorithm 3.

2.2.2.4 A Synthetic Data Example

As we did in Section 2.2.1.8 for DPMM, we would like to end reviewing of ddCRP by presenting an example on clustering of visual data as well. We generate a random dataset of point observations in 2D space which is presented in Figure 2.5(a) where two clusters of the two circles centered on the origin are clearly visible.

Input: $X = \{x_1 \dots x_N\}$
Input: Similarity function F
Input: ddCRP parameter α
Result: Assignments $c = \{c_1 \dots c_N\}$
Result: $C(i) \forall i$, where $C(i)$ denotes the cluster that x_i belongs to

```

// Set initial assignments, e.g.,  $c_i \leftarrow i, C(i) \leftarrow \{i\} \forall i$ 

// Repeat for #Gibbs iterations
for  $g \in \{1 \dots \#Gibbs\}$  do
  for  $i \in \{1 \dots N\}$  do
    Sample  $c_i \propto p(c_i = j | X, c_{-i}, \alpha, F)$  // Using Equation (2.20)
    if  $c_i = i$  then
      |  $C(i) \leftarrow i$  // Assigned to itself
    else
      |  $C(i) \leftarrow C(c_i)$  // To another observation
    end
    // Update assignments for all
    for  $x_j \in X$  do
      |  $C(x_j) \leftarrow C(x_{c_j})$ 
    end
  end
end
end
  
```

Algorithm 3: Gibbs sampling algorithm for ddCRP

We run the clustering algorithm for ddCRP presented in Algorithm 3 with only 5 Gibbs iterations. We model the similarity between observations in Equation 2.20 as:

$$F(i, j) = e^{-\|x_i - x_j\|}, \quad (2.21)$$

which is basically the exponential of the Euclidean spatial distance between the two observations. The similarity enforces nearby observations to be assigned to each other with a higher likelihood. In order to enforce the clusters to capture the circles, we penalize the cases where the observations that constitute a cluster are not circular by employing the variance of the distance of the observations, that constitute a cluster, to the origin. Thus, we define the cluster likelihood in Equation 2.20 as:

$$p(X | z(\{c_i \cup c_{-i}\})) \propto \prod_k e^{-\mathcal{G}(k)}, \quad (2.22)$$

where k iterates over all clusters at each sampling step and the function \mathcal{G} is defined as:

$$\mathcal{G}(k) = \begin{cases} 1000 & \forall c_x = k; \text{Var}(\|x\|) > 1e - 5 \\ 10 & \text{otherwise} \end{cases} \quad (2.23)$$

which penalizes clusters with a relatively high variance of the distance of the assigned observations to the origin to enforce the circular smoothness. We assign a nonzero value even to the likelihood of the clusters with a low variance to control the number of clusters, since the product in Equation (2.22) decreases by the number of clusters.

In Figure 2.5(b) we present the clustering results where clusters are distinguished with unique colors and shapes where the two clusters are separated successfully.

2.3 Multiple Visual Object Tracking

Visual object tracking refers to the problem of tracking objects in a scene captured with a regular camera. At each time, a single frame representing the field of view of the camera is captured. The universal unit of digital imaging is the *pixel* (a portmanteau of the words *picture* and *element*) which corresponds to the uniform spatial samples taken from the scene. The uniform sampling of the pixels do not carry any information related to regions, objects, boundaries or anything other than color or intensity values at each independent spatial point. This eventually brings an additional step of object detection to the tracking process. As presented in the introduction, the two major approaches are tracking by detection and detection free tracking.

2.3.1 Tracking by Detection

Without loss of generality, object detectors find the regions of interest on the frame, which are likely to enclose an object of interest. In tracking by detection, object detection and tracking are two separate processes. First, objects of interest are detected at each frame and they are associated with tracks through time, which is the main problem of interest in this approach. Each detection is -almost always- represented by a point in the feature space which is constituted of a broad set of features (e.g. location, size, color) obtained by the visual data acquisition process. Then at a new frame, each detection is:

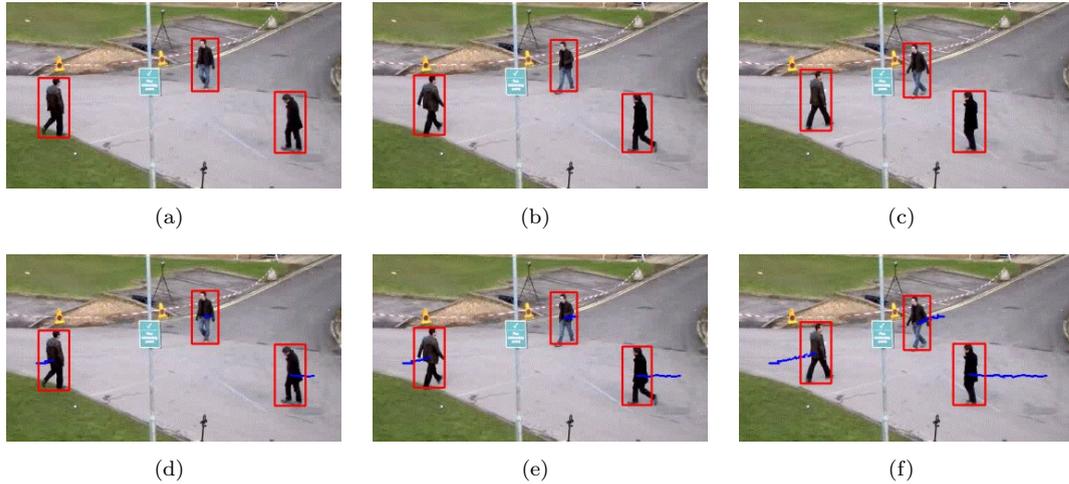


FIGURE 2.6: Three sample frames from a video sequence with outputs of person detections denoted with red rectangles (upper row) and track associations with trajectories denoted with blue traces (lower row).

- Assigned to one of the existing tracks, or
- Used to initialize a new track, or
- Considered as clutter

An example of tracking by detection is presented in Figure 2.6 where the upper row presents the outputs of an object (person) detector. At each frame, the detection results are extracted first and associated with the nearest track as presented in the lower row.

Extracting full tracks is not easy under imperfect data acquisition conditions like occlusion, clutter, and missed object detections which may lead to prematurely terminated tracks or drifts. A common approach to overcome these obstacles is to extract short but reliable tracks (i.e., tracklets) and group them into complete tracks afterwards. This is usually a two level process, where in the first level detections are associated with tracklets as explained above, but the tracklets are terminated if there is ambiguity in the association. The second level employs a grouping step and tracklets are grouped into full tracks.

A symbolic representation of the tracking process with tracklets is presented in Figure 2.7 where the detection results are represented with dots on image frame and collapsed in time in Figure 2.7(a). In Figure 2.7(b) the detections are associated with tracklets and tracklets are depicted by trajectories with directions. Figure 2.7(c) contains groups of

tracklets where each group is presented with a unique color and finally in Figure 2.7(d) output representation of full tracks are presented.

Tracklet extraction is usually straightforward where the detections are associated with tracklets in a greedy fashion using the affinities of the detection results with the tracklets with respect to usual features like size, location, and color. An additional data association problem is introduced by tracklet extraction since grouping of the tracklets requires modeling complex tracklet features and association (of tracklets to tracks) likelihoods. For instance, in Figure 2.7(c), tracklets with the same colors mean that they are associated with the same complete track in the final grouping step.

2.3.2 Detection Free Tracking

Detection free tracking refers to the tracking process where there is no object detection step is employed. Not employing a specific object detector is useful if different classes of objects are required to be tracked or it's not feasible (e.g., computationally expensive) to train detectors for different types of objects that are required to be tracked.

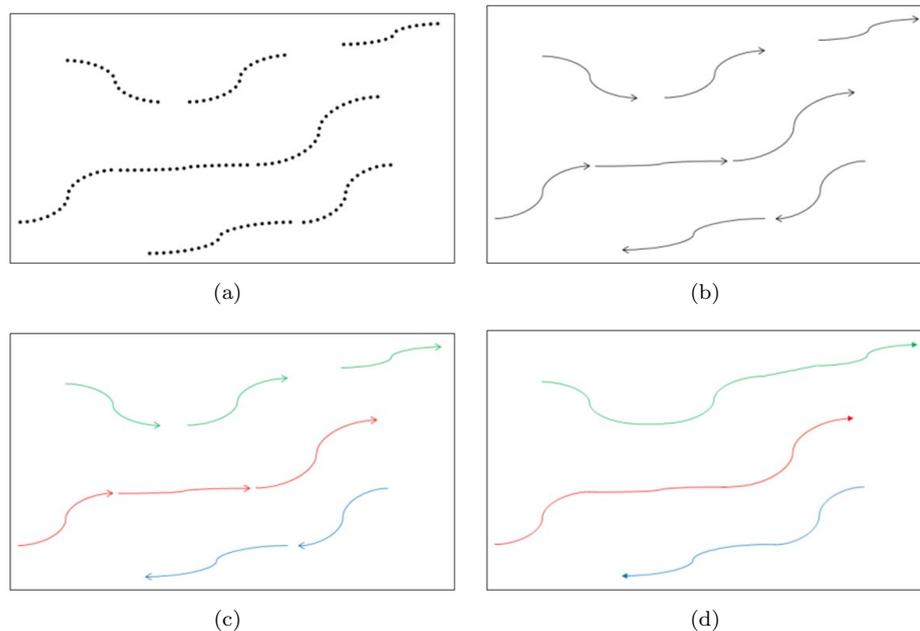


FIGURE 2.7: Symbolic representation of tracking using tracklet extraction and grouping where detections (a), tracklets (b), groups of tracklets (c), and full tracks (d) are presented.



FIGURE 2.8: A sample frame (a), optical flow vectors (b), foreground blobs (c), and tracking results (d).

Usually points or regions of interest are extracted in the frame and trajectory information around the extracted points or regions is obtained. A classic example of detection free tracking in this sense is the famous optical flow [61][76] algorithm which extracts uniform vectors of motions for small regions of a frame using the derivative of image intensity between consecutive frames. Since each region in frame f is associated with a displacement vector, it is eventually associated with a region in frame $f + 1$, hence tracked.

However there is still an outstanding association problem that is required to be solved, to associate optical flow vectors with distinct objects. A very naive and frame level solution [77] to this association problem is to obtain blobs by morphological operations and track those extracted blobs. The steps for a sample frame is presented in Figure 2.8, which is the output of the sample code in [77]. Beginning with the original acquired frame (Figure 2.8(a)), displacement vectors for small regions of an image are extracted (Figure 2.8(b)) which give the location of those regions in the next frame. Then foreground blobs are obtained (Figure 2.8(c)) which is used to encapsulate motion vectors with distinct objects (Figure 2.8(d)).

Chapter 3

Multiple Object Tracking by Superpixel Clustering

3.1 Introduction

In this chapter, we present our work [78] on a multi-object tracker for unknown number of objects which is based on nonparametric clustering. Our tracking framework employs DPMM (Section 2.2.1) within a sequential tracker that keeps assignments of multiple observations to multiple targets between frames while maintaining multiple parallel assignment and eventually tracking hypotheses.

Our framework enables detection and tracking of an unknown and variable number of objects in a fully automatic fashion without any initial labeling. Since no constraints on the number of clusters is required by the DPMM, we can track hypotheses of unknown number of clusters at the same time. At each frame, we extract foreground superpixels and cluster them into objects and track by propagating clusters across consecutive frames.

Within the scope of this chapter and DPMM clustering framework, we use the following terms for the following entities: *observations* for any atomic observation (specifically in our case, foreground superpixels) to be associated to a target, *targets* for *clusters* (used interchangeably) of observations obtained by DPMM tracking, *objects* for tracked objects that are formed by one or more targets, and *hypotheses* for tracking hypotheses that define historical target states and observation to target associations.

3.2 Related Work and Motivation

DPMM caught attention very recently in sequential object tracking applications. [79] is an example in speech processing domain where the authors track unknown number of formants in the spectrum through time. The samples in the spectrum are handled as observations and clustering hypotheses of those samples at each time step are inferred with a particle filter. The clustering hypotheses are generated sequentially for each time with DPMM, where the authors explore the whole association space of observations to clusters and keep the top best hypotheses to use in the clustering process for the observations in the next time step.

In [17] authors perform object tracking for an unknown number of objects. They take simple difference of color values of pixels between two consecutive frames and obtain a set of pixels (having significant change in color values) as the set of observations. Then, they cluster those pixel observations into objects using color and spatial features using DPMM. The work presents two inference methods for DPMM using SMC and MCMC sampling. SMC inference (referred as SMC with local Gibbs iterations) employs a particle filter and for each frame, samples clustering hypotheses and maintains parallel tracking hypothesis between frames. The second inference method (referred as particle MCMC) performs batch inference over sequence level tracks with an initial estimate obtained with SMC which, according to the authors, yields better results than SMC.

In this work we also follow a sequential clustering approach and obtain clustering hypotheses at each frame and maintain those clustering hypotheses between frames. Instead of sampling frame level assignments or sequence level tracks as in [17], we explore the whole frame level association space inspired by [79] and aim to track online all *feasible* associations at each frame. In addition, we select observations using a GMM based background modeling algorithm as opposed to arbitrarily using all image pixels or simple frame differencing like [17] and in order to reduce the number of tracked association hypotheses, we employ superpixels as atomic observations.

By incorporating superpixels and employing an efficient tracking hypothesis pruning scheme, we keep the total number of hypotheses low and tractable. After obtaining tracking hypotheses for unknown number of targets using superpixels, we refine pixel level boundaries using MRF that incorporates spatial statistics of the clusters/targets

obtained during the previous tracking stages into the refinement process. Finally, we carry out a temporal grouping of clusters to combine -potentially- different parts of the same tracked object into one.

3.3 Foreground Superpixels as Observations

3.3.1 Superpixel Extraction

Since our work in this chapter does not incorporate any prior object information into the detection and tracking process, the only information unit we have is the pixels on the scene. We aim to avoid using raw pixels as observations within the DPMM clustering framework, for two reasons. First, using the high number of pixels and trying to cluster them all increases the computational complexity of the task. Second, considering our ultimate objective of assigning observations to objects, the high number of pixels carry highly redundant information for this clustering task, since the assumption that most of the neighboring pixels belong to the same object is highly rational.

Superpixel extraction is a modern [80] image segmentation technique that aims to partition the image into small (as opposed to the whole objects), uniform (keeping within color or intensity variation low) and natural (taking borders into consideration) regions in an efficient (as opposed to high number of pixels) representation. The two desirable properties [81] of a superpixel extraction scheme are; obtaining perceptually meaningful areas while reducing the model complexity for further processing.

Thus, to reduce the number of observations and decrease the DPMM observation to target association time, we incorporate superpixels that segment the image into small, compact and almost uniform regions while keeping color variation within regions low. Simple linear iterative clustering (SLIC) [82] is a very recent superpixel extraction technique, that works with few parameters, in linear ($O(n)$) complexity and on $5D$ image space; spatial x , y and three channels of the Lab color space since it is a perceptually uniform color space [82]. The distance measure in this $5D$ space between two entities i

and k is defined as:

$$D = d_{lab} + \frac{\lambda_p}{s_p} d_{xy}, \quad (3.1)$$

$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2},$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2},$$

where s_p is the parameter that defines the sizes of the cells that the algorithm initially divides the image into and λ_p is a regularizer parameter to control the importance of spatial proximity against color uniformity. After initially dividing the image into uniform cells of size s_p , the center of each cell is taken as the location at which the image gradient, i.e.:

$$G(x, y) = \|I(x + 1, y) - I(x - 1, y)\|^2 + \|I(x, y + 1) - I(x, y - 1)\|^2, \quad (3.2)$$

where $I(x, y)$ is the Lab color vector at position (x, y) and $\|\cdot\|$ is the L_2 norm, is lowest. Then each pixel in the image is associated with the nearest (with respect to Equation (3.1)) cell center and cell centers are updated as the average $5D$ vector of the associated pixels. This process is repeated iteratively until the update of the cell centers are less than a threshold value.

With a small computational overload, SLIC superpixel extraction significantly decreases the number of observations by around 95% compared to the number of pixels; an example of which can be seen in Figure 3.1(a) and Figure 3.1(b).

3.3.2 Background Modeling

The background modeling approach proposed by Stauffer-Grimson [83], handles each pixel as a multidimensional color (e.g., red green blue) random vector X . The vectors are supposed to be generated by a mixture of Gaussians. Each mixture component (i.e., Gaussian) allows to capture the deviations for a background object over time and since a pixel can capture different objects over time, more than one Gaussian are used. The GMM defines the likelihood of the random vector X for a single pixel using a weighted sum of K number of Gaussians:

$$p(X) = \sum_{j=1}^K w_j \times \mathcal{N}(X, \mu_j, \sigma_j), \quad (3.3)$$

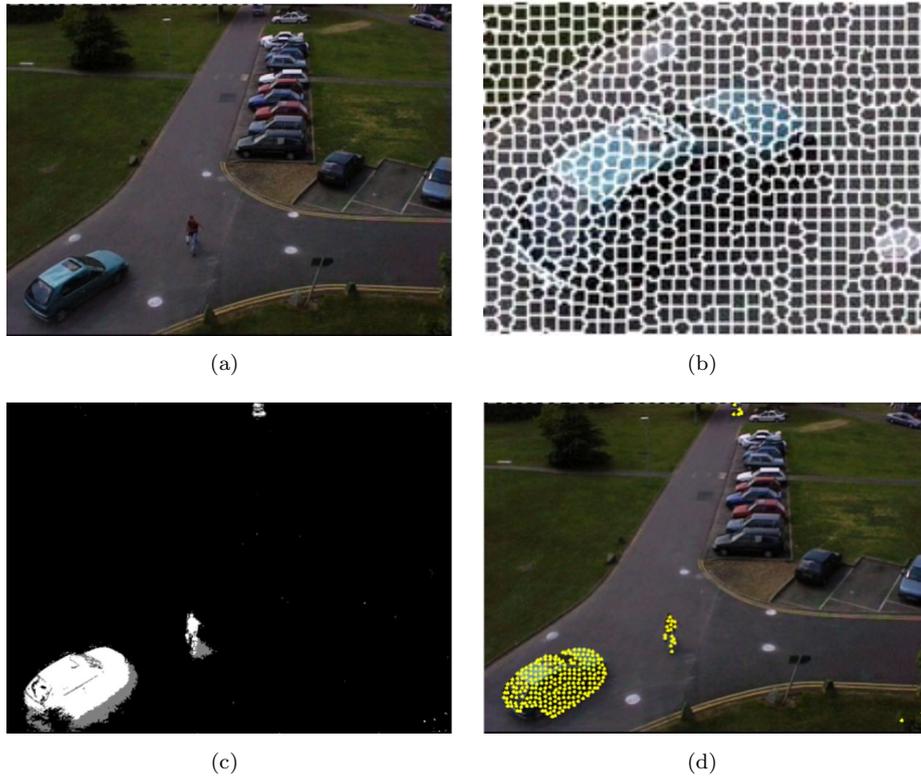


FIGURE 3.1: An example frame (a), superpixel borders for the bottom left part of the frame (b), the foreground probability map for each pixel (c) and centers of superpixels that contain foreground pixels (d).

where the parameters of the distributions (μ_j and σ_j), as well as the weight of each Gaussian component (w_j), are initialized with fixed values and updated dynamically using the obtained pixel values (X) over time with:

$$\begin{aligned}
 w_j &= w_j + \alpha(o_x - w_j), \\
 \mu_j &= \mu_j + o_x(\alpha/w_j)\delta_j, \\
 \sigma_j^2 &= \sigma_j^2 + o_x(\alpha/w_j)(\delta_j^2 - \sigma_j^2),
 \end{aligned} \tag{3.4}$$

where $\delta_j = \mu_j - X$, α is a constant parameter and o_x is the *ownership* indicator which is 1 for the Gaussian that X belongs to and 0 for the others. The obtained pixel value X belongs to j th Gaussian if $\delta_j^2/\sigma_j^2 \leq 3$ for the smallest value of δ_j . If an obtained pixel does not belong to a Gaussian, a new one is initialized and added to the mixture. A slight modification to the update rule of the weight values is introduced by [84], such

that:

$$w_j = w_j + \alpha(o_x - w_j) - \alpha c_T, \quad (3.5)$$

where the introduced negative bias term c_T is used to discard mixture components with a negative weight after the update.

In any time, there exists a Gaussian mixture for each pixel, which can be used to infer the background likelihood of the pixel using Equation (3.3). Since some of the Gaussians in the mixtures may be generated by temporary foreground objects, only B Gaussians, cumulative sum of which are lower than a fixed threshold, are assumed to belong to the background:

$$p(X_i | X_i \in \text{Background}) = \sum_{j=1}^B \left(\frac{w_j}{\sum_{k=1}^B w_k} \right) \times \mathcal{N}(X_i; \mu_j, \Sigma_j), \quad (3.6)$$

where B is calculated by:

$$B = \operatorname{argmin}_b \left(\sum_{i=1}^b w_i (1 - c_f) \right), \quad (3.7)$$

where c_f is a parameter to control when a *new* object (or a new color value for a particular pixel) can blend into the existing background model. Considering α in Equation (3.4) and c_f in Equation (3.7), this is approximately $\log(1 - c_f) / \log(1 - \alpha)$ number of frames [84].

Equation (3.6) can be used to calculate the probability that the obtained pixel's feature vector X_i belongs to the background using the Bayes rule:

$$p(X_i \in BG | X_i) = \frac{p(X_i | X_i \in BG) \times p(X_i \in BG)}{p(X_i, X_i \in BG) + p(X_i, X_i \in FG)}. \quad (3.8)$$

We extract foreground regions on the frame using an implementation of this [84] GMM based background representation that models the previous color changes of each pixel using a mixture of Gaussians. As observations to cluster with DPMM for a frame, we select superpixels that contain pixels whose foreground probability is higher than a specified threshold (κ_f).

The number of observations by the superpixel extraction and background suppressing is drastic. For the example frame shown in Figure 3.1, the number of total pixels is $\sim 160,000$ and only $\sim 4,500$ of them are foreground pixels with probabilities higher than κ_f . Finally we end up with only around 200 foreground superpixels (in Figure 3.1(d)) which is a much more suitable number of observations to cope with.

3.4 Observation and Target Models

We model each observation (i.e., foreground superpixel) X using the spatial center of the superpixel (i.e., x and y pixel coordinates) and the mean value of the a and b pixel color components in the Lab color space. Similarly, we model each cluster/target using the mean and variance of the same components along with the spatial covariance. For computational ease, we do not model Gaussian models with full covariance matrices for targets and we only analyze the covariance between spatial components (i.e., in Σ_{xy}), since it is strongly related to the appearance of the target on the image by approximating the target appearance with a rotated ellipse. Thus, each observation is defined with four parameters; $X : (\mu_x, \mu_y, \mu_a, \mu_b)$ and each target with six parameters; $\theta : (\mu_{xy}, \Sigma_{xy}, \mu_a, \sigma_a, \mu_b, \sigma_b)$, where the observation likelihood of X_n for target k at frame f with parameters θ_{k_f} (i.e., Equation 2.15) is:

$$p(X_n|\theta_{k_f}) = \mathcal{N}(X_{xy}|\mu_{xy}^{k_f}, \Sigma_{xy}^{k_f}) \times \mathcal{N}(X_a|\mu_a^{k_f}, \sigma_a^{k_f}) \times \mathcal{N}(X_b|\mu_b^{k_f}, \sigma_b^{k_f}) \quad (3.9)$$

Here, μ_{xy} models the spatial center and μ_a and μ_b model the average color values for the targets. The spatial variance (i.e., diagonals of Σ_{xy}) values model the spread of the target in the respective direction and the spatial covariance (i.e., nondiagonal element of Σ_{xy}) models the spatial orientation of the target.

Equation (3.9) in Equation (2.16) together define the assignment prior and likelihood probability of an observation to an existing or a new target. For a new cluster, the integral in Equation (2.15) is calculated over the whole prior distribution. As presented in Section 2.2.1.7, the integration of the prior for Gaussian distribution in Equation (2.15), when the parameters are unknown, can be approximated with a moment matching Gaussian distribution [74]. So we replace the integration with a Gaussian that is centered on the frame and having a variance that covers the whole frame. The color components

have a similar coverage, specifically:

$$\int_{\theta} p(X_n|\theta) dG_0(\theta) \approx \mathcal{N}(X_x|\mu_x^0, \sigma_x^0) \times \mathcal{N}(X_y|\mu_y^0, \sigma_y^0) \times \mathcal{N}(X_a|\mu_a^0, \sigma_a^0) \times \mathcal{N}(X_b|\mu_b^0, \sigma_b^0),$$

$$\begin{aligned} \mu_x^0 &= \frac{w}{2}, & \mu_y^0 &= \frac{h}{2}, \\ \sigma_x^0 &= \frac{w}{2}, & \sigma_y^0 &= \frac{h}{2}, \\ \mu_a^0 &= \mu_b^0 = 0, \\ \sigma_a^0 &= \sigma_b^0 = 50, \end{aligned}$$
(3.10)

where w is the width and h is the height of each frame in the scene.

3.5 Target Assignment and Tracking with Clustering

3.5.1 Tracking Hypotheses

At each frame, we have more than one alternative tracking hypothesis and define a hypothesis (h) as a set of K_h number of targets ($\{\theta_1 \dots \theta_{K_h}\}$) and associations of observations at frame f to those targets. Specifically, each target contains historical information for more than one frame and we denote parameters of a target k at frame f with $\theta_{k_f} : (\mu_{xy}^{k_f}, \Sigma_{xy}^{k_f}, \mu_a^{k_f}, \sigma_a^{k_f}, \mu_b^{k_f}, \sigma_b^{k_f})$.

As reviewed in Section 3.2, [17] defines two inference methods (i.e., based on SMC and MCMC) for tracking with DPMM. Section 3.2 also reviewed [79], which defines a regular SMC/particle filtering framework; but at each time and for each observation explores the whole assignment space in a Rao-Blackwellized [5] fashion, where an initial estimate of the target parameters is calculated using the past dynamics, and new hypotheses are generated for each assignment, where the best few of those are kept through time.

Considering the initial update of the parameters of targets in the hypotheses, we follow a similar approach with [79]. At each frame f , we initially estimate the positions of

the targets inherited from previous frame $f - 1$, using motion dynamics of the last two frames:

$$\mu_{\hat{x}\hat{y}}^{k_f} = \mu_{xy}^{k_{f-1}} + (\mu_{xy}^{k_{f-1}} - \mu_{xy}^{k_{f-2}}), \quad (3.11)$$

where $\mu_{\hat{x}\hat{y}}^{k_f}$ denotes the position estimate of target k at frame f and $\mu_{xy}^{k_{f-1}}$ and $\mu_{xy}^{k_{f-2}}$ denote the position of the same target at frame $f - 1$ and $f - 2$ respectively.

Then, we handle the observations one by one and calculate association probabilities of observations to an existing or a new target with Equation (2.16) using Equation (3.9). Here, each new association represents a new hypothesis with an updated weight using:

$$w_{h'} = w_h \times p(c_n | X_n, c_{-n}, \Theta, \alpha), \quad (3.12)$$

where w_h is the previous weight of the hypothesis and $p(c_n | X_n, c_{-n}, \Theta, \alpha)$ is the assignment probability of observation X_n calculated with Equation (2.16). The parameters of the target that the observation is assigned to is also updated in the new hypothesis taking the new assignment into consideration.

For each frame f , the DPMM clustering inherits K_{f-1} number of clusters from the previous frame and performs clustering of the new observations to those existing (or new) clusters. Note that, some of the inherited clusters may be kept with new observation assignments, some of them may be dropped if no observations are assigned to them, and some new clusters may be generated. Altogether, they form K_f number of clusters as the tracking result for frame f which are inherited to the next frame $f + 1$ later. Addition of new clusters is controlled by the parameter α in Equation (2.16), and there is no special handling for the deletion of the clusters. The association algorithm for a single frame f is presented in Algorithm 4.

Our association scheme differs from [17] in the sense that the whole assignment space is explored once like [79] and cluster parameters are updated deterministically using the statistics of the superpixel assignments instead of random sampling of the assignments and cluster parameters. The motivation behind the deterministic update is the number of assignments being finite and keeping only top few best assignment hypotheses mostly being enough to perform drift free tracking. In the Section 3.5.3 we are going to present

our pruning approach to select the top hypotheses using a common measure in object tracking applications to calculate the strength of a set of tracking hypotheses.

3.5.2 Transition Probabilities

Although the application domain between [79] and our work is different (i.e., speech formant tracking vs. object tracking), the idea of exploring the whole assignment space is similar between the two works. A difference introduced by our method is the weight update rule of the hypotheses which considers the transition probabilities of the targets/s/clusters. While updating the weights of the hypotheses, after evaluating assignment probabilities for all observations and thus deriving new hypotheses with updated weights using Equation (3.12) in a frame, we aim to remove the hypotheses that contain targets with unusual changes in their states. We achieve this by calculating the following transition probabilities for each cluster and update the weights of the relevant hypothesis:

$$w_h = w_h \times \prod_{k \in h} p(\theta_{k_f} | \theta_{k_{f-1}}), \quad (3.13)$$

where the transitions are calculated for clusters inherited from previous frame ($f - 1$) and kept in current frame (f). The transition probability $p(\theta_{k_f} | \theta_{k_{f-1}})$ in Equation (3.13) is calculated as:

$$\begin{aligned} p(\theta_{k_f} | \theta_{k_{f-1}}) &= \mathcal{N}(\mu_{xy}^{k_f} | \mu_{\hat{x}\hat{y}}^{k_f}, \Sigma_{xy}^{k_{f-1}}) \times \\ &\quad \mathcal{N}(\sigma_x^{k_f} | \sigma_x^{k_{f-1}}, 0.1 \sigma_x^{k_{f-1}}) \times \\ &\quad \mathcal{N}(\sigma_y^{k_f} | \sigma_y^{k_{f-1}}, 0.1 \sigma_y^{k_{f-1}}), \end{aligned} \quad (3.14)$$

where $\mu_{\hat{x}\hat{y}}^{k_f}$ denotes the initial spatial estimates of the positions of the targets estimated from their previous motions with Equation (3.11).

Considering also that the variance of the spatial components of a cluster is proportional to its size, the first probability in Equation (3.14) represents the typical assumption that the position of the tracked target conforms with the past dynamics with an uncertainty proportional to its size. Similarly, the latter two probabilities represent the assumption that the spatial variance, thus size of the tracked target changes with same proportion.

3.5.3 Hypothesis Pruning

In order to keep the set of hypotheses tractable, we propose intermediate pruning steps to reduce the number of hypotheses. During evaluating assignment probabilities for each observation and generating new hypotheses, we do not process a new hypothesis any further if the updated likelihood of the new hypothesis -with Equation (3.12)- is lower than a threshold (κ_p).

After all assignment probabilities for an observation are calculated and new hypotheses are generated, we prune hypotheses by selecting the top $\min(N_{\kappa_e}, N_{max})$ best hypothesis, where N_{max} is an upper limit on the number of hypotheses and N_{κ_e} is the size of the subset, *effective number* of which is higher than a specified threshold (κ_e). Similarly after updating the weights of all hypotheses, using the transition probabilities in Equation (3.13) for all of their targets, we again prune the hypotheses by selecting top $\min(N_{\kappa_e}, N_{max})$ ones.

Effective number of a set of hypotheses, introduced by [85] and [86], is a measure of how efficient (i.e., non-*degenerated* as called in [4]) the set is, used in the framework of particle filtering [4] for object tracking applications and defined as:

$$N_{eff} = \frac{1}{\sum w_j^2}. \quad (3.15)$$

In our experiments we have observed that keeping less than 10 hypotheses, which is a highly tractable number, is enough to obtain good tracking results.

3.6 Post-Processing and Output Representation

3.6.1 Grouping Targets into Objects

Even with suitable α values of Equation (2.16), there may be cases where parts of an object may be assigned to different clusters/targets, an example of which is depicted by the body parts of persons in Figure 3.2, clustered into more than one target because of color differences in Figure 3.2(a). Thus, we run a final step to detect and group those different parts of an object into one object as in Figure 3.2(b), and present the final tracking output by representing objects as those grouped targets.

Input: $H_{f-1} = \{h_1^{f-1} \dots h_{K_{f-1}}^{f-1}\}$ hypothesis from frame $f - 1$

Input: $\{X_1 \dots X_n\}$ foreground superpixels of frame f

Result: $H_f = \{h_1^f \dots h_{K_f}^f\}$ hypothesis for frame f

```

foreach  $h$  in  $H_{f-1}$  do
  foreach  $\theta_k$  in  $h : \{\theta_1 \dots \theta_{K_h}\}$  do
     $\mu_{\hat{x}\hat{y}}^{k_f} \leftarrow \mu_{xy}^{k_{f-1}} + (\mu_{xy}^{k_{f-1}} - \mu_{xy}^{k_{f-2}})$  // Initial estimate with Equation (3.11)
  end
end

```

$H_f \leftarrow H_{f-1}$

```

foreach  $X$  in  $\{X_1 \dots X_n\}$  do
   $H'_f \leftarrow \{\}$ 
  foreach  $h$  in  $H_f$  do
    // New hypotheses derived by assigning to an existing target
    foreach  $\theta_k$  in  $h : \{\theta_1 \dots \theta_{K_h}\}$  do
       $h' \leftarrow h$  // Initialize new hypothesis
       $w_{h'} \leftarrow w_h \times N_k \times p(X|\theta_{k_f})$  // Weights for existing targets
      if  $w_{h'} > \kappa_p$  then
         $\theta'_k \leftarrow \theta_k \cup X$  // Assign  $X$  to  $\theta_k$ , update parameters
         $h' \leftarrow h' \cup \theta'_k$  // Update hypothesis with new  $\theta_k$ 
         $H'_f \leftarrow \{H'_f, h'\}$  // Add new hypothesis to the result
      end
    end
    // Additional hypothesis defining addition of a new target
     $w_{h'} \leftarrow w_h \times \alpha \times \int_{\theta} p(X|\theta) d\theta$  // Weight for new target
     $\theta'_k \leftarrow X$  // Initialize  $\theta_k$  parameters with  $X$ 
     $h' \leftarrow h' \cup \theta'_k$ 
     $H'_f \leftarrow \{H'_f, h'\}$ 
  end
   $H_f \leftarrow Prune(H'_f, \kappa_e, N_{max})$  // Section 3.5.3
end

```

```

foreach  $h$  in  $H_f$  do
  foreach  $\theta_k$  in  $h : \{\theta_1 \dots \theta_{K_h}\}$  do
     $w_h = w_h \times p(\theta_{k_f}|\theta_{k_{f-1}})$  // Using Equation (3.14)
  end
end

```

$H_f \leftarrow Prune(H_f, \kappa_e, N_{max})$ // Section 3.5.3

Algorithm 4: Superpixel to target association and generating new tracking hypotheses



FIGURE 3.2: Original targets (a) and grouped targets (b) for an example frame where a target is represented by the isocontour ellipse of its spatial components for $\sigma = 1$.

To decide whether any two targets can be merged into one, we analyze the historical motion of the targets by measuring the similarity of the motions of the pairs of targets. To measure the similarity, we use cross-correlation of historical spatial (x and y components) positions of the targets. Cross-correlation for x component of any two targets, historical positions of which are known, is calculated as:

$$Corr(x_1, x_2) = \frac{\sum_f (x_1(f) - \bar{x}_1) \times (x_2(f) - \bar{x}_2)}{\sqrt{\sum_f (x_1(f) - \bar{x}_1)^2} \times \sqrt{\sum_f (x_2(f) - \bar{x}_2)^2}}, \quad (3.16)$$

and similarly for $Corr(y_1, y_2)$ using $y_1(f)$, $y_2(f)$, \bar{y}_1 and \bar{y}_2 , where for simplicity, we represent historical spatial positions of a target (corresponding to $\mu_{xy}^{k_f}$ in Section 3.5.1) as two time series; $x_k(f)$ and $y_k(f)$ where k denotes the index of the target, f denotes frame number and \bar{x}_k and \bar{y}_k denote the means of the two series.

In case the sum of two cross-correlations (i.e., $Corr(x_1, x_2) + Corr(y_1, y_2)$) values exceeds a threshold (κ_c) value for any two targets, we assume those targets *move* together. In addition to the correlation between the movement of the targets, we also check whether the targets are spatially *close* to each other, by checking that the spatial distances between the cluster centers in x and y directions (i.e., $\mu_x^{k_f}$ and $\mu_y^{k_f}$) are less than a specific ratio (κ_v) of the variances of the spatial components of the clusters (i.e., $\sigma_x^{k_f}$ and $\sigma_y^{k_f}$) since the value of the variance of the spatial components is proportional to the size of the target.

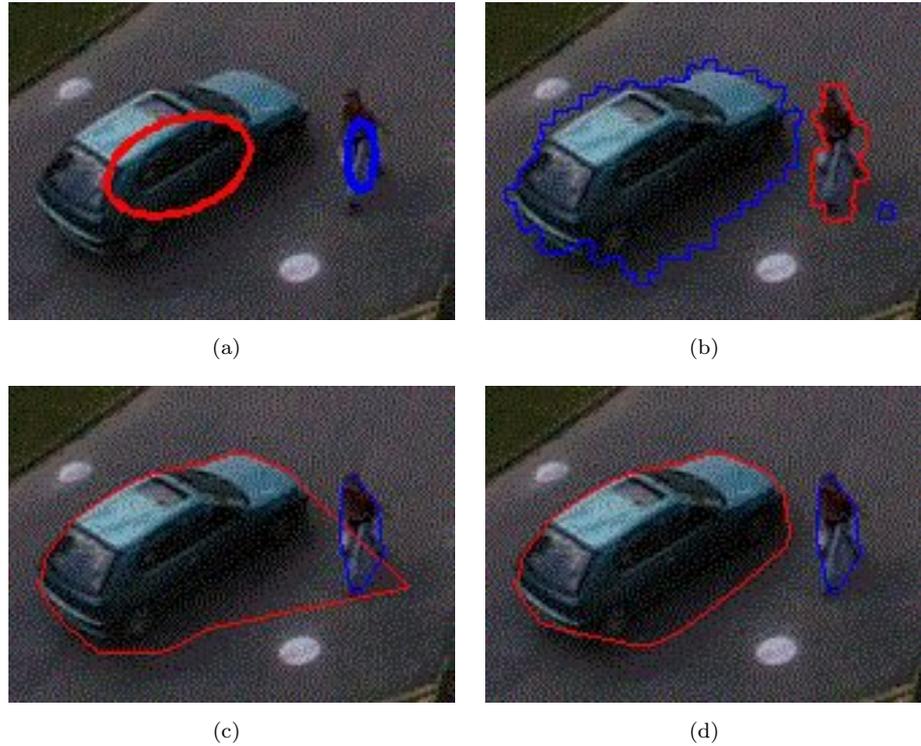


FIGURE 3.3: A target represented by the isocontour ellipse of its spatial components for $\sigma = 1$ (a), superpixel clustering results represented by superpixel borders (b), MRF labeling results represented by α -shapes boundaries of the pixels labeled as the target where distance condition in Equation (3.18) is not imposed (c) and imposed (d).

3.6.2 Refining Object Boundaries

Tracking by the DPMM clustering scheme presented in Section 3.5 generates clusters and assignment probabilities of foreground superpixels to those clusters/targets. To compensate border artifacts caused by the quick but rough superpixel extraction (as in Figure 3.3(b)), we apply a pixelwise refinement step using MRF [87], which are commonly used graphical models in image labeling tasks to obtain smooth labeling maps where the labeling task is considered as an optimization problem and the energies for labeling of the image are defined for individual pixels within local neighborhoods on a uniform grid.

Having a graphical model where nodes \mathbf{n} correspond to pixels and vertices v to set of neighboring pixels, the aim is to find the lowest overall energy E of a labeling \mathcal{L} for

image \mathcal{I} , which is calculated as sum of unary and pairwise energies as:

$$E(\mathcal{L}) = \sum_{u \in \mathbf{n}} E(\mathcal{L}_u) + \sum_{(u_1, u_2) \in \mathbf{v}} E(\mathcal{L}_{u_1}, \mathcal{L}_{u_2}), \quad (3.17)$$

where $E(\mathcal{L}_u)$ is the cost of labeling individual pixels (unary term) and $E(\mathcal{L}_{u_1}, \mathcal{L}_{u_2})$ is the cost of labeling neighboring pixels (pairwise term), which is used to enforce smoothness. In the literature of research on MRF [88], our model is considered as a *grid model* network (since we use regular pixel grid) with *pairwise* interactions (since we consider interactions only between neighboring pixels).

In our work, we aim to label each pixel in the frame as one of the targets obtained with the DPMM tracking or the background. During that process, we employ MRF as with any other labeling task [87], however derive unary term from the DPMM clustering process. Since the DPMM clustering process results in target clusters with statistics for spatial and color values; using position and color values of the pixels and Equation (3.9) can be used to calculate the likelihood of the pixels for each target. To impose this likelihood as the unary term for a single pixel X_n in Equation (3.17), we employ the negative log of the likelihood:

$$E(\mathcal{L}_{X_n}) = \begin{cases} -\log(p(X_n|\theta_k)) & \|X_x - \mu_x\| \leq \kappa_l \sigma_x \text{ and } \|X_y - \mu_y\| \leq \kappa_l \sigma_y \\ \infty & \text{otherwise.} \end{cases} \quad (3.18)$$

We calculate the unary term conditionally and impose a very high cost to the operation of labeling pixels far away from a target in order to prevent cases where such far pixels are wrongfully labeled as a target as in Figure 3.3(c) and obtain smoother labelings as in Figure 3.3(d). For the background label, we again take negative log of the background probability obtained with Equation (3.8), i.e., $-\log(p(X_i \in BG|X_i))$.

For the pairwise term in Equation (3.17), we set 8-pixel neighborhoods and give a fixed energy value (κ_m) if the neighboring pixels have different labels where same labels incur zero penalty, i.e.:

$$E(\mathcal{L}_{u_1}, \mathcal{L}_{u_2}) = \begin{cases} 0 & u_1 = u_2 \\ \kappa_m & \text{otherwise.} \end{cases} \quad (3.19)$$

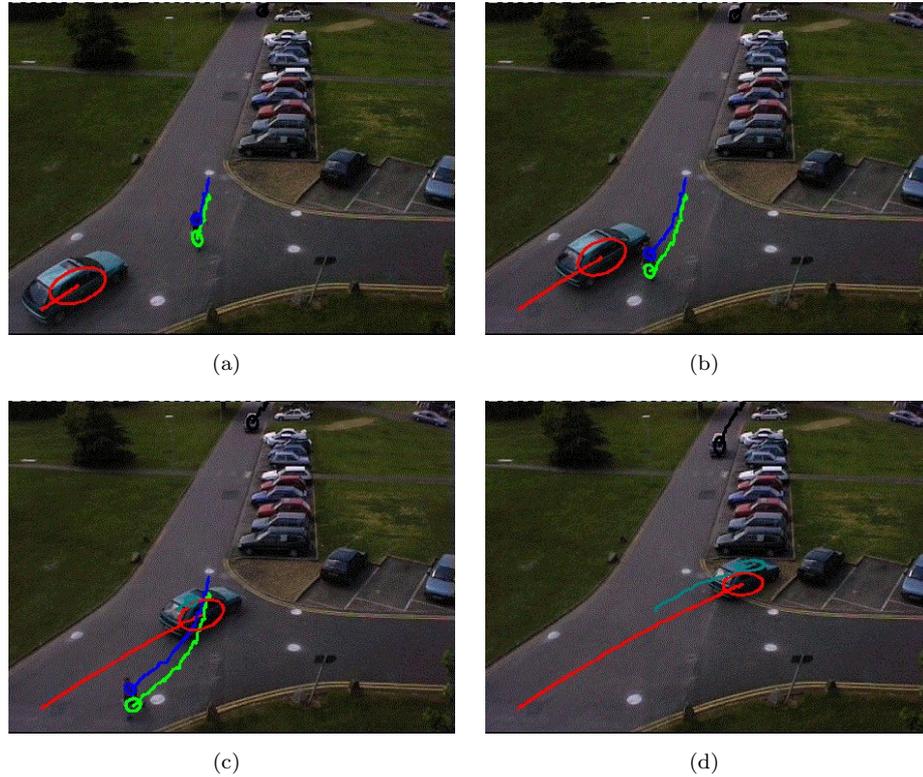


FIGURE 3.4: Example frames from PETS 2001 dataset where targets are not grouped and represented by the isocontour ellipse of their spatial components for $\sigma = 1$.

3.7 Experiments and Results

We implemented the proposed tracking with DPMM clustering algorithm using C#. For superpixel extraction, we employed the SLIC superpixels [82] implementation in VLFeat library [89]. For refining with MRF, we used FastPD MRF optimization [90][91] library. To extract the foreground pixels, we applied the GMM based background modeling implementation [83][84] of EmguCV/OpenCV [92][93]. Object borders for MRF results are represented by α -shapes [94] implementation of CGAL library [95][96].

We run the experiments on a sequence of 200 frames in PETS 2001 [97] and 100 frames in PETS 2009 [98] datasets where the α parameter of Equation (2.16) and any other parameter value is fixed across sequences. We present our visual and quantitative results with $\alpha = 1$ value.

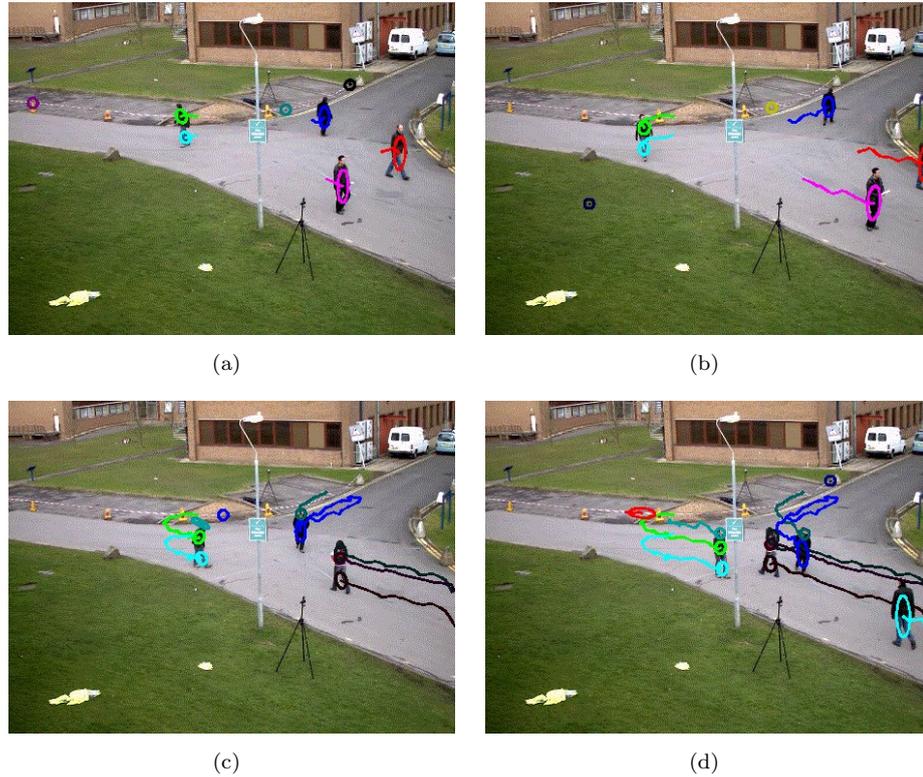


FIGURE 3.5: Example frames from PETS 2009 dataset where targets are not grouped and represented by the isocontour ellipse of their spatial components for $\sigma = 1$.

3.7.1 Visual Tracking Results

Figure 3.4 and Figure 3.5 present the tracking results where targets are represented by the isocontour ellipses of their spatial components for $\sigma = 1$, as well as their past tracks superimposed onto the image. The results demonstrate that the proposed tracking algorithm works accurately in complex situations where some tracked objects are partially occluded by others like the last two example frames from the PETS 2009 sequence. In Figure 3.5(c), it can be seen that the pedestrian that entered the scene from right passes in front of the pedestrian that was walking in the middle of the scene from the beginning. The proposed tracker continued to track those two pedestrians in the following frames -Figure 3.5(d)- without having any drift.

Figure 3.6 and Figure 3.7 show the success of the proposed target grouping scheme presented in Section 3.6.1. The two example frames correspond to same example frames presented in Figure 3.4 and Figure 3.5 and show that parts detected and tracked as

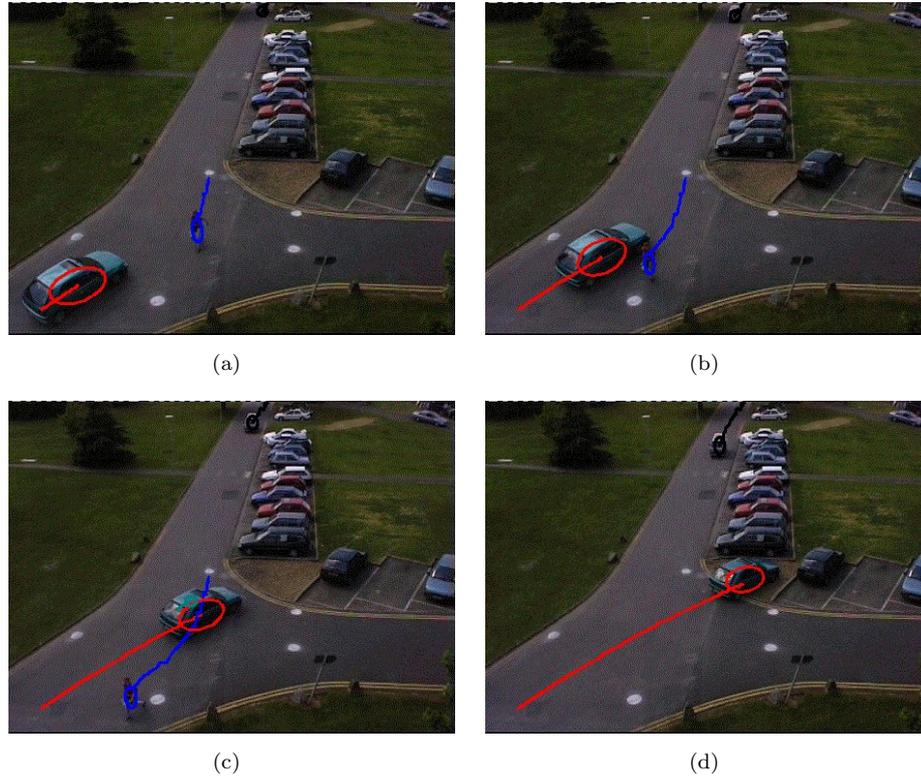


FIGURE 3.6: Example frames from PETS 2001 dataset where targets are grouped as presented in Section 3.6.1 and represented by the isocontour ellipse of their spatial components for $\sigma = 1$.

separate targets -for instance upper and lower body parts of people with different colors in Figure 3.7- are successfully grouped into a single object.

Figure 3.8 and Figure 3.9 display the tracking results where grouped targets are represented by α -shapes [94] boundaries of the pixels refined and labeled as the target with MRF as presented in Section 3.6.2. The refinement step provides pixelwise assignments for targets. Using those assignments, boundaries, which are not necessarily convex, are calculated as sets of pixels that represent the border containing all pixels assigned to the target cluster. The results show that even in complex situations that objects come together, like in Figure 3.8(b), the boundaries of them can be detected accurately by taking the grouping of the targets into consideration.

3.7.2 Quantitative Tracking Results

We report the *maximum online tracking accuracy* (MOTA) scores as defined in [99]. To calculate MOTA, for each frame, correspondence between the ground truth objects

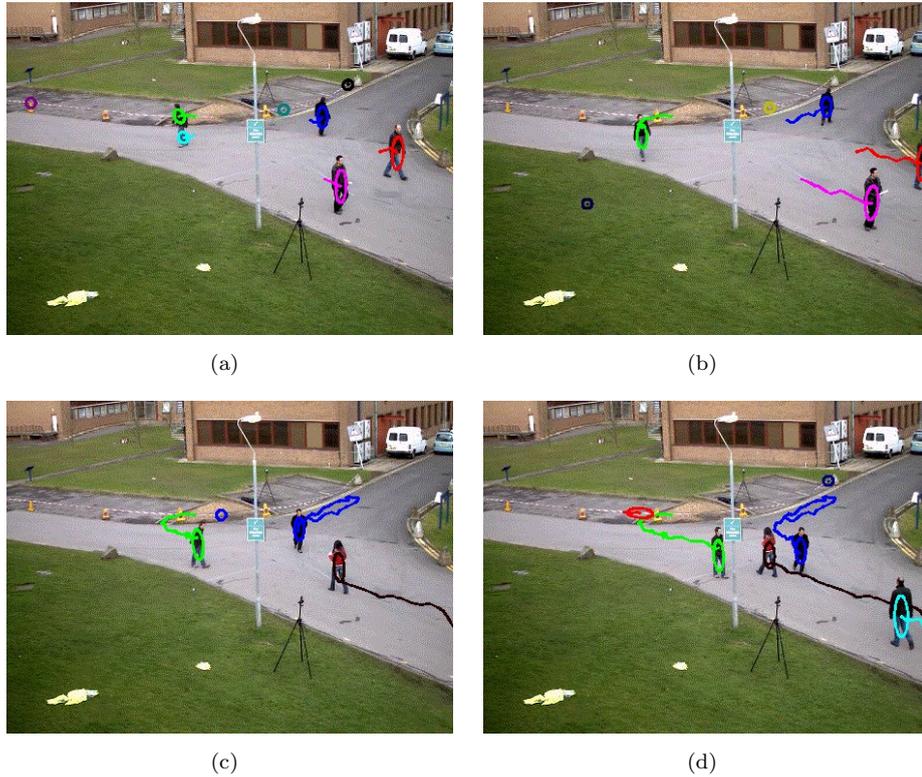


FIGURE 3.7: Example frames from PETS 2009 dataset where targets are grouped as presented in Section 3.6.1 and represented by the isocontour ellipse of their spatial components for $\sigma = 1$.

and the estimated target positions in the tracking hypothesis is calculated where an estimated target is assumed to correspond to a ground truth object if the spatial overlap between the two exceeds a threshold (t_d). In addition to frame level success rates like matches (i.e., matched target estimations and ground truth objects), misses (i.e., ground truth objects that are not matched with a target estimate) and false positives (i.e., target estimates that are not matched with a ground truth object), MOTA also tries to capture the success rate of consistent tracking of objects over time, by incorporating identity change/mismatch errors into the calculation as well. Considering those, MOTA is calculated as:

$$MOTA = 1 - \frac{\sum_f (m_f + fp_f + mm_f)}{\sum_f g_f} \quad (3.20)$$

where m_f , fp_f , mm_f and g_f are number of misses, false positives, mismatch errors and ground truth objects for frame f respectively and calculated for all frames in a sequence. We also report *seconds per frame* (SPF) which is the average time in seconds to process one frame.

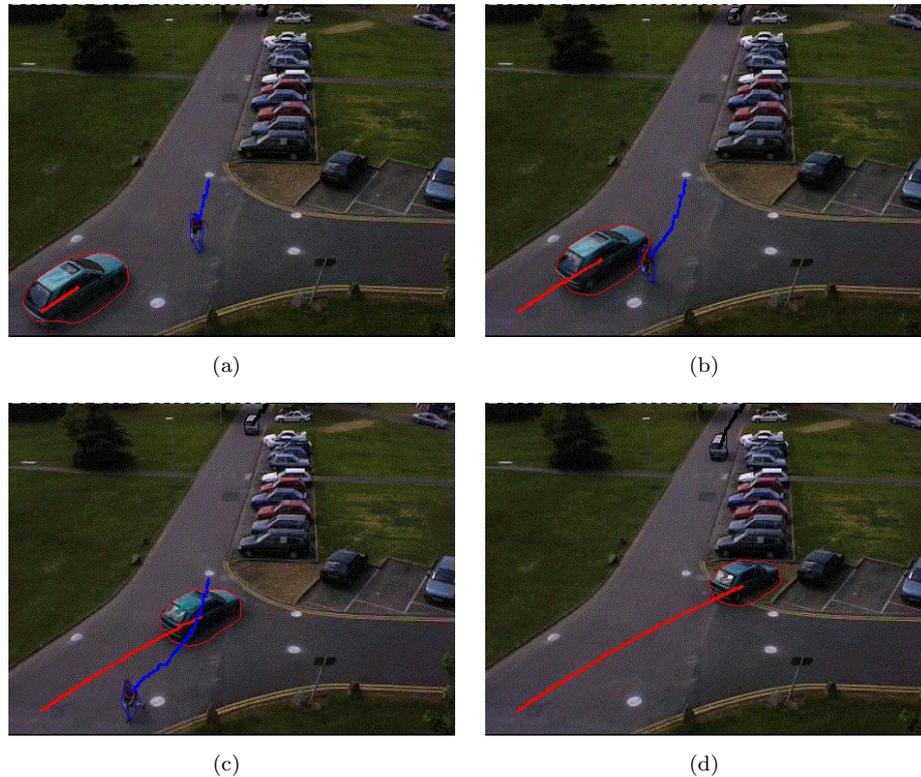


FIGURE 3.8: Example frames from PETS 2001 dataset where targets are grouped as presented in Section 3.6.1 and represented by α -shapes boundaries of the pixels refined and labeled as the target with MRF as presented in Section 3.6.2.

	MOTA	SPF
PETS 2001 Dataset1 Testing Camera2		
[17]	24.18	15.92
Proposed	96.67	4.17
PETS 2009 S2-L1 12-34 View 1		
[17]	63.66	16.34
Proposed	77.32	4.75

TABLE 3.1: Comparative results for tracking by superpixel clustering on PETS 2001 and PETS 2009 datasets.

In Table 3.1, we report the MOTA values (for $t_d = 0.3$) and average SPF values with an Intel Xeon 2.40 GHz CPU for the proposed method for the sequences from the PETS 2001 and PETS 2009 (using ground truth data from [14]) datasets, compared with the values for the method proposed in [17]. For all results, we filter clutter by removing targets that appear less than 10 frames and for [17], we repeat the particle MCMC experiments with different parameter values, and report the best result.

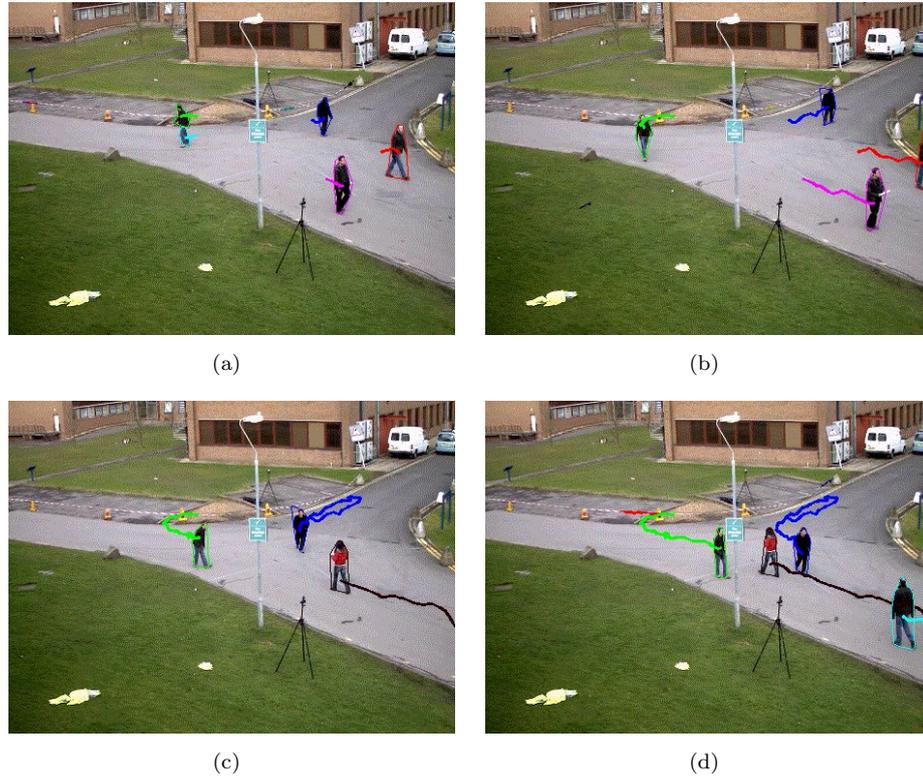


FIGURE 3.9: Example frames from PETS 2009 dataset where targets are grouped as presented in Section 3.6.1 and represented by α -shapes boundaries of the pixels refined and labeled as the target with MRF as presented in Section 3.6.2.

For our proposed method, we present the results where object boundaries are refined with MRF as described in Section 3.6.2 and an example presented in Figure 3.3(d). For both our method and [17], a target border is calculated as the smallest rectangle enclosing the target.

The results demonstrate that the proposed method significantly outperforms [17]. The primary reason is that our method calculates target parameters from superpixel assignments and represents target borders using pixel level refinements with MRF, as opposed to sampling all in [17]. In addition, we extract the foreground superpixels by a more robust background generation method instead of the simple frame differences as in [17].

3.7.3 Sensitivity Analysis Results

In Table 3.2, we present the tracking accuracy scores of our method for different superpixel sizes (i.e., s_p parameter in Equation (3.1)). Increasing the size of the superpixels, reduces the number of observations to be processed by DPMM clustering, so decreases

	MOTA	SPF
PETS 2001 Dataset1 Testing Camera2		
$s_p = 9$	94.81	8.95
$s_p = 25$	96.67	4.17
$s_p = 100$	78.15	3.58
PETS 2009 S2-L1 12-34 View 1		
$s_p = 9$	83.39	5.34
$s_p = 25$	77.32	4.75
$s_p = 100$	52.07	4.15

TABLE 3.2: Tracking results for tracking by superpixel clustering on PETS 2001 and PETS 2009 datasets with different superpixel sizes.

	MOTA	SPF
PETS 2001 Dataset1 Testing Camera2		
Noisy	96.67	5.97
Clean	96.67	4.17
PETS 2009 S2-L1 12-34 View 1		
Noisy	74.76	5.34
Clean	77.32	4.75

TABLE 3.3: Tracking results for tracking by superpixel clustering on PETS 2001 and PETS 2009 datasets with and without noise.

the running time. However, it is also clear from the results that, increasing the size of the superpixels may begin to introduce a negative impact on the tracking accuracy as well. The reason for that is, as the superpixels get larger, there is risk of grouping some pixels of the nearby targets or background into same superpixels, thus losing the distinction between targets and deforming their boundaries at the observation level that causes tracking drift problems. Smaller superpixels can obtain more refined observations that distinguish objects better, but by the cost of increasing the number of observations, thus clustering time. Taking both the running time and the tracking accuracy values in Table 3.2 for both datasets, the most preferable size is around $s_p = 25$.

In Table 3.3, we report tracking accuracies when detection noise is added by adding a random number of false observations. The number of false observations are controlled such that during foreground extraction, each background superpixel is chosen falsely as foreground with some particular probability—set as 0.001 for the presented result. The effect of noise can be seen on the running time; noisy observations introduce false targets, which involves more calculations in the process of the target assignment during

the generation of tracking hypotheses with DPMM. Table 3.3 shows that the tracking accuracy changes minimally for noisy observations, which indicates that the proposed method is robust to observation errors.

3.8 Discussions and Future Work

We use DPMM in visual object tracking within a framework that handles multiple tracking hypotheses between frames and have shown that our method performs much better than a very recent work [17] that also incorporates DPMM. Employing DPMM to cluster superpixel observations over time, allows us to detect and track unknown number of objects in a fully automatic fashion, without any initial labeling required. Since our method is based on superpixels and incorporates an efficient pruning step, the number of hypotheses does not grow in memory and is tractable. It also achieves refinement of object boundaries with MRF after employing a target grouping step to compensate errors introduced by superpixel extraction or DPMM clustering.

The main advantage of our presented approach is that, it works without any domain knowledge and does not require any object detector or any domain/application specific steps. The main drawback is that, the presented approach works sequentially between neighboring frames and does not incorporate any long term (longer than a few frames) information into the tracking process. However it is important to emphasize that, the presented approach has the potential to be employed as a baseline for a hierarchical tracking system (for instance as a tracklet extractor [22]) or can be enhanced by introducing auxiliary processes to control the object appearances or disappearances.

Chapter 4

Robust Multiple Object Tracking by Tracklet Clustering

4.1 Introduction

In this chapter, we present our work [100] on tracklet clustering and compare it with the existing tracking by detection methods. Such methods rely on a set of candidate object locations extracted by an object detector (usually one specific for the application, e.g., a HOG [42] detector for pedestrian tracking) at each frame and aim to assign or associate those detections to existing or new tracks over time. Independent from the method being used, the decision criteria to assign a detection to a track essentially determines the success of the method. An overconservative criterion often dismisses the correct assignments yielding incomplete and partitioned tracks, whereas an overrelaxed criterion causes false assignments that quickly lead into intermingles and drift from the actual trajectories.

Extracting short but highly confident sequences of tracks and then merging them hierarchically into longer and more complete tracks is a well established [22] technique to overcome this dilemma. Here, we present a nonparametric tracklet clustering approach for tracking an unknown number of objects. We take object detection results and construct short yet reliable tracklets as well as extract their color, spatial and temporal features.

Then, using ddCRP [75], we cluster those tracklets into longer tracks. By our definition, each *cluster of tracklets* corresponds to a *track*, yielding a *distinct trajectory of a single object* over time. Thus we present a hierarchical multi-object tracker system, which builds *tracklets* from object detection results at each frame and generates *clusters of tracklets* representing *tracks of objects*, in a nonparametric fashion.

4.2 Related Work and Motivation

In conventional tracking by detection applications, it is assumed that there is at most one observation per object and one observation is generated either from an object or from clutter. The solutions which are known as *multiple hypothesis trackers* [6][7] follow this assumption and update the states for each object using only the association of the detection to the object at each step.

Employing DPMM in tracking by detection applications has its roots in [101], in which authors cluster detections into an unknown number of trajectories using only information of their $2D$ positions in time. This work employs a MCMC sampler which handles assignments one by one and calculates the sampling probabilities of trajectory assignments conditioned on the assignments of all of the other observations in all times. The MCMC sampler allows to sample overall assignments of all detections to unknown number of objects.

A recent study [102] on tracklet linkage, attempts to extract tracklets from face detections and group them using pairwise tracklet similarities and hidden Markov models. Although, this method requires the number of groups (i.e., tracks) to be known a priori and actually aims primarily clustering the face detections rather than tracking them, the face clustering results can be indirectly interpreted to bear tracking results and more importantly the similarity between tracklets, inspired by [24], bears properties that we are also interested in such as taking motion dynamics into consideration.

Another example is [103], which builds complex cluster models for face detections to group them into tracks using temporally coherent CRP and DPMM. This work also prioritizes on clustering tracklets into objects and aims object discovery and models clusters based on color distribution and temporal proximity. Our work models only similarities between tracklets and does not try to use any cluster models. This fundamental

distinction of ddCRP from the CRP (and hence the DPMM) is detailed in Section 2.2.2. In addition, while calculating tracklet similarities, we use temporal information (e.g., position change in time) in addition to appearance (e.g., color) information, to cluster tracklets into tracks.

4.3 Tracklet Generation

4.3.1 Detections and Tracklet Models

The foundation of our object tracking system is the tracklet generation step which associates object detection outputs between consecutive frames. We aim to define the outputs of tracklet extraction step as generic as possible and try to obtain an overall tracking system that can be used with any kind of object detector and consequently find place in a broad range of applications.

We define each object detection d as a rectangular area centered on d_x, d_y with size d_w, d_h . Apart from location and size features, we also define a detection by its color distribution as well. We employ the same background modeling approach [83][84] reviewed in Section 3.3.2 and use only the pixels in the detection area that have a nonzero foreground probability value (i.e., Equation (3.8)). We use the *Lab* color values of the pixels and calculate two histograms d_a and d_b , corresponding to the *Lab* color channels of the non-background pixels in the detection region.

Similarly we define the parameters of a tracklet ϕ inherited from frame $f - 1$ by $\phi_x^{f-1}, \phi_y^{f-1}, \phi_w^{f-1}, \phi_h^{f-1}$ and ϕ_a^{f-1} and ϕ_b^{f-1} . By our definition, at a particular frame in time, a tracklet is defined by its final position, size and accumulated color histogram values; i.e., $\phi_x^f, \phi_y^f, \phi_w^f$ and ϕ_h^f are actually position and size of the last detection assigned to the tracklet at frame f and ϕ_a^f and ϕ_b^f are accumulated color histograms of all detections historically assigned to the tracklet through time until frame f .

4.3.2 Assigning Object Detections to Tracklets

We stick to the assumption that a single object can exist at only one location and is represented with at most (considering missed detections) one detection. Consecutively

at each frame, object detections are assigned to existing tracklets inherited from the previous frame based on similarity of color, location and size features, or a new tracklet is initiated starting from the current frame and we do not maintain multiple tracking hypotheses during tracklet extraction. In case a detection is assigned to a tracklet, the position and size of the tracklet is updated for the processed frame as the position and size of the detection and color histograms are accumulated accordingly.

For frame f , we process all object detections ($D_f = \{d_1, d_2 \dots\}$) one by one and calculate the affinities between the detections and the tracklets inherited from the previous frame $f - 1$ (i.e., $\Phi_{f-1} = \{\phi_1, \phi_2 \dots\}$). We define the affinity $A(\phi, d)$ of detection d to the tracklet ϕ similar to [102] and [24] as:

$$\begin{aligned} A(\phi, d) = & \mathcal{N}(d_x | \phi_x^{f-1}, \sigma_x) \times \mathcal{N}(d_y | \phi_y^{f-1}, \sigma_y) \times \\ & \mathcal{N}(d_u | \phi_u^{f-1}, \sigma_u) \times \mathcal{N}(d_v | \phi_v^{f-1}, \sigma_v) \times \\ & \mathcal{N}(S(d_a, \phi_a^{f-1}) | 0, \sigma_a) \times \mathcal{N}(S(d_b, \phi_b^{f-1}) | 0, \sigma_b), \end{aligned} \quad (4.1)$$

where $\mathcal{N}(x | \mu, \sigma)$ is the likelihood value of the random variable x under Gaussian distribution with mean μ and standard deviation σ . We calculate this likelihood for position and size values of the tracklet ϕ and the detection d and give higher likelihood values if values for the two are close to each other—controlled by the fixed standard deviation values $\sigma_x, \sigma_y, \sigma_w$ and σ_h as parameters.

The histogram similarities $S(d_a, \phi_a)$ and $S(d_b, \phi_b)$ in Equation (4.1) between two histograms (h_1 and h_2) are defined using the following sum of ratios over all N histogram bins [104]:

$$S(h_1, h_2) = 1 - \frac{1}{N} \sum_{i=1}^N \frac{\min(h_1^i, h_2^i)}{\max(h_1^i, h_2^i)}, \quad (4.2)$$

which takes values between 0 (best case, i.e., $h_1 = h_2$) and 1 (worst case) and we give higher likelihood values for smaller values of $S(d_a, \phi_a)$ and $S(d_b, \phi_b)$ in Equation (4.1)—again controlled by the fixed standard deviation values σ_a and σ_b as parameters.

If no detections at frame f can be assigned to an existing tracklet inherited from frame $f - 1$, it is terminated at its last location and size at frame $f - 1$ and removed from the current set Φ of tracklets that is used to assign new detections on the following frames ($f + 1 \dots$). In the end, a tracklet is defined by the sequence of position and size values

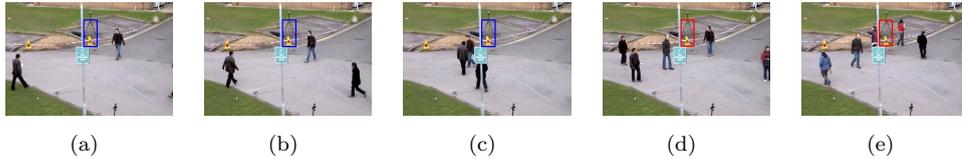


FIGURE 4.1: Three detections depicted with blue rectangles that appear only for a single frame. After three such false positives, the detections from this area are immediately filtered out; where two such example detections are depicted with red rectangles.

(i.e., sequences of $\phi_x, \phi_y, \phi_w, \phi_h$) of the constituting detections and accumulated color histogram values (i.e., accumulated ϕ_a and ϕ_b) over time.

To prevent ambiguity and drifts, we make sure that a detection is assigned to a tracklet if the affinity score between the tracklet and the detection is significantly greater than the affinities between the detection and all of the other tracklets, otherwise a new tracklet is initialized with the detection. In addition, there is no mechanism so far to prevent the case that more than one detection being assigned to a tracklet. After all detections for a frame are processed, only the detection with the best affinity score is assigned to the tracklet and using other detections, new tracklets are initialized.

4.3.3 Filtering False Object Detections and Outlier Tracklets

4.3.3.1 False Object Detections

We aim to filter out potential false object detections before beginning to assign detections to tracklets in a frame, so we eliminate any object detection results that do not have any foreground pixels. In addition, we also try to learn the regions in the scene that regularly produce false detections (even containing foreground pixels) by an online process. We keep record of detections that live only for a single frame without being assigned to a tracklet and after a region produces such single frame detections for a certain κ_t number of times, we begin to eliminate detections from that region as well, an example for which can be seen in Figure 4.1.

On the first three frames, the detection results depicted with the blue rectangles are -falsely- detected as objects by the object detector, have foreground pixels (because of the movement of the ribbon by the wind) and each last only one frame without being assigned to or generating new tracklets. After 3 such detections (i.e., $\kappa_t = 3$), the

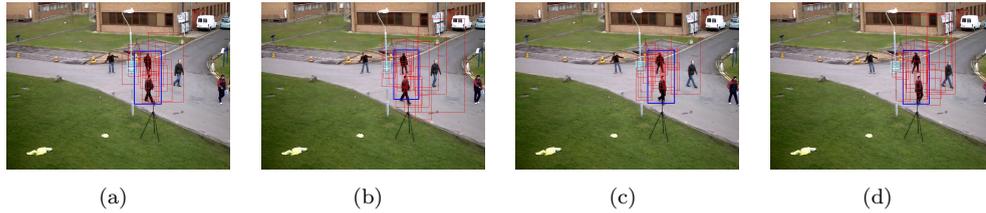


FIGURE 4.2: An example tracklet that lasts four frames. The detections that constitute the tracklet are shown in blue and each one is much larger than its neighbors which are drawn in red (every 50th detection is drawn).

detection results from this area are filtered out (like the detection results depicted with red rectangles in the last two frames) automatically before tracklet assignment step. Together with filtering the detection results without foreground pixels, these two simple eliminations help us to filter out false detections even before tracklet assignment and prevent possible local drifting.

4.3.3.2 Outlier Tracklets

Like filtering out single detections that are most likely false positives, we also aim to filter out false positives at tracklets level as well. We are interested in filtering out tracklets that have abnormal size, appear in the parts of the scene that are not likely to produce tracklets and are part of a larger tracklet.

After all frames are processed and the tracklets are generated (i.e., before clustering, which will be introduced in the next section), we calculate neighborhood statistics for each detection at every frame. For each detection d , we search every frame and collect other detections from all frames, rectangular areas of which intersect with d , to form the spatial neighborhood N_d of d . Note that neighboring detections N_d do not necessarily belong to the same frame, object or tracklet with d and try to capture statistics for that part of the scene rather than any specific object.

For detections that constitute N_d , we calculate two statistics; \widetilde{N}_d , the median size of the detections in N_d and $|N_d|$, the number of detections in N_d . Using these statistics we filter out tracklets:

- (a) All detections of which are at least κ_s times larger in height than the median of their neighborhoods. An example for this can be seen in Figure 4.2 where a short tracklet

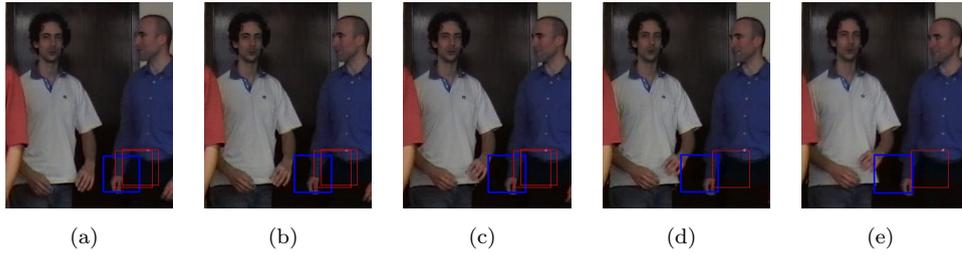


FIGURE 4.3: Five example frames belonging to a tracklet. The detections that constitute the tracklet are shown in blue and each one has much less number of neighbors (which are drawn in red) than the total number of frames so the tracklet is filtered out.

and the object detection results that constitute it are shown with blue rectangles. Every 50th spatio-temporal neighboring detection of those are also shown with red rectangles. Since every (blue) detection that constitute the tracklet is much larger than the median of their neighboring detections, that tracklet is filtered out as an outlier. Using such size statistics of the local neighbors for each detection allows us to apply a perspective invariant size filtering to the detections, as opposed to a global size filter for all regions of the frame.

- (b) Average number of neighbors of the detections of which are smaller than a ratio κ_d of the total number of frames in the sequence. An example for this can be seen in Figure 4.3 where example frames for a short tracklet and the object detection results that constitute it are shown with blue rectangles with neighboring detection of those also shown with red rectangles. Since every (blue) detection that constitute the tracklet has much less number of neighboring detections, that tracklet is filtered out as an outlier. Using the number of neighbors of the detections allows us to filter out detections which are probable to be outliers, as they appear in the parts of the scene with a low overall tracklet generation likelihood.

We finally remove the tracklets, all detections of which intersect with a larger detection in area by at least half of their size on the scene. Such detections usually appear as false detections for parts of an object which is already covered by the larger detection that they are part of. An example for this can be seen in Figure 4.4 where the red rectangles are object detection results that constitute a tracklet and each of them intersect with a larger -blue- object detection result at each frame. The *smaller* tracklet actually tracks only a part of the same object that the *larger* tracklet tracks and is eventually filtered out.

The overall pseudocode for tracklet generation is presented in Algorithm 5.

4.4 Tracklet Clustering With Distance Dependent CRP

The main contribution of our work is the clustering scheme we propose, to group the tracklets extracted in Section 4.3 with ddCRP, into tracks. Within this definition and the ddCRP clustering framework (Section 2.2.2); *tracklets* extracted in Section 4.3 are *observations* and *clusters* of those are *tracks*. We denote a single tracklet/observation with ϕ_i and the set of all tracklets with Φ as in Section 4.3. The reason we prefer ddCRP over CRP and DPMM is the flexibility of integrating our custom similarity function $F(i, j)$ in Equation (2.18) directly into the clustering process.

We use a pairwise similarity function between tracklets that takes the changes of position, size and color features of tracklets over time into account. Within a conventional DPMM framework that tries to model clusters with a base mixtures model, this would not be possible easily, since it would not be easy to integrate a mixture model that covers such a diverse range of features. Besides, employing ddCRP with pairwise similarities allows us to sample pairwise assignments quickly since pairwise similarities do not change by different assignments—as opposed to updated cluster parameters after each assignment.

In summary, instead of defining a complex cluster model and assigning tracklets to clusters as well as updating cluster parameters after each assignment, we only define and calculate tracklet similarities once and obtain the clusters automatically by sampling pairwise tracklet assignments.



FIGURE 4.4: An example tracklet, the detections constitute which are shown with red rectangles. Each detection intersects with a larger detection so the *smaller* tracklet is eventually filtered out.

```

Input: For frame  $f = 1, \dots, F$ 
 $D_f$  detections,  $BG_f$  background for  $f$ 
Result:  $\Phi = \{\phi_1, \phi_2 \dots\}$  set of tracklets

/* Go over all frames and extract tracklets */

 $\Phi \leftarrow \{\}$ 
 $FD \leftarrow \{\}$  // Set of false detection regions

for  $f \in \{1 \dots F\}$  do
  for  $d \in D_f = \{d_1, d_2 \dots\}$  do
    // Background or false detections
    if  $BG_f(d) = 1$  or  $FD(d) \geq \kappa_t$  then
      |  $D_f \leftarrow D_f - \{d\}$ 
    else
      |  $\Phi \leftarrow d$  // Update  $\Phi$  using Equation (4.1)
    end
  end

  // Update false detection areas
  for  $\phi \in \Phi$  do
    if  $\phi_f = \emptyset$  and  $Length(\phi) = 1$  then
      |  $\Phi \leftarrow \Phi - \{\phi\}$ 
      |  $FD(\phi^{f-1}) = FD(\phi^{f-1}) + 1$ 
    end
  end
end

/* Post-process and filter tracklets */

// For all tracklets and for detections that constitute each tracklet
for  $\phi \in \Phi$  and  $d \in \phi$  do
   $N_d \leftarrow \{\}$ ,  $C \leftarrow \{\}$ 
  for  $\phi_n \in \Phi$  and  $d_n \in \phi_n$  do
    if  $Area(d \cap d_n) > 0$  then
      |  $N_d \leftarrow N_d \cup \{d_n\}$ 
      | if  $SameFrame(d, d_n)$  and  $Area(d \cap d_n) \geq 0.5 \cdot Area(d)$  then
        |  $C \leftarrow C \cup d_n$ 
      | end
    end
  end

  // Check neighborhood statistics of all detections of the tracklet
  if  $d_v / \tilde{N}_d > \kappa_s$  or  $avg(|N_d|) / F < \kappa_d$  or  $d \in C$ 
     $\forall d \in \phi$  then
      |  $\Phi \leftarrow \Phi - \{\phi\}$ 
    end
end

```

Algorithm 5: Generating tracklets from Object detection results for the whole sequence

4.4.1 Extracting Tracklet Features for Clustering

For tracklets extracted in Section 4.3, we define features as vectors that hold historical information of position and size and overall color information. These features correspond to the size and position information for each frame and accumulated color information as defined in Section 4.3.1. Within the scope of this section and ddCRP clustering, we define each tracklet as an observation to be clustered, i.e., $\phi : \{\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{h}, \mathbf{f}, \mathbf{a}, \mathbf{b}\}$.

\mathbf{f} is the vector of frame numbers when the tracklet is visible. \mathbf{x} , \mathbf{y} , \mathbf{w} and \mathbf{h} are vectors of same length with \mathbf{f} and hold position and size information of the tracklet at each corresponding frame. \mathbf{a} and \mathbf{b} are color histograms of *Lab* color channels accumulated over all frames.

Before presenting the tracklet similarity function, we want to elaborate feature vectors for two temporally non-overlapping tracklets ϕ_1 and ϕ_2 . Let tracklet ϕ_1 be the former one and visible between frames f_1^s and f_1^e , i.e., $\mathbf{f}_1 = \{f_1^s, f_1^s + 1 \dots f_1^e - 1, f_1^e\}$ and ϕ_2 between frames f_2^s and f_2^e and $f_1^e < f_2^s$ and $d_f > 0$ where $d_f = f_2^s - f_1^e$ (i.e., the tracklets do not temporally overlap).

\mathbf{x}_1 is a vector of the same length with \mathbf{f}_1 , having values $\mathbf{x}_1 = \{x_{f_1^s}, x_{f_1^s+1} \dots x_{f_1^e-1}, x_{f_1^e}\}$ denoting x position and similarly \mathbf{y}_1 , \mathbf{w}_1 and \mathbf{h}_1 denoting y position, width and height of the tracklet ϕ_1 over time. Same set of vectors are also extracted for tracklet ϕ_2 , i.e., \mathbf{x}_2 , \mathbf{y}_2 , \mathbf{w}_2 and \mathbf{h}_2 . Finally, \mathbf{a}_1 , \mathbf{b}_1 , \mathbf{a}_2 and \mathbf{b}_2 denote the a and b color histograms of ϕ_1 and ϕ_2 , accumulated between frames f_1^s and f_1^e (for ϕ_1) and f_2^s and f_2^e (for ϕ_2).

4.4.2 Tracklet Similarity Function

We use a pairwise similarity function based on the probability of two tracklets ϕ_1 and ϕ_2 belonging to the same tracked object and obtain two likelihoods; F_{12} which seeks the probability that ϕ_2 is *similar* to ϕ_1 when extrapolated to the same time that ϕ_2 is visible (similar to [102]) and F_{21} which seeks the probability that ϕ_1 is *similar* to ϕ_2 when extrapolated to the same time that ϕ_1 is visible.

For F_{12} , we extrapolate four values; $\hat{x}_{f_2^s}$ which is the x value at f_2^s extrapolated from the vector \mathbf{x}_1 ; $\hat{y}_{f_2^s}$ from \mathbf{y}_1 , $\hat{w}_{f_2^s}$ from \mathbf{w}_1 and $\hat{h}_{f_2^s}$ from \mathbf{h}_1 . Similarly, for F_{21} we extrapolate $\hat{x}_{f_1^e}$ from \mathbf{x}_2 , $\hat{y}_{f_1^e}$ from \mathbf{y}_2 , $\hat{w}_{f_1^e}$ from \mathbf{w}_2 and $\hat{h}_{f_1^e}$ from \mathbf{h}_2 .

Then we define the likelihood F_{12} , which yields higher values where the extrapolated and observed sizes and positions for two tracklets are close (with the variances being proportional to the size of the object), histograms are similar and the tracklets are closer temporally, as:

$$\begin{aligned}
F_{12}(\phi_1, \phi_2) = & \mathcal{N}(x_{f_2^s} | \hat{x}_{f_2^s}, w_{f_1^e}) \times \mathcal{N}(y_{f_2^s} | \hat{y}_{f_2^s}, h_{f_1^e}) \times \\
& \mathcal{N}(w_{f_2^s} | \hat{w}_{f_2^s}, 0.1 w_{f_1^e}) \times \mathcal{N}(h_{f_2^s} | \hat{h}_{f_2^s}, 0.1 h_{f_1^e}) \times \\
& \mathcal{N}(s(a_1, a_2) | 0, \sigma_a) \times \mathcal{N}(s(b_1, b_2) | 0, \sigma_b) \times \\
& \mathcal{N}(d_f | 0, |f_1|),
\end{aligned} \tag{4.3}$$

where the histogram similarities in $s(a_1, a_2)$ and $s(b_1, b_2)$ is as defined in Equation (4.2). We calculate $F_{21}(\phi_1, \phi_2)$ similar to Equation (4.3) for the other set of extrapolated (i.e., $\hat{x}_{f_1^e}$, $\hat{y}_{f_1^e}$, $\hat{w}_{f_1^e}$ and $\hat{h}_{f_1^e}$) and observed values and define the final similarity value $F(\phi_1, \phi_2)$ as:

$$F(\phi_1, \phi_2) = \begin{cases} \max(F_{12}, F_{21}) & F_{12} > \epsilon \text{ and } F_{21} > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

We still stick to the assumption that at one frame one object is represented with at most one detection, so we set similarity immediately as zero for temporally overlapping tracklets. In other words, before calculating F_{12} and F_{21} we set $F(\phi_1, \phi_2)$ immediately to 0 if $d_f \leq 0$ where $d_f = f_2^s - f_1^e$.

Equation (4.3) does not impose tracklets to be sequential, thus occlusion handling is implicitly integrated into the model through the tracklet similarity function since we can assign nonsequential tracklets to each other and potentially cover the missed detections in between.

4.4.3 Cluster Likelihood

After the assignment prior $F(i, j)$ (i.e., Equation (2.18)), we define our cluster likelihood $p(\Phi | \{c_i \cup c_{-i}\})$ (i.e., Equation (2.19)), where the $\{c_i \cup c_{-i}\}$ term denotes the cluster structure after c_i occurs. We impose a hard limit on the cluster likelihood to prevent temporally overlapping tracklets to constitute a cluster even indirectly after a pairwise assignment.

The reason that we check the temporal tracklet overlaps at every assignment is that, even the tracklets that are assigned to each other with c_i do not temporally overlap (since the pairwise assignment prior $F(i, j)$ is 0 for temporally overlapping tracklets), a tracklet assigned directly or indirectly to any of those two can temporally overlap a tracklet assigned directly or indirectly to the other one.

Let $P(c)$ be the set of all directly and indirectly connected tracklet pairs implied by assignments c , then the cluster likelihood that imposes the hard limit to prevent temporal tracklet overlaps that we employ is:

$$p(\Phi|\{c_i \cup c_{-i}\}) \propto \begin{cases} 1 & \text{if } \forall (\phi_i, \phi_j) \in P(c); \mathbf{f}_i \cap \mathbf{f}_j = \emptyset \\ 0 & \text{otherwise} \end{cases} \quad (4.5)$$

where we set sampling probability of pairwise assignment of two tracklets, if there are other tracklets that temporally overlap and assigned (directly or indirectly) to those two, to 0 and prevent this pairwise assignment.

4.4.4 Sampling Tracklet Assignments

For ease of implementation and in order to speed up likelihood calculations, before applying Gibbs sampling, we -like [102]- construct a similarity matrix M where $M_{ij} = F(\phi_i, \phi_j)$ from Equation (4.4). M is symmetric since for temporally overlapping tracklets both M_{ij} and M_{ji} are 0 and for non-overlapping ones, i.e., $M_{ij} = M_{ji} \Leftrightarrow F(\phi_i, \phi_j) = F(\phi_j, \phi_i) = \max(F_{ij}, F_{ji})$. These tracklet similarities are calculated once and same values (of $M_{ij} = F(\phi_i, \phi_j)$) are used during clustering, since similarities of tracklet pairs do not change.

We start with an empty set of pairwise assignments (i.e., no clusters, every tracklet represents a distinct object trajectory). We handle tracklets one by one and iteratively perform Gibbs sampling for all and sample assignments to other tracklets using pairwise assignment probabilities in similarity matrix M and cluster likelihoods by Equation (4.5).

The overall pseudocode for tracklet clustering is presented in Algorithm 6.

Input: $\Phi = \{\phi_1, \phi_2 \dots\}$ set of tracklets
Result: $\mathbf{C} = \{c_1, c_2 \dots\}$ clusters of tracklets, i.e., tracks

```

//  $C(\phi_i)$  denotes cluster that  $\phi_i$  belongs to

/* Initialize objects */
 $C(\phi_i) = i \ \forall i$  // Every tracklet is a cluster alone
 $M \leftarrow []$  // Empty similarity matrix

/* Calculate pairwise similarities */

for  $\phi_1 \in \Phi$  do
  for  $\phi_2 \in \Phi$  do
    if  $\phi_1 \cap \phi_2 \neq \emptyset$  then
      |  $M[\phi_1, \phi_2] = M[\phi_2, \phi_1] \leftarrow 0$ 
    else
      |  $M[\phi_1, \phi_2] = M[\phi_2, \phi_1] \leftarrow F(\phi_1, \phi_2)$  // Equation (4.4)
    end
  end
end

/* Gibbs sampling */

for  $\phi_i \in \Phi$  do
  Sample  $c_i \propto p(c_i | \Phi, c_{-i}, \alpha, M)$  // Equation (4.5)

  // Sample assignment for  $\phi_i$ 

  if  $c_i = i$  then
    |  $C(\phi_i) = i$  // Assigned to itself
  else
    |  $C(\phi_i) = C(\phi_{c_i})$  // Assigned to another tracklet
  end

  // Update assignments for all

  for  $\phi_k \in \Phi$  do
    |  $C(\phi_k) = C(\phi_{c_k})$ 
  end
end

```

Algorithm 6: Algorithm for clustering tracklets with ddCRP into tracks

4.4.5 Output Representation of Tracks

At any time of Gibbs sampling, a cluster is defined by tracklets that have been assigned to each other directly or indirectly through others. At the end of Gibbs sampling iterations, we obtain tracklet assignments and clusters/tracks as a byproduct of those.

Since similarities of temporally overlapping tracklets are zero by Equation (4.4) and two temporally overlapping tracklets cannot constitute a cluster indirectly because of the hard cluster likelihood imposed by Equation (4.5), tracklets clustered into a track do not overlap temporally.

As defined in Section 4.1, each cluster of tracklets correspond to a trajectory of a distinct object, i.e., a track. The temporally non-overlapping tracklets in a cluster allow us to output each track by simply ordering the tracklets that constitute the cluster with respect to their timestamps and interpolate for the missing frames between the consecutive tracklets accordingly.

Because there may be missing detections (due to occlusions or simply false negatives of the object detector), we linearly interpolate positions and sizes for the missing frames between the last and first frame of the temporally consecutive tracklets in a cluster, and use the interpolated values as part of the trajectory for those frames in the output representation.

4.5 Experiments and Results

As mentioned in Section 4.3.1, our tracking system (both during tracklet extraction and clustering them into tracks) can work with any type of underlying object detector. We present our experiments on video sequences with different application types; particularly PETS 2009 [98] (person tracking), TownCentre [105] (person tracking), SPEVI [10] (face tracking), TUD Stadmitte [106] (person tracking) and ETH [107] (person tracking with moving cameras) datasets.

Before tracklet extraction, we run Haar-like features based face detector [43] to extract faces as object detection results for SPEVI dataset. For the other datasets we use the

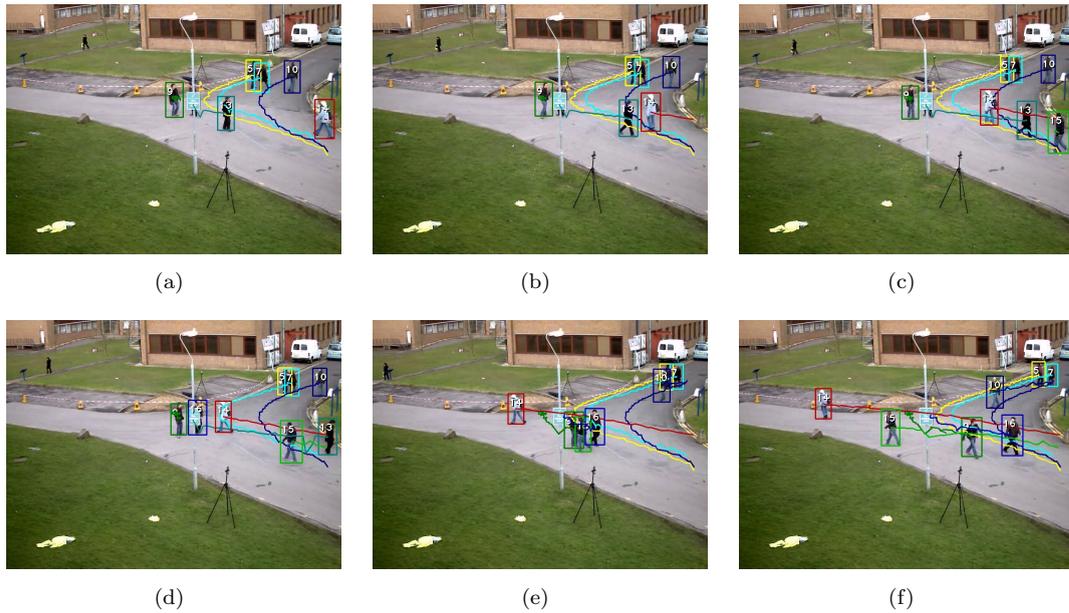


FIGURE 4.5: Example frames from PETS 2009 dataset with tracklet clustering results, i.e., each trajectory on the frame corresponds to a *cluster* of tracklets.

same object detection outputs with the previous work that we compare our results with, in order to perform an unbiased comparison.

We implemented the proposed tracking with tracklet clustering algorithm using C#. To extract the foreground pixels, we applied the the GMM based background modeling implementation [83][84] of EmguCV/OpenCV [92][93]. We also employed the Haar-like features based object detection implementation in the same library for face detection in SPEVI dataset.

We use the same parameter values for all datasets during tracklet extraction and we report our clustering results using α values that give the best results for each dataset.

4.5.1 Visual Results

In Figures 4.5, 4.6, 4.7 and 4.8, we show example visual results with and without the proposed clustering scheme separately. We indicate the tracklet only and final clustering results with distinctly colored and numerically labeled rectangles as well as their trails.

An example of a single tracked object by many clustered tracklets is the person labeled after clustering as number 14, entering the scene from right in Figure 4.5(a). In Figures

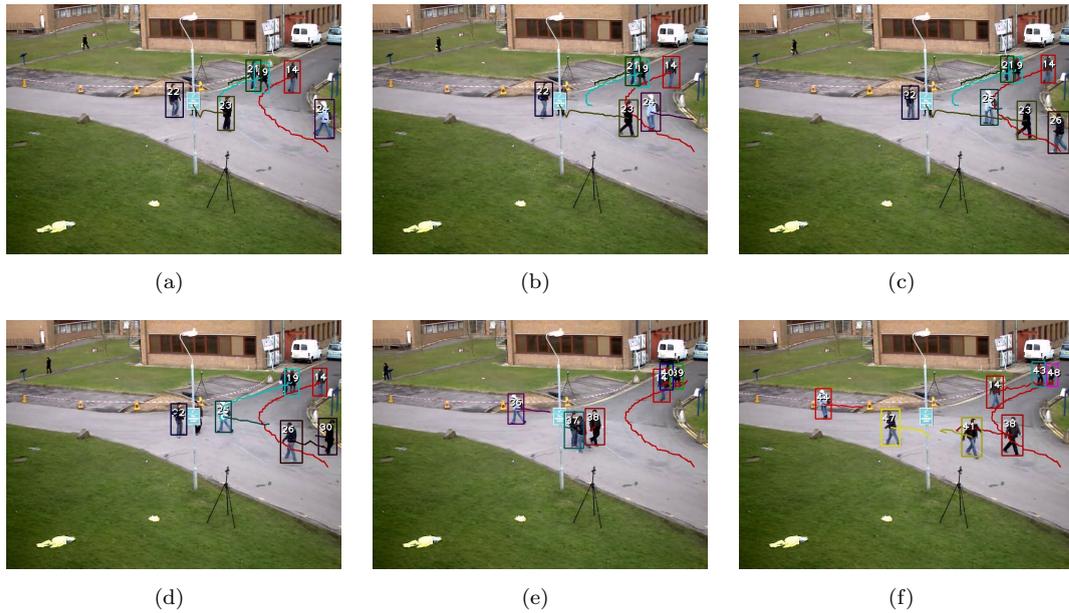


FIGURE 4.6: Example frames from PETS 2009 dataset without any clustering, i.e., each trajectory on the frame corresponds to a tracklet.

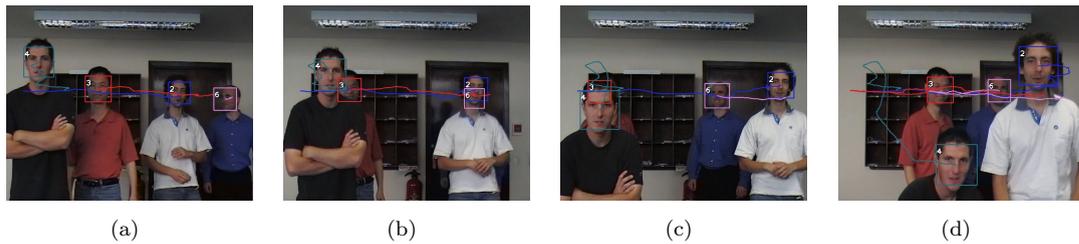


FIGURE 4.7: Example frames from SPEVI Frontal dataset with tracklet clustering results, i.e., each trajectory on the frame corresponds to a *cluster* of tracklets.

4.6(a), 4.6(c), 4.6(e) and 4.6(f) four distinct tracklets can be seen with labels 24, 25, 36 and 44 which are eventually clustered into the same track since they belong to the same label in Figure 4.5.

Likewise, the tracks for persons labeled as 5 and 7 in Figure 4.5 have been tracked with more than one tracklet as seen in Figure 4.6 before being clustered into tracks.

Examples of total occlusions can be seen in Figure 4.7(b) for track labeled as 6 and Figure 4.5(c) for the track labeled as 3, where no tracklets exist on those frames (check Figures 4.8(b) and 4.8(c)); however the tracking has not been interrupted and continue with the same label in Figure 4.7.

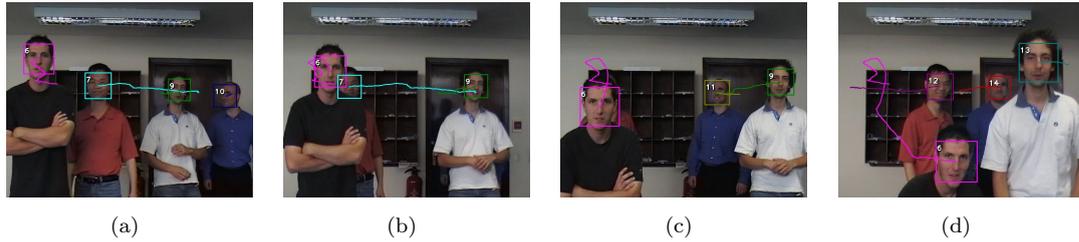


FIGURE 4.8: Example frames from SPEVI Frontal dataset without any clustering, i.e., each trajectory on the frame corresponds to a tracklet.

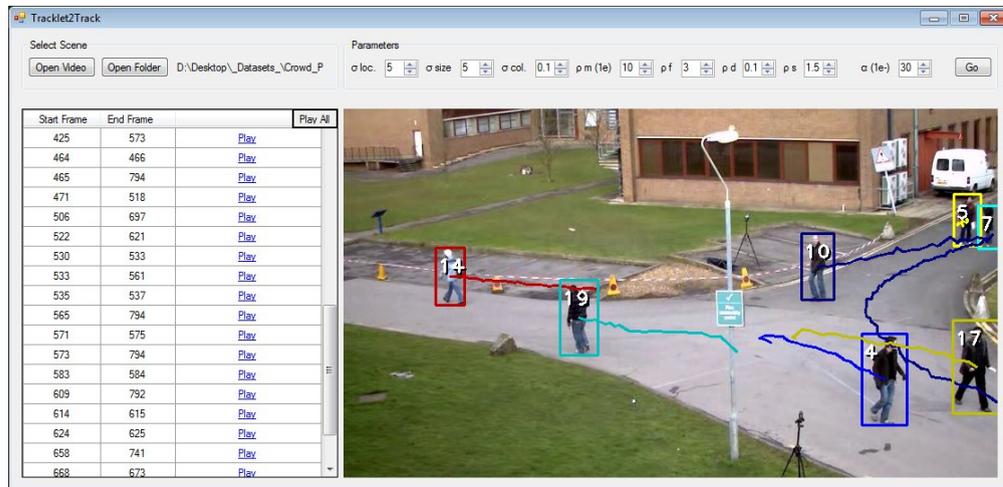


FIGURE 4.9: Graphical user interface that summarizes the tracking results per object together with entry/exit times.

4.5.2 Output Representation

In Figure 4.9 we show a screenshot of our application to run the whole system end to end and present tracking results in a convenient graphical user interface. The application summarizes the video and displays the detected tracks/objects as well as their entry and exit times to and from the scene.

4.5.3 Quantitative Results

In Table 4.1, we give our numerical results and comparisons with the results of [14] for PETS 2009, of [105] for TownCentre of [103] for SPEVI and of [15] for TUD Stadmitte and ETH datasets.

As mentioned previously, as we use the same object detection results with the relevant work, while comparing the numerical results we use the same ground truth data with the relevant work that we compare our results with. We report the number of mostly tracked (MT), partially tracked (PT), and mostly lost tracks (ML) with respect to the ground truth tracks as defined in [108], maximum online tracking accuracy (MOTA) as defined in [99] and Section 3.7.2 and precision/recall values as defined in [109]. As done for MOTA in Section 3.7.2, for each frame, correspondence between the ground truth objects and the estimated object positions in the tracking result is calculated. Besides, for MT, PT and ML the overall success for ground truth tracks is considered as considering the number of ground truth objects that are tracked for more than 80% of the time that they appear on the scene for MT and the number of ground truth objects that are not tracked for more than 20% of the time that they appear on the scene for ML. Consequently PT considers the remaining objects that are not mostly tracked or mostly lost. We use the implementation of [14] to calculate these numerical values.

As well as compared results with previous work (in rows with citations) and results obtained with our proposed tracklet clustering scheme (rows with **Proposed** header), we also present results for each dataset using only extracted tracklets in Section 4.3 (rows with **Tracklets** header) without applying the proposed clustering scheme in order to present the improvement introduced by the proposed tracklet clustering scheme and using clustering results obtained with the tracklet similarity function without the spatial components (i.e., without x , y , w and h in Equation (4.3) in rows with **No Spatial**) in order to emphasize the importance of the similarity function and the advantage of our proposed spatial similarity.

We ran our overall tracking algorithm on SPEVI frontal sequence also using face detections of [102] and compared with the ground truth of the same work and able to track all 9 tracks in their ground truth with only 1 identity switch—as opposed to their 5 *mostly tracked* tracks with 10 switches.

4.5.4 Sensitivity Analysis

We run the ddCRP clustering algorithm with different α values and report the change in MOTA results in Figure 4.10. For each dataset, as the α values increase, the results

	Prec.	Recl.	MT	PT	ML	MOTA
PETS 2009 S2-L1 12-34 View 1						
[14]	90.8	93.5	18	1	0	83.5
Tracklets	94.2	86.9	16	3	0	78.8
No Spatial	92.6	87.5	16	3	0	78.0
Proposed	92.6	92.4	18	1	0	84.5
TownCentre						
[105]	82.0	79.0	-	-	-	61.3
Tracklets	88.0	71.2	92	114	24	59.2
No Spatial	85.2	71.7	94	112	24	56.9
Proposed	84.2	78.9	127	87	16	63.5
TUD Stadtmitte						
[15]	96.7	87.0	7	3	0	-
Tracklets	96.7	78.0	6	4	0	72.2
No Spatial	90.6	84.4	7	3	0	73.6
Proposed	93.3	88.3	9	1	0	80.8
ETH (Bahnhof & Sunnyday)						
[15]	90.4	79.0	85	31	9	-
Tracklets	91.3	65.7	47	65	12	56.1
No Spatial	90.0	64.8	47	66	11	54.1
Proposed	86.9	73.6	63	51	10	61.2
SPEVI Frontal Face						
[103]	98.0	78.2	0	4	0	75.8
Tracklets	97.6	84.5	4	0	0	81.1
No Spatial	93.5	91.3	4	0	0	84.5
Proposed	96.9	91.9	4	0	0	88.7

TABLE 4.1: Comparative results for tracking by tracklet clustering on PETS 2009, TownCentre, TUD Stadmitte, ETH and SPEVI datasets.

converge to the values in Table 4.1 where only tracklets are used without clustering (rows with **Tracklets** header) values.

This makes perfect sense, since by definition in Equation (2.20), higher α values yield in more clusters (i.e., observations assigned to themselves) eventually yielding no practical clustering where every observation (i.e., tracklet) is a cluster (i.e., distinct track) by itself alone.

For lower α values, where proposed clustering scheme is practically in effect, the results do not oscillate drastically between different α values which shows that the algorithm is robust to α parameter for the datasets used.

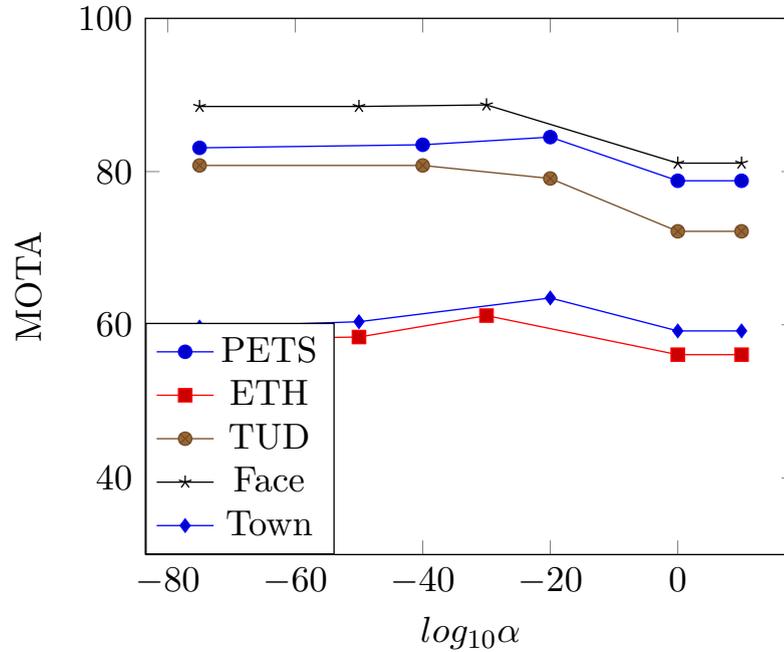


FIGURE 4.10: Sensitivity analysis results for the α parameter by analyzing the change of MOTA with different parameter values.

4.5.5 Running Speed

With an Intel Xeon 2.40 GHz CPU, for TUD and ETH sequences, excluding image I/O and object detections; our algorithm including tracklet extraction and clustering, runs at 50 FPS being 5 \times faster than [15] that runs at 10 FPS and 100 FPS for SPEVI sequence being $\sim 10\times$ faster than [103].

4.6 Improving Detection Free Superpixel Tracking

In this section, we present the results of our attempt to combine our previous work on detection free tracking by superpixel clustering, presented in Chapter 3, with the tracklet clustering approach presented in this chapter to improve the detection free tracking results.

4.6.1 Motivation

Before discussing our motivation behind this, we would like to emphasize the difference between the phrases *tracklet* and *track* once more.

In the literature of works that employ tracklets, *track* refers to the whole historical movement of an object, whereas *tracklet* refers to shorter segments of a track. Obviously, tracking is performed in a bottom-up approach; i.e., initially, tracklets are extracted as short sequences of tracks and then they are merged into complete tracks [22]. Clearly, the difference is only by interpretation; output of any tracking system can be considered as a set of incomplete segments of complete tracks and can be fed into another system to merge them into complete tracks.

Our main motivation in this section is to take the output of the superpixel tracker presented in Chapter 3 and feed them into the tracklet clustering system presented in this chapter to improve the tracking accuracy. Thus, the immediate outputs of the former, i.e., spatio-temporal clusters of superpixels, are considered as tracklets that are clustered by the latter.

4.6.2 Extracting The Tracklets

The tracker presented in Chapter 3 treats clusters of superpixels as whole tracks of an object (or part of an object) in time. Although there is a naive grouping applied to the tracks (presented in Section 3.6.1) analyzing temporal movement, it aims to group parts of an object that move together into one, rather than merging short segments of tracks from different times.

To obtain short but reliable segments of tracks (as consistent with the definition of *tracklet*), we introduce a *sanity* check into target/cluster transition which is calculated with Equation (3.14). We interpret this probability; so that if, at a frame and for a particular target, it's under a specific threshold (κ_r), we interpret it as an invalid transition and terminate the track of that particular target at that frame. Note that this is very similar to terminating a tracklet when no detection with a sufficient affinity is assigned to it in Section 4.3.2.

An example of track termination can be seen in Figure 4.11; without track termination, when targets with similar features come close, drifting may occur as happening to three people on the upper-left part of the scene in Figure 4.11(b) and Figure 4.11(c). With track termination, the tracking is terminated and new track segments are initialized in Figure 4.11(e) and Figure 4.11(f).

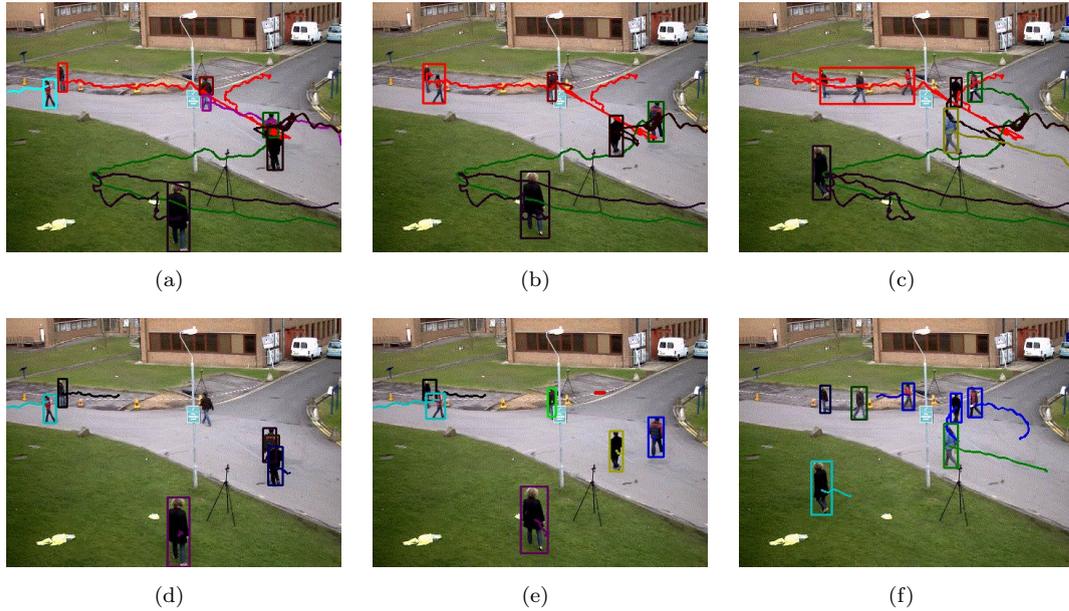


FIGURE 4.11: A tracking sequence without (a-c) and with (d-f) track termination.

4.6.3 Clustering To Obtain Complete Tracks

Similar to Section 4.4.1, we define track segments as an observation to be clustered, i.e., $\phi : \{\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{h}, \mathbf{f}, \mathbf{a}, \mathbf{b}\}$. Similarly, \mathbf{f} is the vector of frame numbers when the track segment is visible and \mathbf{x} , \mathbf{y} , \mathbf{w} and \mathbf{h} hold position and size information of the track segment at each corresponding frame; defined by the rectangle enclosing the pixels that fall into the object boundaries refined by MRF. \mathbf{a} and \mathbf{b} are accumulated color histograms of *Lab* color channels over all frames. Using the set of track segments (Φ), we sample clusters using Algorithm 6, as well as Equation 4.3 and Equation 4.5.

4.6.4 Improved Detection Free Tracking Results

We run the proposed improved detection free tracking by hierarchical superpixel clustering on PETS 2009 [98] and TUD Campus [106] datasets and present MOTA values in Table 4.2. While extracting short segments by superpixel clustering (i.e., Chapter 3), we filter clutter by removing any targets that consist of only one superpixel and apply the outlier filtering steps presented in Section 4.3.3.2. Since, by the nature of Stauffer-Grimson method, the background cannot be modeled immediately, we start the tracking from fifth frame and extrapolate the tracks in the first frames by just simply repeating

	Prec.	Recl.	MT	PT	ML	MOTA
PETS 2009 S2-L1 12-34 View 1						
[17]	70.2	51.8	5	13	1	28.2
Segments	74.7	86.9	15	4	0	55.2
Proposed	83.3	74.3	11	7	1	58.2
TUD Campus						
[19]	77.6	53.8	2	4	2	38.0
Segments	74.1	41.6	1	5	2	26.1
Proposed	79.1	61.1	3	4	1	43.2

TABLE 4.2: Comparative results for tracking by hierarchical superpixel clustering on PETS 2009 and TUD Campus datasets.

the track locations of the fifth frame. We compare with MOTA values on the same datasets with the results of two recent works on detection free tracking, specifically [17] for PETS 2009 dataset and [19] for TUD Campus dataset.

To obtain results for PETS 2009 with [17], we run the tracking experiments with the SMC implementation of the authors with different set of parameters and report the best result with $t_d = 0.2$ for MOTA on the ground truth data of [14]. For TUD Campus with [19], we use the tracking output that the authors provided with $t_d = 0.5$ for MOTA on the ground truth data that the authors provided. For our presented method, we run the experiments with different α and κ_r values and present the best result on the same ground truth data using the same (with the compared work) MOTA overlap parameter (t_d), all of which are calculated with the implementation of [14]. For all methods we filter final tracks that are shorter than 10 frames.

In addition to the results compared with previous state of the art, we also present results for each dataset using only the track segments, i.e., only the terminated tracks without any further tracklet clustering (rows with **Segments**). It is clear that track termination and second level clustering improves the tracking, thus we can achieve superior results compared to the recent state of the art detection free trackers.

4.7 Discussions and Future Work

We have presented a tracklet clustering based object tracker which is robust to occlusions, misses, and short tracking errors. We demonstrated qualitative visual results and

compared with the state of the art methods. Our main contribution is the tracklet clustering scheme, which does not depend on how the tracklets are extracted or what kind of object detectors are used to extract the tracklets. We define color, spatial, and temporal features of tracklets in our work, but the set of features can easily be extended by integrating new features into the tracklet similarity (Equation (4.3)).

Our results are superior or competitive with state of the art methods except ETH datasets; which may indicate that the proposed algorithm is rather more suitable for stationary cameras. The main advantage of our method is the simplicity of the clustering algorithm, which does not require training complex models or optimizations. This results in the speed of the proposed method being much higher than the compared work. Precision values being higher than the recall in almost all of our results indicate misses during frames, investigation of which is left as future work. We also show that the tracklet clustering is actually neutral to the tracklet extraction method and can be used to improve any system that yields track segments, as we use it to improve the detection free tracking results of the method presented in Chapter 3.

Thanks to their flexible nature, ddCRP are a promising tool for nonparametric clustering problems where the clusters are complex and cannot be easily modeled by general probabilistic models. Since the tracklet similarities are being calculated once and same similarity values are being used in Gibbs sampling iterations, the speed of the clustering process, even without any special optimization or parallelization, is quite high.

Chapter 5

People Counting by Clustering Person Detector Outputs

5.1 Introduction

In this chapter, we present our work [110] on people counting using nonparametric clustering. We present a novel clustering based framework that takes responses of a generic person detector [42] as its input.

Since even the best generic person detectors have inconsistent outputs and one person can be detected multiple times because of overlapping search windows and repetitive searches through pyramidal multiscale schemes, a post-detection bundling step is crucial to distinguish the individual people in the scene. For this we fuse different types of color, spatial and temporal features into clustering.

We use the implicit nonparametric nature of DPMM to estimate the distinct responses of people and groups of people. Within the scope of this chapter and the DPMM clustering framework, we use outputs of the person detector to be grouped into a person or a group of people as *observations* and groups of person detector outputs (i.e., observations) as *clusters*.

Ideally, only detector outputs belonging to a particular person are expected to be clustered into a single cluster, however in practice more than one person can occupy a

cluster. We aim to estimate the number of people inside a cluster using the features already extracted for clustering, using a proposed metric which is invariant to perspective distortion. In the end we obtain several clusters containing one or more people, where an estimate of number of people in each cluster is also calculated, total of which gives the estimate of people in the scene.

5.2 Related Work and Motivation

As revisited in Section 1.2, people counting with stationary single cameras can broadly be analyzed in two main groups:

1. Detection based methods infer the number of people in the scene from detector responses.
2. Regression based methods try to find a correspondence between image features and the number of people.

Our method is primarily a detection based method since we use HOG detector outputs as the basis of our clustering framework. However, the final number of people inside a cluster of HOG detections is estimated using a custom metric, which is invariant to perspective differences in the scene and benefits from the change of the number of extracted features. Our motivation in this work is to benefit from a common person detector within a framework free from any complex training process, but also use the information held in the number of the detected features without losing perspective invariance.

5.3 Extracting Person Detector Outputs For Observations

5.3.1 HOG Person Detector

We apply a HOG person detector [42] at each frame, which first calculates gradients of the image in horizontal and vertical directions and accumulates histograms of gradient directions within small cells throughout the image. For an image window, the concatenated normalized values of these histograms of cells constitute the HOG features.

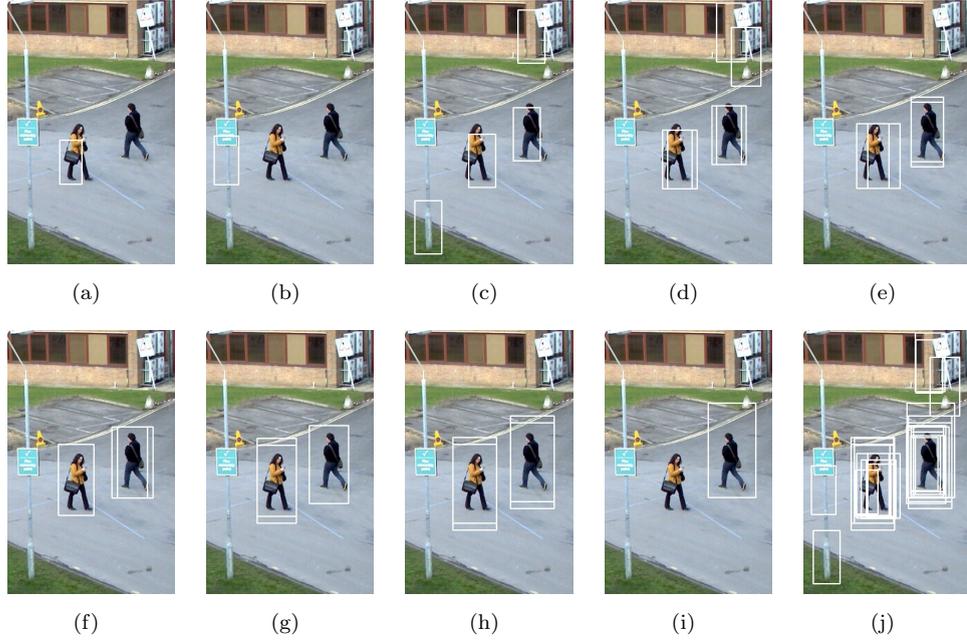


FIGURE 5.1: HOG detections with different scales separately (a)-(i) and all detections for a frame with different scales together (j).

In training, HOG features are extracted for a large number of positive (person) and several order of magnitude more negative (non-person) images and a classifier (like a linear support vector machine [111] classifier) is trained. During the detection process, a search window strides on the image and the classifier decides from the HOG features extracted for the position of the search window whether it contains a person or not. The size of the search window is repeatedly upscaled (or more commonly image is downsampled) at each iteration thus a pyramidal multiscale search is performed. In Figure 5.1, HOG detections obtained with different scale sizes, superimposed onto the frame image are presented.

5.3.2 Aggregating Detections from Consecutive Frames

We aggregate the person detections over three consecutive frames to determine the set of detection areas to be clustered for a frame. To supplement the detections, we compute two sets of optical flow [20] maps using keypoints [61], one between the previous frame and the current frame, and the other between the current frame and the next frame. Using these optical flow maps, we project the detection locations in the previous and the next frames to their estimated positions in the current frame. The shift vector for

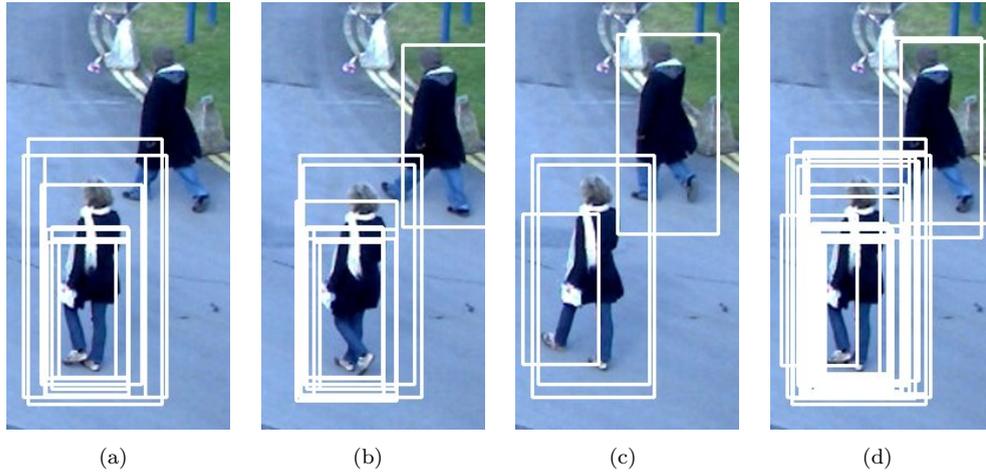


FIGURE 5.2: Detections from an example frame itself (a), from the previous frame (b), next frame (c) and detections from the frame itself and shifted detections from the previous and next frames together (d).

a detection area is taken as the average of the optical flow vectors of the keypoints that are covered by that detection area.

The motivation behind using detections from multiple frames is to improve compensation for the potential missed positives for single frames rather than handling long term occlusions. In addition to using optical flow vectors to shift detection areas on neighboring frames, we use these extracted optical flow vectors and keypoints on the following steps to employ as a subset of features (Section 5.4.2), estimate the number of people in the clusters (Section 5.6) and learning the optimal α value for Equation (2.16) (Section 5.7) as well.

An example of the proposed aggregation scheme can be seen in Figure 5.2, where detections from an immediate neighbor of a frame are aggregated and superimposed onto the example frame together with the detections of the frame itself. Thanks to the shifts of detections from the previous and the next frames (Figures 5.2(b) and 5.2(c)), the occluded pedestrian which is missed by the detector in the frame itself (Figure 5.2(a)) is covered by the detections from the previous frame (Figure 5.2(d)).

5.3.3 Filtering False Detections

In addition, we employ the same background modeling approach [83][84] reviewed in Section 3.3.2 and use only detections with a significant amount of pixels with foreground



FIGURE 5.3: Foreground map of a scene (a) and HOG based person detections (b), where detections filtered out because of size are depicted with yellow and because of foreground probability are depicted with red borders. Blue represents the final remaining detections.

values (i.e., Equation (3.8)). We filter out the detections of the person detector, in case the pixels within a detection area have average foreground probability value less than a predefined threshold (κ_b). We also remove out the foreground areas that are larger than a predefined size (κ_h).

By applying these two simple heuristics, we aim to reduce the false positive detections. An example of the response of these two filters is presented in Figure 5.3(a), where yellow bordered detections are filtered with the size threshold and the red bordered detections are filtered with the foreground threshold using the foreground probability values of the pixels as shown in Figure 5.3(b).

5.4 Observation And Cluster Models

5.4.1 Color and Spatial Features

After obtaining detection areas extracted from the frame itself and its immediate neighbor frames, we extract feature sets for each detection and model them as *observations* for the DPMM clustering stage. We model each observation using the spatial center of the detection area (i.e., x and y pixel coordinates) and the mean value of the a and b foreground pixel color components in the *Lab* color space.

5.4.2 Temporal Features

In addition to the local color and spatial information, we integrate temporal information about the movement of the people by employing an additional set of features derived from the optical flow maps between the neighboring frames. For this, we employ histogram of oriented optical flows (HOOF) [112] based features which involves calculating histograms of optical flow values at different bins of angles where each optical flow vector contributes to a histogram bin corresponding to its orientation weighted with its magnitude.

We represent histograms with four bins and compute four additional features for each detection area as defined in [112]. Each bin value is the normalized sum of the magnitudes of the optical flow vectors in relevant directions. Specifically, per [112], the direction of the optical vector (ϕ) determines the bin number (B) that the magnitude of the optical flow vector contributes to as:

- $\frac{7\pi}{4} > \phi \geq \frac{5\pi}{4} \Rightarrow B = 1$
- $2\pi > \phi \geq \frac{7\pi}{4} \Rightarrow B = 2$
- $\frac{5\pi}{4} > \phi \geq \pi \Rightarrow B = 2$
- $\frac{\pi}{4} > \phi \geq 0 \Rightarrow B = 3$
- $\pi > \phi \geq \frac{3\pi}{4} \Rightarrow B = 3$
- $\frac{3\pi}{4} > \phi \geq \frac{\pi}{4} \Rightarrow B = 1$

The motivation behind employing optical flow based features is that since we already extract optical flow vectors while aggregating detections in neighboring frames in Section 5.3.2, employing these features do not bring any significant overhead and integrate information about movement direction through time of subjects into the clustering process easily.

Figure 5.4 presents detections in an example frame where the optical flow vectors are also drawn as lines lengths of which are proportional to their magnitude and in the same direction with the optical flow vector. The normalized sum values of HOOF bins are also presented in the same figure below each corresponding detection.

5.4.3 Cluster Model

Similarly, we model each cluster using the mean and variance of the same color and spatial components as well as the values of the HOOF bins of the observations that are assigned to a cluster. So in our model, every component is a dimension in the feature space and clusters have parameters estimated with multidimensional Gaussian distributions. For computational reasons, we do not model Gaussian models with full covariance matrices for clusters but only with the covariance coefficients between the spatial components.

In summary, each observation is defined with 8 parameters; $X:(X_x, X_y, X_a, X_b, X_{h_1}, X_{h_2}, X_{h_3}, X_{h_4})$ and corresponding to spatial, color and HOOF bin components

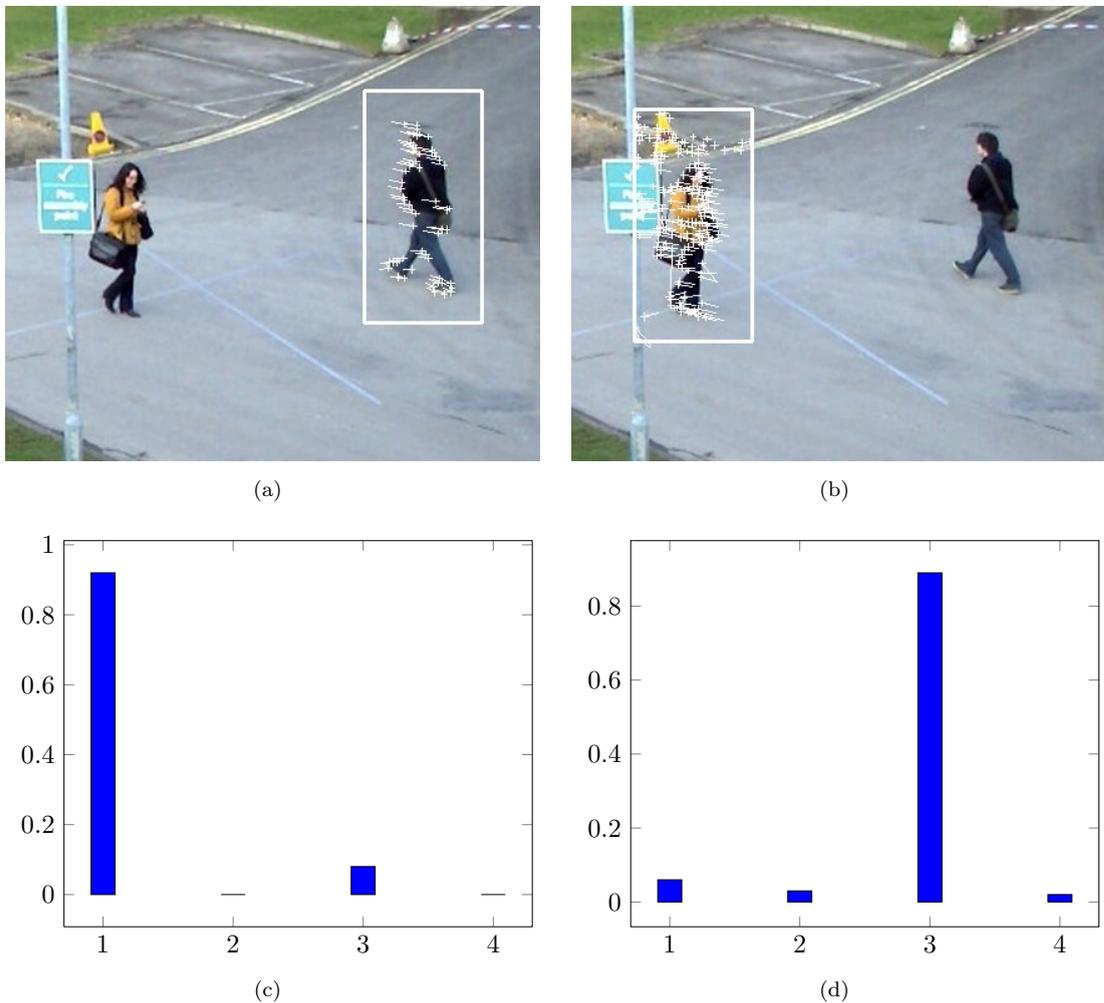


FIGURE 5.4: Two HOG detection areas with their optical flow vectors (a) and (b); their corresponding HOOF features with four bins (c) and (d).

respectively. Consecutively, clusters are modeled with 17 parameters; $\theta : (\mu_x, \mu_y, \Sigma_{xy}, \mu_a, \sigma_a, \mu_b, \sigma_b, \mu_{h_1}, \mu_{h_2}, \mu_{h_3}, \mu_{h_4}, \sigma_{h_1}, \sigma_{h_2}, \sigma_{h_3}, \sigma_{h_4})$.

5.4.4 Cluster Likelihood

Under the presented model, the likelihood that an observation X_n is generated by a cluster k with parameters θ_k is:

$$p(X_n|\theta_k) = \mathcal{N}(X_{xy}|\mu_{xy}^k, \Sigma_{xy}^k) \times \mathcal{N}(X|\mu^k, \sigma^k) \times \mathcal{N}(X|\mu^k, \sigma^k) \times \mathcal{N}(X|\mu^k, \sigma^k) \times \mathcal{N}(X|\mu^k, \sigma^k) \times \mathcal{N}(X|\mu^k, \sigma^k) \quad (5.1)$$

where the parameters of the Gaussians are estimated from the observations that are assigned to the clusters. Equation (5.1) in Equation (2.16) together define the assignment probability of an observation to an existing or a new cluster.

We again (like Section 3.4) replace the integral, for a new cluster, in Equation (2.15) with a Gaussian that is centered on the feature space and having a variance that covers the whole feature space, specifically:

$$\int_{\theta} p(X_n|\theta) d\theta \approx \mathcal{N}(X_x|\mu_x^0, \sigma_x^0) \times \mathcal{N}(X_y|\mu_y^0, \sigma_y^0) \times \mathcal{N}(X_a|\mu_a^0, \sigma_a^0) \times \mathcal{N}(X_b|\mu_b^0, \sigma_b^0), \times$$

$$\mathcal{N}(X_{h_1}|\mu_{h_1}^0, \sigma_{h_1}^0) \times \mathcal{N}(X_{h_2}|\mu_{h_2}^0, \sigma_{h_2}^0) \times \mathcal{N}(X_{h_3}|\mu_{h_3}^0, \sigma_{h_3}^0) \times \mathcal{N}(X_{h_4}|\mu_{h_4}^0, \sigma_{h_4}^0)$$

$$\mu_x^0 = \frac{w}{2}, \quad \times \quad \mu_y^0 = \frac{h}{2},$$

$$\sigma_x^0 = \frac{w}{2}, \quad \times \quad \sigma_y^0 = \frac{h}{2},$$

$$\mu_a^0 = \mu_b^0 = 0,$$

$$\sigma_a^0 = \sigma_b^0 = 50,$$

$$\mu_{h_1}^0 = \mu_{h_2}^0 = \mu_{h_3}^0 = \mu_{h_4}^0 = 0.5,$$

$$\sigma_{h_1}^0 = \sigma_{h_2}^0 = \sigma_{h_3}^0 = \sigma_{h_4}^0 = 0.5,$$
(5.2)

where w is the width and h is the height of each frame in the scene.

5.5 Sampling Target Assignments and Clustering

Using the set of extracted observations and corresponding features for each frame, we perform iterative Gibbs sampling and sample assignments for each observation (i.e., HOG detection). We evaluate the observations one by one and calculate the association probabilities of observations to an existing or to a new cluster with Equation (2.16) using Equation (5.1) and sample assignments of observations to clusters proportional to calculated association probabilities.

Although in the next step (Section 5.6) we propose a way to handle cases where more than one person are assigned to the same cluster, during clustering we prevent clusters to grow in spatial size. The most extreme of this scenario is the case where all detections in one frame being assigned to the same cluster, thus practically clustering step bringing no additional information to the count estimation process. We implement this enforcement to the clustering process implicitly by modifying the Gibbs sampling probability with another probability value with respect to the spatial sizes of the clusters.

For each frame, we calculate the following statistics for spatial sizes of observations (i.e., width and height of the rectangular areas of the HOG detections): $\mu_w, \mu_h, \sigma_w, \sigma_h$. Using these per frame statistics, we update the sampling probability $p(c_n = k; \alpha)$ of assignment of observation n to cluster k as:

$$p(c_n = k | \dots) = p(c_n = k | \dots) \times p(w_k | \mu_w, \sigma_w) \times p(h_k | \mu_h, \sigma_h), \quad (5.3)$$

where w_k and h_k are the width and height of cluster k if X_n is assigned to it, respectively. Obviously clusters in extreme sizes will have smaller sampling probabilities and thus eventually be filtered out during Gibbs sampling.

5.6 Inferring the Number of People from Clusters

The ideal outcome of the clustering process described in the previous section is that every person on the scene being represented with one distinct cluster, thus the number



FIGURE 5.5: Detections from an example frame with the keypoints inside the detection area (a)-(g) and a single cluster with all of the keypoints inside the cluster (h).

of clusters being equal to the number of people in the scene. In practice, this may not always be achieved and people which appear close to each other in the scene and having similar appearance features may be clustered into a single cluster, so taking the cluster count itself may be misleading. In addition to controlling the cluster size with the update presented in Equation (5.3), we present an additional measurement to infer the number of people in a cluster.

As mentioned in Section 5.2, using the number of feature points in the scene is a commonly employed approach such as, [40] using the number of corners detected in the overall scene to infer the number of people in the frame and [46] training a regressor on the number of clustered interest points to estimate the number of people in the cluster. Following these, we also propose a keypoint based measure since the keypoints extracted

in Section 5.3.2 for optical flow are already available without any computational overhead.

In practice, a person is usually covered by many overlapping detections and the number of keypoints within those overlapping detections do not vary much—as well as the cluster they constitute if they cover the same person. On the other hand, if the detections that constitute a cluster are related to different people, the number of total keypoints in the cluster will be much more than the number of keypoints within the separate detection areas, since the union of keypoints come from different detection sources.

On top of these assumptions we propose to use the following measure to estimate the number of people N_k in a cluster k :

$$N_k = \left\lceil \frac{p_k}{\bar{p}_{n \in k}} \right\rceil, \quad (5.4)$$

where p_k is the total number of keypoints in cluster k and $\bar{p}_{n \in k}$ is the average number of keypoints in the detection areas that constitute the cluster k and $\lceil x \rceil$ is the nearest integer to x . In Figure 5.5, an example for the proposed metric can be seen where a few detections that are clustered together are shown. HOG detections clearly belong to separate people and average number of keypoints on a single detection for all of the detections assigned to the shown cluster is 76 and the total number of keypoints inside the cluster is 151, which results in $N_k = 2$ in Equation (5.4).

The overall algorithm for clustering and estimating the number of people is shown in Algorithm 7.

5.7 Learning The α Parameter

Selecting the optimal value of α parameter in DPMM clustering (i.e., Equation (2.16)) is a problem by itself and in a clustering problem such as presented, where the number of observations (i.e., HOG detection outputs) vary widely across time, using a single α parameter value may not be suitable. We present a learning algorithm to estimate a dynamic value for α across time. We want to avoid long training times and huge training datasets and the training process to work with as little user intervention as possible.

```

Input:  $I_f, I_{f+1}, I_{f-1}$  Frame  $f$ , previous and next frames
Input:  $X^f = \{X_1^f \dots X_n^f\}$  HOG detections for frame  $f$ 
Input:  $X^{f-1}, X^{f+1}$  HOG detections for previous and next frames
Result:  $K_f = \{k_1 \dots\}$  clusters of HOG detections for frame  $f$ 
Result:  $N_f = \{N_{k_1}, N_{k_2} \dots\}$  number of people in each cluster

// Calculate frame level detection statistics
 $\{\mu_w, \mu_h, \sigma_w, \sigma_h\} \leftarrow Statistics(X)$ 

// Extract keypoints ( $P$ ) and calculate optical flows ( $O$ )
 $\{P_f, O_f\} \leftarrow OpticalFlow(I_f, I_{f+1})$ 
 $\{P_{f-1}, O_{f-1}\} \leftarrow OpticalFlow(I_{f-1}, I_f)$ 

// Aggregate detections
 $\hat{X}^{f+1} \leftarrow Shift(X^{f+1}, P_f, O_f)$ 
 $\hat{X}^{f-1} \leftarrow Shift(X^{f-1}, P_{f-1}, O_{f-1})$ 
 $X \leftarrow X^f \cup \hat{X}^{f-1} \cup \hat{X}^{f+1}$ 

// Filter detections
 $X \leftarrow \{\forall x \in X ; FG(x) > \kappa_b\} \cup \{\forall x \in X ; h_x > \kappa_h\}$ 

// Clustering with DPMM
 $K_f \leftarrow \theta_0$  // Initialize empty clusters
for # of Gibbs iterations do
  for  $n=1$  to  $N$  do
    // Remove from and update the existing cluster
    if  $\exists! \theta_t : X_n \in \theta_t$  then
       $X_n \notin \theta_t$ 
      Update  $\theta_t$ 
    end

    // Sample new assignment
    Sample  $t \propto p(c_n = t | \dots)$  // Using Equation (5.3)
    if  $t$  is new then
      Init  $\theta_t$  with  $X_n$ 
       $K_f \leftarrow \{H, \theta_t\}$ 
    else
       $X_n \in \theta_t$ 
      Update  $\theta_t$ 
    end
  end
end

// Estimate numbers in each cluster
foreach  $K$  in  $K_f$  do
   $N_f = Count(K, P_f, X)$  // Using Equation (5.4)
end

```

Algorithm 7: HOG detection to cluster association and estimating number of people

Input: $GT = \{GT_{f_1} \dots GT_{f_n}\}$ Ground truth counts of people in training frames
Input: $A = \{\alpha_1 \dots \alpha_n\}$ Set of α values to train
Input: $\forall f \in GT : I_f, I_{f+1}, I_{f-1}$ Frame f , previous and next frames
Input: $\forall f \in GT : X^f = \{X_1^f \dots X_n^f\}$ HOG detections for frame f
Input: $\forall f \in GT : X^{f-1}, X^{f+1}$ HOG detections for previous and next frames
Result: $\forall f \in GT, \forall \alpha \in A : T = \{(P, \epsilon^\alpha)\}$

```

T ← {} // Empty set of correspondences
foreach f ∈ GT do
  {P, O} ← OpticalFlow(If, If+1)
  foreach α ∈ A do
    (P, α; ε) ← Count(f, α) // Using Algorithm 7
    T ← {T, (P, α; ε)} // Add to set of correspondences
  end
end

```

Algorithm 8: Learning algorithm for optimal α value by collecting the error statistics

Under the assumption that the optimal value of α , varying through time, is related to the density of people in the scene we learn the correspondence of the optimal value to the density of the scene by running the proposed algorithm with different α values on a few representative video frames selected as the training set. The density of the scene is represented by the number of optical flow keypoints extracted in Section 5.3.2. In the end of this training, we obtain a set of (not necessarily one to one) correspondences between the pairs of the number of keypoints and the α values (P, α) and absolute error for each frame (ϵ) in the training set, i.e., $(P, \alpha; \epsilon)$.

During actual counting, we take the subset of the correspondence where the number of keypoints are close to the current frame and assign the best α value from the number of keypoints and the subset of the correspondences where the average error is smallest. To obtain the subset of *close* correspondences, all correspondences are ordered with respect to the differences between the number of keypoints in the correspondence and the current frame. Then, starting from the first one, they are added to the subset one by one, until the difference of a correspondence is double the difference of the previous one. The learning algorithm and the selection of the optimal value for the α parameter is presented in Algorithms 8 and 9.

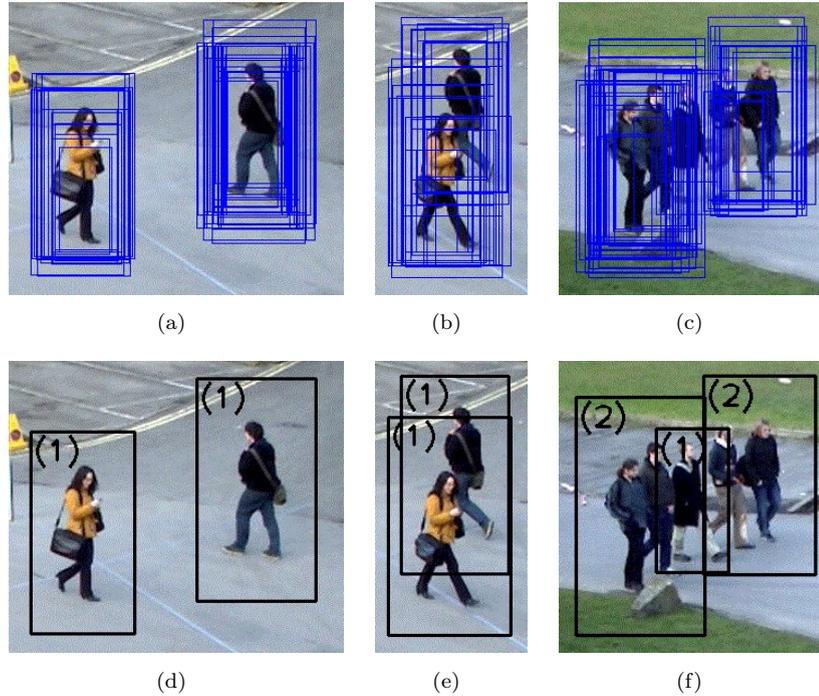


FIGURE 5.6: Example frames with HOG detections (first row) and clusters with estimated number of people (second row) for PETS 2009 dataset.

Input: I_f, I_{f+1}, I_{f-1} Frame f , previous and next frames

Input: $\forall f \in GT, \forall \alpha \in A : T = \{(P, \alpha; \epsilon)\}$ Correspondences from Algorithm 8

Result: α^* optimal α value for frame f

```

// Number of keypoints for frame  $f$ 
 $\{P_f, O_f\} \leftarrow OpticalFlow(I_f, I_{f+1})$ 

// Get close correspondences
 $T_{sort} \leftarrow Sort(T, |P - P_f|)$ 
 $T_{opt} \leftarrow T_{sort}(1)$ 
for  $n=2$  to  $Count(T_{sort})$  do
    | if  $|P(n) - P_f| > 2 \times |P(n-1) - P_f|$  then break
    |  $T_{opt} \leftarrow \{T_{opt}, T_{sort}(n)\}$ 
end

// Return  $\alpha$  with minimum average error
 $\alpha^* \leftarrow \operatorname{argmin}_{\alpha} (avg(\epsilon) : \forall \alpha \in T_{opt})$ 
    
```

Algorithm 9: Selection algorithm for the optimal α value during actual counting

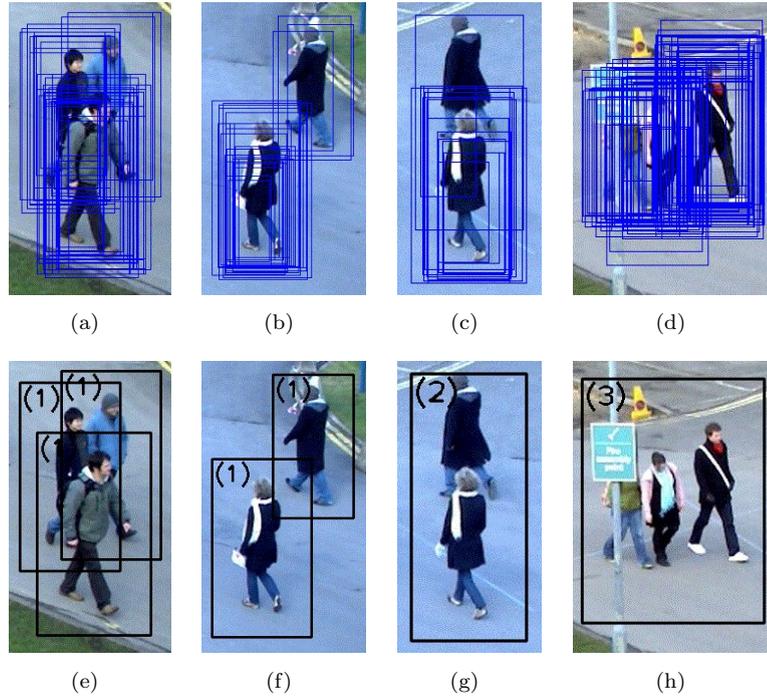


FIGURE 5.7: Example frames with HOG detections (first row) and clusters with estimated number of people (second row) for PETS 2009 dataset.

5.8 Experiments and Results

We present our experiments on PETS 2009 [98], Peds2 [113] and BEHAVE [114] datasets. To extract the foreground pixels, we applied the GMM implementation [83][84]; for person detection, the raw output of the HOG implementation and to extract optical flows, the optical flow implementation of EmguCV/OpenCV [92][93] libraries.

We did not train specific HOG models for the video sequences and used a generic HOG model [115] trained on completely separate set of videos and shipped with EmguCV; by giving manual HOG parameters, like the upscaling of the video frames or classifier thresholds, for each dataset.

While applying the proposed DPMM clustering algorithm, we took 10% of the video frames as training set and ran the proposed training scheme proposed in Section 5.7 with different α values of the DPMM clustering (Equation (2.16)). Finally we applied the clustering with the optimal α to the overall video sequence and obtained the counting results.



FIGURE 5.8: Example frames with HOG detections (first row) and clusters with estimated number of people (second row) for BEHAVE dataset.

5.8.1 Visual Results

We present some example scenes in Figures 5.6 and 5.7 for PETS 2009 dataset, Figure 5.8 for BEHAVE dataset and Figure 5.9 for Peds2 dataset. The scenes depict examples of cases where detections for each person are clustered into separate clusters successfully because of the actual spatial distance (e.g., Figure 5.6(d)) or color variations (e.g., Figure 5.6(e)) between the people being high enough.

There are also cases where a cluster contains more than one person, because of very high overlap between detections (e.g., Figure 5.7(h)) or nearby detections with similar color values (e.g., Figure 5.7(g)), and the proposed measure in Section 5.6 can successfully estimate the number of people in the cluster in such cases.

Figures 5.7(f) and 5.7(g) depict a case when two people begin to be clustered as one while coming closer, because of having similar colors and the number of detections for one (on the lower) being much higher than the other since the assignment probability for a cluster increases with the number of detections assigned to it (by the N in the numerator

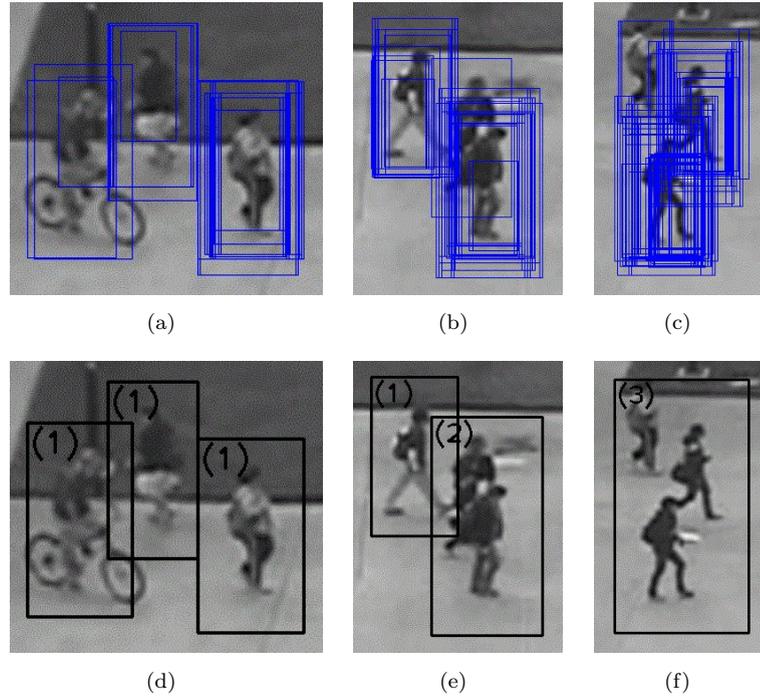


FIGURE 5.9: Example frames with HOG detections (first row) and clusters with estimated number of people (second row) for Peds2 dataset.

in Equation (2.16)). Even so, the number of people is again inferred successfully with the proposed measure in Section 5.6 in Figure 5.7(g).

5.8.2 Quantitative Results

After the counts of people for each frame are estimated, as applied in [40] and [46] we also apply a final low pass filter to the number of people to smooth out the number of countings. Particularly, denoting the number of people for a single frame estimated at the end of section Section 5.6 with N^f , the low pass filtered number of people (\hat{N}^f) for the frame f with a five frame window is:

$$\hat{N}^f = \frac{1}{5} \sum_{i=f-2}^{f+2} N^i. \quad (5.5)$$

In Table 5.1, we compare the error values of the people count estimations of the proposed method with the results of [44], [46] and [49] as well as results obtained by applying mean shift clustering [116] to the extracted detections and features for the two video sequences of the PETS 2009 dataset. In Table 5.2, we compare the error values of the proposed

	PETS 2009 S1-L1 13-57 View 1	PETS 2009 S1-L1 13-59 View 1
[44]	5.95 (30.00%)	2.08 (11.00%)
[46]	1.92 (8.70%)	2.24 (17.30%)
[49]	1.36 (6.80%)	2.55 (16.30%)
Mean Shift	3.05 (13.57%)	4.20 (29.92%)
Proposed	1.47 (7.35%)	1.50 (10.74%)

TABLE 5.1: Comparative people counting results with MAE and MRE values for PETS 2009 dataset.

clustering method with the results obtained by applying the mean shift clustering for the first test sequence of Peds2 dataset.

We report two error values; *mean absolute error (MAE)* which is the average value of the absolute error per frame, i.e.:

$$MAE = \frac{1}{F} \sum_{f=1}^F |G_f - C_f|, \quad (5.6)$$

and *mean relative error (MRE)* which is the average value of the ratio of the absolute error to the ground truth value per frame, i.e.:

$$MRE = \frac{1}{F} \sum_{f=1}^F \frac{|G_f - C_f|}{G_f}, \quad (5.7)$$

where F is the total number of frames in the sequence, G_f is the ground truth count for frame f and C_f is the counting result obtained by the relevant method for frame f .

5.8.3 Running Time

On a PC with 2.50 GHz dual-core CPU, extracting detections and features for DPMM clustering took ~ 6 seconds per frame in average, where most tasks (i.e., foreground extraction, HOG detection and optical flow calculation) implicitly benefited from CPU parallelization—thanks to EmguCV’s multithreaded nature. DPMM clustering with Gibbs sampling and the rest of the steps took ~ 1 sec. per frame in average, with no special parallelization employed.

	Mean Shift Clustering	Proposed Method
UCSD Peds2	4.58 (16.83%)	1.30 (4.90%)

TABLE 5.2: Comparative people counting results with MAE and MRE values for UCSD Peds2 dataset.

5.9 Discussions and Future Work

We present a people counting system by applying DPMM clustering on person detector outputs using different features like color, spatial and temporal. The proposed algorithm benefits from the nonparametric nature of the DPMM to handle unknown number of clusters. In our work we used HOG detectors, however the proposed algorithm is neutral to the detector being used and can be applied to any person detector which generate similar outputs.

While inferring the number of people in a cluster, only the neighborhood of the cluster is taken into consideration, since the overgrowth of clusters is prevented with Equation (5.3). Thus the proposed measure in Equation (5.4) is perspective invariant and different than [40] which assumes an overall ratio for the whole scene. The advantage of our method over [46] is that we do not need to train regressors and instead employ Equation (5.4) to infer the number of people in a cluster. Compared to [49] our method still has simpler training procedure and competitive performance. Only the number of people in a few number of frames is required, which is used while learning the α value.

The success of the overall algorithm relies on the success of the baseline detector. Better detectors for heavily crowded scenes or occlusions can be considered. The proposed method is suitable for sparsely or moderately crowded scenes. In overcrowded scenes, HOG detector may fail to distinguish targets and long term target occlusions occur.

Chapter 6

Conclusion and Future Work

We have employed nonparametric clustering and attacked common problems in computer vision.

In Chapter 3 we have presented a detection free tracker that can work on videos without incorporating any object or scene information. We reduce the clustering processing time by employing foreground superpixels and even without the refinement process we obtain superior results than a very recent similar work. With the introduced refinement process, we compensate artifacts introduced by the superpixel extraction and segment objects borders successfully.

The tracklet clustering scheme with ddCRP that we have presented in Chapter 4 is computationally very fast and robust enough to compete with the state of the art algorithms for stationary cameras. We have demonstrated that the overall tracking system is neutral to the underlying object detector or even without the object detectors, it can be used to improve detection free tracking. Consecutively, we have not incorporated any scene information into the system, which is still an area for improvement; where for instance, [117] is a recent work on semantic tracking of multiple pedestrians.

Finally we have presented a hybrid people counter system in Chapter 5. Clustering with DPMM allowed us to obtain local clusters of people and we were able to estimate the number of people in the clusters with the proposed metric. This allowed us to employ person detectors without worrying much about segmenting distinct people one by one. Also we present a quick way to learn the clustering parameter by only providing people count of a few frames.

6.1 Future Work

An open area of research is to enhance the tracking systems presented in Chapter 3 and Chapter 4 so that they can learn the optimal parameters (like α) from a few training scenes as well. However we have already presented that the superpixel tracker in Chapter 3 can handle the cases where an object is tracked by multiple targets by the grouping scheme and the tracklet clustering in Chapter 4 is already very robust to the α parameter.

DPMM clustering step can be optimized in running time by parallelization as shown in [118] where the cluster assignment probabilities are calculated in parallel, which yields clustering tasks running 4 times faster with 8 processors. A similar research on parallelization of ddCRP clustering is an open issue and the rise of multi core architectures promises potential results.

Bibliography

- [1] Rudolph Emil Kalman, “A new approach to linear filtering and prediction problems,” *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.
- [2] H.W. Sorenson, *Kalman filtering: theory and application*, IEEE Press selected reprint series. IEEE Press, 1960.
- [3] S. Julier and J. Uhlmann, “A new extension of the Kalman filter to nonlinear systems,” in *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*, 1997.
- [4] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking,” *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 174–188, Feb 2002.
- [5] Arnaud Doucet, Nando de Freitas, Kevin P. Murphy, and Stuart J. Russell, “Rao-blackwellised particle filtering for dynamic bayesian networks,” in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, 2000, UAI '00, pp. 176–183, Morgan Kaufmann Publishers Inc.
- [6] Songhwei Oh, S. Russell, and S. Sastry, “Markov chain monte carlo data association for multi-target tracking,” *Automatic Control, IEEE Transactions on*, vol. 54, no. 3, pp. 481–497, March 2009.
- [7] Donald B. Reid, “An algorithm for tracking multiple targets,” *IEEE Transactions on Automatic Control*, vol. 24, pp. 843–854, 1979.
- [8] Thomas Fortmann, Yaakov Bar-Shalom, and Molly Scheffe, “Multi-target tracking using joint probabilistic data association,” 1980, pp. 807–812.

- [9] Mahler R., “A theoretical foundation for the stein-winter probability hypothesis density multitarget tracking approach,” in *Proceedings on MSS National Symposium on Sensor and Data Fusion*, June 2002.
- [10] E. Maggio, E. Piccardo, C. Regazzoni, and A. Cavallaro, “Particle phd filtering for multi-target visual tracking,” in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, April 2007, vol. 1, pp. I–1101–I–1104.
- [11] Simo Srkk, Aki Vehtari, and Jouko Lampinen, “Rao-blackwellized particle filter for multiple target tracking,” *Information Fusion Journal*, vol. 8, 2007.
- [12] Katsuhiko Ishiguro, Takeshi Yamada, and Naonori Ueda, “Simultaneous clustering and tracking unknown number of objects,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. June 2008, pp. 1–8, IEEE.
- [13] Wenhan Luo, Xiaowei Zhao, and Tae-Kyun Kim, “Multiple object tracking: A review,” *CoRR*, vol. abs/1409.7618, 2014.
- [14] A. Milan, K. Schindler, and S. Roth, “Detection- and trajectory-level exclusion in multiple object tracking,” in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013, pp. 3682–3689.
- [15] Bo Yang and R. Nevatia, “An online learned crf model for multi-target tracking,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 2034–2041.
- [16] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” in *Proceedings of the Eighteenth International Conference on Machine Learning*, San Francisco, CA, USA, 2001, ICML '01, pp. 282–289, Morgan Kaufmann Publishers Inc.
- [17] Willie Neiswanger, Frank Wood, and Eric P. Xing, “The dependent dirichlet process mixture of objects for detection-free tracking and object modeling,” in *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, 2014, pp. 660–668.

- [18] K. Fragkiadaki and J. Shi, “Detection free tracking: Exploiting motion and topology for segmenting and tracking under entanglement,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, June 2011, pp. 2073–2080.
- [19] L. Lin, Y. Lu, C. Li, H. Cheng, and W. Zuo, “Detection-free multiobject tracking by reconfigurable inference with bundle representations,” *Cybernetics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.
- [20] J. J. Gibson, *The Perception of the Visual World*, Houghton Mifflin, Boston, MA, 1950.
- [21] Anton Milan, Laura Leal-Taix, Konrad Schindler, and Ian Reid, “Joint tracking and segmentation of multiple targets,” in *CVPR*, 2015.
- [22] Chang Huang, Bo Wu, and Ramakant Nevatia, “Robust object tracking by hierarchical association of detection responses,” in *Computer Vision ECCV 2008*, 2008, vol. 5303 of *Lecture Notes in Computer Science*, pp. 788–801, Springer Berlin Heidelberg.
- [23] Niall McLaughlin, Jesus Martinez del Rincon, and Paul Miller, *Online Multiperson Tracking With Occlusion Reasoning and Unsupervised Track Motion Model*, pp. 37–42, 2013.
- [24] Zhen Qin and Christian R. Shelton, “Improving multi-target tracking via social grouping,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [25] Xu Yan, Ioannis A Kakadiaris, and Shishir K Shah, “What do i see? modeling human visual perception for multi-person tracking,” in *Computer Vision–ECCV 2014*, pp. 314–329. Springer, 2014.
- [26] J. B. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. Le Cam and J. Neyman, Eds. 1967, vol. 1, pp. 281–297, University of California Press.

- [27] Seung-Hwan Bae and Kuk-Jin Yoon, “Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning,” in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, June 2014, pp. 1218–1225.
- [28] Li Zhang, Yuan Li, and R. Nevatia, “Global data association for multi-object tracking using network flows,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, June 2008, pp. 1–8.
- [29] Afshin Dehghan, Yicong Tian, Philip H.S. Torr, and Mubarak Shah, “Target identity-aware network flow for online multiple target tracking,” *IEEE International Conference on Computer Vision and Pattern Recognition (IEEE CVPR)*, 2015.
- [30] Xinchu Shi, Haibin Ling, Junliang Xing, and Weiming Hu, “Multi-target tracking by rank-1 tensor approximation,” in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, June 2013, pp. 2387–2394.
- [31] Wenhan Luo, Tae-Kyun Kim, B. Stenger, Xiaowei Zhao, and R. Cipolla, “Bi-label propagation for generic multiple object tracking,” in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, June 2014, pp. 1290–1297.
- [32] H. Possegger, T. Mauthner, P.M. Roth, and H. Bischof, “Occlusion geodesics for online multi-object tracking,” in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, June 2014, pp. 1306–1313.
- [33] Xiaojing Chen, Zhen Qin, Le An, and B. Bhanu, “An online learned elementary grouping model for multi-target tracking,” in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, June 2014, pp. 1242–1249.
- [34] Fabio Poiesi, Riccardo Mazzon, and Andrea Cavallaro, “Multi-target tracking on confidence maps: An application to people tracking,” *Comput. Vis. Image Underst.*, vol. 117, no. 10, pp. 1257–1272, Oct. 2013.
- [35] Siyu Tang, Mykhaylo Andriluka, and Bernt Schiele, “Detection and tracking of occluded people,” *International Journal of Computer Vision*, vol. 110, no. 1, pp. 58–69, 2014.

- [36] Manoranjan Paul, ShahME Haque, and Subrata Chakraborty, “Human detection in surveillance videos and its applications - a review,” *EURASIP Journal on Advances in Signal Processing*, vol. 2013, no. 1, 2013.
- [37] Jonathan Connell, Quanfu Fan, Prasad Gabbur, Norman Haas, Sharath Pankanti, and Hoang Trinh, “Retail video analytics: an overview and survey,” 2013.
- [38] C. Pane, M. Gasparini, A. Prati, G. Gualdi, and R. Cucchiara, “A people counting system for business analytics,” in *IEEE Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, 2013, pp. 135–140.
- [39] I.J. Amin, A.J. Taylor, F. Junejo, A. Al-Habaibeh, and R.M. Parkin, “Automated people-counting by using low-resolution infrared and visual cameras,” *Measurement*, vol. 41, no. 6, pp. 589 – 599, 2008.
- [40] Antonio Albiol, Maria Julia Silla, Alberto Albiol, and José Manuel Mossi, “Video analysis using corner motion statistics,” *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, pp. 31–38, 2009.
- [41] Wei chuan Lin, Winston K. G. Seah, and Wei Li, “Exploiting radio irregularity in wireless networks for automated people counting,” .
- [42] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, June 2005, vol. 1, pp. 886–893 vol. 1.
- [43] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2001.*, 2001, vol. 1, pp. I–511–I–518 vol.1.
- [44] V.B. Subburaman, A. Descamps, and C. Carincotte, “Counting people in the crowd using a generic head detector,” in *IEEE Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, 2012, pp. 470–475.
- [45] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151.
- [46] Donatello Conte, Pasquale Foggia, Gennaro Percannella, Francesco Tufano, and Mario Vento, “A method for counting moving people in video surveillance videos,” *EURASIP Journal on Advances in Signal Processing*, , no. 1, 2010.

- [47] Victor Lempitsky and Andrew Zisserman, “Learning to count objects in images,” in *Advances in Neural Information Processing Systems 23*, J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, Eds., pp. 1324–1332. Curran Associates, Inc., 2010.
- [48] Zhaoxiang Zhang and Min Li, “Crowd density estimation based on statistical analysis of local intra-crowd motions for public area surveillance,” *Optical Engineering*, vol. 51, no. 4, pp. 047204, 2012.
- [49] Donatello Conte, Pasquale Foggia, Gennaro Percannella, and Mario Vento, “Counting moving persons in crowded scenes,” *Machine Vision and Applications*, vol. 24, no. 5, pp. 1029–1042, 2013.
- [50] H. Fradi and J. Dugelay, “People counting system in crowded scenes based on feature regression,” in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, Aug 2012, pp. 136–140.
- [51] Carl Edward Rasmussen and Christopher K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*, The MIT Press, 2005.
- [52] A.B. Chan, Z.-S.J. Liang, and N. Vasconcelos, “Privacy preserving crowd monitoring: Counting people without people models or tracking,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, June 2008, pp. 1–7.
- [53] A.B. Chan and N. Vasconcelos, “Bayesian poisson regression for crowd counting,” in *Computer Vision, 2009 IEEE 12th International Conference on*, Sept 2009, pp. 545–551.
- [54] “Front matter,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 35, no. 3, 1973.
- [55] Santi Segui, Oriol Pujol, and Jordi Vitria, “Learning to count with deep object features,” June 2015.
- [56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information*

- Processing Systems 25*, F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, Eds., pp. 1097–1105. Curran Associates, Inc., 2012.
- [57] Homa Foroughi, Nilanjan Ray, and Hong Zhang, “Robust people counting using sparse representation and random projection,” *Pattern Recogn.*, vol. 48, no. 10, pp. 3038–3052, Oct. 2015.
- [58] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Yi Ma, “Robust face recognition via sparse representation,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 2, pp. 210–227, Feb 2009.
- [59] G. Antonini and J. P. Thiran, “Counting pedestrians in video sequences using trajectory clustering,” *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 16, no. 8, pp. 1008–1020, Aug. 2006.
- [60] V. Rabaud and S. Belongie, “Counting crowded moving objects,” in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, June 2006, vol. 1, pp. 705–711.
- [61] Bruce D. Lucas and Takeo Kanade, “An iterative image registration technique with an application to stereo vision,” in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, San Francisco, CA, USA, 1981, IJCAI’81, pp. 674–679, Morgan Kaufmann Publishers Inc.
- [62] Martin Ester, Hans peter Kriegel, Jrg Sander, and Xiaowei Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” 1996, pp. 226–231, AAAI Press.
- [63] Thomas S. Ferguson, “A bayesian analysis of some nonparametric problems,” *Ann. Statist.*, vol. 1, no. 2, pp. 209–230, 03 1973.
- [64] Charles E. Antoniak, “Mixtures of dirichlet processes with applications to bayesian nonparametric problems,” *Ann. Statist.*, vol. 2, no. 6, pp. 1152–1174, 11 1974.
- [65] Dilan Görür and Carl Edward Rasmussen, “Dirichlet process gaussian mixture models: Choice of the base distribution,” *J. Comput. Sci. Technol.*, vol. 25, no. 4, pp. 653–664, 2010.

- [66] Radford M. Neal, “Markov chain sampling methods for dirichlet process mixture models,” *Journal of Computational and Graphical Statistics*, vol. 9, no. 2, pp. 249–265, 2000.
- [67] David J. Aldous, “Exchangeability and related topics,” in *cole d’t de Probabilits de Saint-Flour XIII 1983*, P.L. Hennequin, Ed., vol. 1117 of *Lecture Notes in Mathematics*, pp. 1–198. Springer Berlin Heidelberg, 1985.
- [68] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan, “An introduction to mcmc for machine learning,” *Machine Learning*, vol. 50, no. 1-2, pp. 5–43, 2003.
- [69] Stuart Geman and Donald Geman, “Stochastic relaxation, gibbs distributions, and the bayesian restoration of images,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 6, no. 6, pp. 721–741, Nov. 1984.
- [70] Mike West, Peter Mller, and Michael D. Escobar, “Hierarchical priors and mixture models, with application in regression and density estimation,” 1993.
- [71] Xiaodong Yu, “Gibbs sampling methods for dirichlet process mixture model: Technical details,” University of Maryland, College Park, 2009.
- [72] Michael D. Escobar and Mike West, “Bayesian density estimation and inference using mixtures,” *Journal of the American Statistical Association*, vol. 90, pp. 577–588, 1994.
- [73] Steven N. MacEachern, “Computational methods for mixture of dirichlet process models,” in *Practical Nonparametric and Semiparametric Bayesian Statistics*, Dipak Dey, Peter Mller, and Debajyoti Sinha, Eds., vol. 133 of *Lecture Notes in Statistics*, pp. 23–43. Springer New York, 1998.
- [74] Erik Sudderth, *Graphical Models for Visual Object Recognition and Tracking*, Ph.D. thesis, Massachusetts Institute of Technology, May 2006.
- [75] David M. Blei and Peter I. Frazier, “Distance dependent chinese restaurant processes,” *J. Mach. Learn. Res.*, vol. 12, pp. 2461–2488, Nov. 2011.
- [76] Berthold K.P. Horn and Brian G. Schunck, “Determining optical flow,” *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

- [77] MathWorks, “Tracking cars using optical flow,” <http://uk.mathworks.com/help/vision/examples/tracking-cars-using-optical-flow.html>.
- [78] Ibrahim Saygin Topkaya, Hakan Erdogan, and Fatih Porikli, *Advances in Visual Computing: 9th International Symposium, ISVC 2013, Rethymnon, Crete, Greece, July 29-31, 2013. Proceedings, Part II*, chapter Detecting and Tracking Unknown Number of Objects with Dirichlet Process Mixture Models and Markov Random Fields, pp. 178–188, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [79] E. Ozkan, I.Y. Ozbek, and M. Demirekler, “Dynamic speech spectrum representation and tracking variable number of vocal tract resonance frequencies with time-varying dirichlet process mixture models,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 17, no. 8, pp. 1518–1532, Nov 2009.
- [80] Xiaofeng Ren and J. Malik, “Learning a classification model for segmentation,” in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Oct 2003, pp. 10–17 vol.1.
- [81] Hongyuan Zhu, Fanman Meng, Jianfei Cai, and Shijian Lu, “Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation,” *CoRR*, vol. abs/1502.00717, 2015.
- [82] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, “Slic superpixels compared to state-of-the-art superpixel methods,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 11, pp. 2274–2282, Nov 2012.
- [83] Chris Stauffer and W.E.L. Grimson, “Adaptive background mixture models for real-time tracking,” in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 1999, vol. 2, pp. –252 Vol. 2.
- [84] Zoran Zivkovic and Ferdinand van der Heijden, “Efficient adaptive density estimation per image pixel for the task of background subtraction,” *Pattern Recogn. Lett.*, vol. 27, no. 7, pp. 773–780, May 2006.
- [85] Jun S. Liu and Rong Chen, “Sequential monte carlo methods for dynamic systems,” *Journal of the American Statistical Association*, vol. 93, pp. 1032–1044, 1998.

- [86] Niclas Bergman, *Recursive Bayesian Estimation : Navigation and Tracking Applications*, Ph.D. thesis, Linkoping Studies in Science and Technology, Linkoping, Sweden, 1999.
- [87] Yuri Boykov, Olga Veksler, and Ramin Zabih, “Fast approximate energy minimization via graph cuts,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001.
- [88] Chaohui Wang, Nikos Komodakis, and Nikos Paragios, “Markov random field modeling, inference & learning in computer vision & image understanding: A survey,” *Comput. Vis. Image Underst.*, vol. 117, no. 11, pp. 1610–1627, Nov. 2013.
- [89] Andrea Vedaldi and Brian Fulkerson, “Vlfeat: An open and portable library of computer vision algorithms,” in *Proceedings of the International Conference on Multimedia*, New York, NY, USA, 2010, MM ’10, pp. 1469–1472, ACM.
- [90] Nikos Komodakis and Georgios Tziritas, “Approximate labeling via graph cuts based on linear programming,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 8, pp. 1436–1453, Aug. 2007.
- [91] Nikos Komodakis, Georgios Tziritas, and Nikos Paragios, “Performance vs computational efficiency for optimizing single and dynamic mrfs: Setting the state of the art with primal-dual strategies,” *Comput. Vis. Image Underst.*, vol. 112, no. 1, pp. 14–29, Oct. 2008.
- [92] “Emgucv software library,” <http://www.emgu.com/>, 2008.
- [93] G. Bradski, “Opencv software library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [94] Fausto Bernardini and Chandrajit L. Bajaj, “Sampling and reconstructing manifolds using alpha-shapes,” in *In Proc. 9th Canad. Conf. Comput. Geom*, 1997, p. pages.
- [95] The CGAL Project, *CGAL User and Reference Manual*, CGAL Editorial Board, 4.6.1 edition, 2015.
- [96] Tran Kai Frank Da, “2D alpha shapes,” in *CGAL User and Reference Manual*. CGAL Editorial Board, 4.6.1 edition, 2015.

-
- [97] PETS2001, “Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance,” <ftp://ftp.pets.rdg.ac.uk/pub/PETS2001/>, 2001.
- [98] PETS2009, “Eleventh IEEE International Workshop on Performance Evaluation of Tracking and Surveillance,” <ftp://ftp.pets.rdg.ac.uk/pub/PETS2009/>, 2009.
- [99] Keni Bernardin and Rainer Stiefelwagen, “Evaluating multiple object tracking performance: The clear mot metrics,” *J. Image Video Process.*, vol. 2008, pp. 1:1–1:10, Jan. 2008.
- [100] Ibrahim Saygin Topkaya, Hakan Erdogan, and Fatih Porikli, “Tracklet clustering for robust multiple object tracking using distance dependent chinese restaurant processes,” *Signal, Image and Video Processing*, pp. 1–8, 2015.
- [101] E. B. Fox, D. S. Choi, and A. S. Willsky, “Nonparametric Bayesian methods for large scale multi-target tracking,” in *Proc. Asilomar Conf. Signals, Systems, and Computers*. November 2006, IEEE.
- [102] Baoyuan Wu, Siwei Lyu, Bao-Gang Hu, and Qiang Ji, “Simultaneous clustering and tracklet linking for multi-face tracking in videos,” in *Computer Vision (ICCV), 2013 IEEE International Conference on*, Dec 2013, pp. 2856–2863.
- [103] Adway Mitra, Soma Biswas, and Chiranjib Bhattacharyya, “Temporally coherent CRP: A bayesian non-parametric approach for clustering tracklets with applications to person discovery in videos,” in *SIAM Conference on Data Mining (SDM), 2015*, 2015.
- [104] Dietrich Van der Weken, Mike Nachtgael, and Etienne E. Kerre, “Some new similarity measures for histograms,” in *ICVGIP 2004, Fourth Indian Conference on Computer Vision, Graphics & Image Processing, Kolkata, India, December 16-18, 2004*, 2004, pp. 441–446.
- [105] Ben Benfold and Ian Reid, “Stable multi-target tracking in real-time surveillance video,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, June 2011, pp. 3457–3464.
- [106] M. Andriluka, S. Roth, and B. Schiele, “Monocular 3d pose estimation and tracking by detection,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, pp. 623–630.

- [107] A. Ess, B. Leibe, K. Schindler, , and L. van Gool, “A mobile vision system for robust multi-person tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR’08)*. June 2008, IEEE Press.
- [108] Yuan Li, Chang Huang, and Ram Nevatia, “Learning to associate: Hybridboosted multi-target tracker for crowded scene,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2953–2960.
- [109] K. Smith, D. Gatica-Perez, J. Odobez, and Sileye Ba, “Evaluating multi-object tracking,” in *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, June 2005, pp. 36–36.
- [110] Ibrahim Saygin Topkaya, Hakan Erdogan, and Fatih Murat Porikli, “Counting people by clustering person detector outputs,” in *11th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2014, Seoul, South Korea, August 26-29, 2014*. 2014, pp. 313–318, IEEE.
- [111] Vladimir N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag New York, Inc., New York, NY, USA, 1995.
- [112] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal, “Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 1932–1939.
- [113] Vijay Mahadevan, Weixin Li, Viral Bhalodia, and Nuno Vasconcelos, “Anomaly detection in crowded scenes,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1975–1981.
- [114] S. Blunsden and R. B. Fisher, “The behave video dataset: ground truthed video for multi-person behavior classification,” *Annals of the BMVA*, vol. 2010, no. 4, pp. 1–11, Jan. 2010.
- [115] M. Enzweiler and D.M. Gavrilu, “Monocular pedestrian detection: Survey and experiments,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2179–2195, 2009.

-
- [116] D. Comaniciu and P. Meer, “Mean shift: a robust approach toward feature space analysis,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, May 2002.
- [117] Francisco Madrigal, Jean-Bernard Hayet, and Frdric Lerasle, “Improving multiple pedestrians tracking with semantic information,” *Signal, Image and Video Processing*, vol. 8, no. 1, pp. 113–123, 2014.
- [118] Sinead Williamson, Avinava Dubey, and Eric P. Xing, “Parallel markov chain monte carlo for nonparametric mixture models.,” in *International Conference on Machine Learning (ICML)*, 2013, vol. 28, pp. 98–106.