

Crowd Monitoring Using Computer Vision

by

David Andrew Ryan, BEng (Hons, 1st Class)

PhD Thesis

Submitted in Fulfilment

of the Requirements

for the Degree of

Doctor of Philosophy

at the

Queensland University of Technology

Image and Video Research Laboratory

Faculty of Built Environment and Engineering

December 2013

Abstract

Public places such as shopping centres and airports are monitored using closed circuit television (CCTV) in order to ensure normal operating conditions. Human operators are typically employed to perform this task, however as CCTV becomes more common it is impossible to monitor all viewpoints due to the number of cameras installed. In recent years, researchers have turned to computer vision in order to monitor crowds automatically. This thesis presents original contributions in four research areas: crowd counting, crowd flow monitoring, queue monitoring and anomaly detection.

The first major contribution of this thesis is in the field of crowd counting. Crowd size is a holistic description of a scene, therefore the majority of existing crowd counting techniques have utilised holistic image features to estimate crowd size. In this thesis, a novel approach is proposed which is based on *local* image features, which are specific to individuals and groups within a scene, so that the total crowd estimate is the sum of all group sizes. An extensive analysis shows that local features consistently outperform holistic features.

Existing approaches to crowd counting are also scene specific, as they are designed to operate in the same environment that was used to train the system. This thesis presents a novel algorithm which utilises camera calibration to achieve *scene invariance* by scaling features appropriately between viewpoints. Additionally,

multi camera crowd counting is achieved by using camera calibration to compensate for regions of overlap within a multi camera network.

The second major contribution of this thesis is in the field of crowd flow monitoring. A novel ‘virtual gate’ is proposed which counts pedestrians as they pass through a hypothetical line, or region of interest. Existing methods have typically fallen into one of two categories: object detection, or regression of optical flow. The virtual gate proposed in this thesis combines these two methods by detecting salient keypoints in an image and accumulating the optical flow of these feature points. Temporal windows and optical flow histograms are also proposed and shown to improve performance.

The third major contribution of this thesis is in the field of queue monitoring. There are currently very few methods for monitoring queue parameters such as queue length, growth rate, arrival rate and service rate. This thesis proposes a novel algorithm which combines crowd counting and virtual gates to monitor queue parameters automatically.

The fourth major contribution of this thesis is in the field of anomaly detection. Abnormal motion patterns may be indicative of dangerous or disruptive behaviour, and they may interfere with the operation of the aforementioned algorithms, therefore we seek to detect such events. Existing approaches typically reduce the optical flow field in some way (through quantisation, dimensionality reduction or histogram binning). This thesis proposes a novel visual representation called *textures of optical flow* which captures the properties of motion patterns in crowded environments by applying traditional textural features directly to an optical flow field. Results demonstrate that the proposed approach outperforms existing algorithms on benchmark anomaly detection sequences.

Keywords

Crowd Monitoring, Crowd Counting, Crowd Flow, Queue Monitoring, Anomaly Detection, Local Features, Scene Invariant, Multi Camera, Virtual Gate, Textures of Optical Flow, Computer Vision, Image Processing

Contents

Abstract	i
List of Tables	xv
List of Figures	xxiv
Notation	xxxiii
Acronyms & Abbreviations	xxxvii
Certification of Thesis	xxxix
Acknowledgments	xli
Chapter 1 Introduction	1
1.1 Motivation and Overview	1
1.2 Aims and Objectives	4

1.2.1 Crowd Counting	4
1.2.2 Crowd Flow	6
1.2.3 Queue Monitoring	6
1.2.4 Anomaly Detection	7
1.3 Scope of Thesis	8
1.4 Original Contributions	9
1.5 List of Publications Arising from this PhD Research	12

Chapter 2 Literature Review 15

2.1 Visual Surveillance Overview	15
2.1.1 Environment modelling	16
2.1.2 Feature extraction and/or Object detection	17
2.1.3 Higher-level event and behaviour understanding	19
2.1.4 Fusion of information from multiple cameras	20
2.2 Motion Detection	20
2.2.1 Optical Flow	21
2.2.2 Adaptive Background Models and Foreground Detection .	26
2.3 Crowd Counting	33

2.3.1	Holistic Approaches	37
2.3.2	Local Approaches	58
2.3.3	Summary	67
2.4	Crowd Flow Monitoring	68
2.5	Queue Analysis	71
2.6	Anomaly Detection	74
2.6.1	Object Detection and Trajectory Analysis	75
2.6.2	Motion Analysis	79
2.6.2.1	Bag of Words	79
2.6.2.2	Hidden Markov Models	82
2.6.2.3	Markov Random Fields	84
2.6.2.4	Other Models	87
2.6.2.5	Summary	88
2.7	Machine Learning	89
2.7.1	Gaussian Mixture Model	89
2.7.2	Hidden Markov Model	90
2.7.3	Gaussian Process Regression	95

Chapter 3 Crowd Counting using Local Features	99
3.1 Introduction	99
3.2 Crowd Counting using Local Features	102
3.2.1 Foreground Detection and Group Segmentation	103
3.2.2 Perspective Normalisation	106
3.2.3 Feature Selection	115
3.2.3.1 Texture	116
3.2.3.2 Size	124
3.2.3.3 Shape	126
3.2.3.4 Edges	128
3.2.3.5 Keypoints	131
3.2.3.6 Proposed Features	133
3.2.4 System Training	135
3.2.5 Regression	144
3.2.6 Tracking Module	146
3.3 Experimental Results	151
3.3.1 Operational Requirements	151

3.3.2	Benchmark Datasets	153
3.3.3	Cross Validation Experiments	157
3.3.3.1	Feature Evaluation	165
3.3.3.2	Regression Models	180
3.3.3.3	Holistic Features	186
3.3.4	Comparison To Other Algorithms	190
3.3.5	Tracking Algorithm	198
3.3.6	Long Term Performance	199
3.4	Visualisation	200
3.5	Summary	202
Chapter 4	Scene Invariant Crowd Counting	205
4.1	Introduction	205
4.2	Scene Invariant Crowd Counting	206
4.2.1	Camera Calibration	207
4.2.2	3D Pedestrian Template	210
4.2.3	Feature Normalisation	213
4.2.4	Operation of the Algorithm	215

4.3 Experimental Results	215
4.3.1 Benchmark Datasets	216
4.3.2 Cross Validation Experiments	219
4.3.2.1 Feature Evaluation	222
4.3.2.2 Regression Models	236
4.4 Summary	236
Chapter 5 Multi Camera Crowd Counting	241
5.1 Introduction	241
5.2 Multi Camera Crowd Counting	242
5.2.1 The Overlap Map	244
5.2.2 Density Map Modification	247
5.2.3 Pixel Density Assignment	248
5.2.4 Other Approaches	250
5.2.5 Ground Truth Annotation	253
5.3 Experimental Results	254
5.3.1 Crowd Counting Across Two Cameras	254
5.3.1.1 PETS 2006 Results	255

5.3.1.2 PETS 2009 Results	258
5.3.2 Scene Invariant Crowd Counting Across a Multi-Camera Network	258
5.3.3 Conclusions	263
5.4 Summary	264
 Chapter 6 Queue Monitoring 267	
6.1 Introduction	267
6.2 Queue Monitoring Framework	268
6.3 The Virtual Gate	273
6.3.1 Virtual Gate Overview	276
6.3.2 Optical Flow Histograms	278
6.3.3 Feature Selection	280
6.3.4 Ground Truth Annotation and System Training	281
6.4 Experimental Results	283
6.4.1 Benchmark Datasets	284
6.4.2 Virtual Gate	288
6.4.2.1 Feature Evaluation	290
6.4.2.2 Regression Models	306

6.4.3 Queue Monitoring	310
6.5 Summary	317
Chapter 7 Anomaly Detection	319
7.1 Introduction	319
7.2 Textures of Optical Flow	322
7.3 Anomaly Detection Algorithm	326
7.3.1 Robust Optical Flow	327
7.3.2 Spatio-Temporal Patches	328
7.3.3 Feature Extraction	331
7.3.3.1 Feature Extraction for Cuboids	332
7.3.3.2 Feature Extraction for Temporal Sequences	334
7.3.3.3 Efficient Implementation of Feature Extraction	336
7.3.4 Classification	338
7.3.4.1 Gaussian Mixture Model	339
7.3.4.2 Hidden Markov Model	340
7.3.4.3 Efficient Implementation of Classification	341
7.4 Experimental Results	343

7.4.1	Benchmark Datasets and Testing Protocol	343
7.4.2	Evaluation of Proposed Algorithm	344
7.4.3	Comparison to Other Algorithms	348
7.5	Summary	351
Chapter 8 Conclusion		355
8.1	Introduction	355
8.2	Summary of Contribution	356
8.3	Future Research	362
Appendix A Related to Chapter 3		363
A.1	Vanishing Point Calculation	363
A.2	Object Size Interpolation	364
A.3	Robust Regression using IRLS	365
Appendix B Related to Chapter 4		369
B.1	Regression Model Comparison Tables	369
Appendix C Hidden Markov Models		377
C.1	Implementation	377

C.2 Multiple Observation Sequences	379
Bibliography	383

List of Tables

2.1	High level summary of holistic crowd counting systems. See the main text in Section 2.3.1 for a full description of these algorithms and acronyms. The reader may also refer to the Acronyms section on page xxxvii.	36
3.1	Frame ranges used in this analysis, on the UCSD dataset [36], and corresponding performance using holistic textures only.	119
3.2	Feature categories used for crowd counting in this thesis. The subscript n indicates the index of the blob under consideration, although an equivalent holistic feature is represented by omitting the subscript, as shown in Equation 3.24 (p. 126), for example.	135
3.3	Various regions in an image. These regions are defined as <i>sets</i> of pixels.	140

3.4 The benchmark datasets used to evaluate the proposed crowd counting algorithm. The total number of frames is listed, and a subset of these frames have been annotated at regular intervals with ground truth, indicated using MATLAB notation (p. xxxiii). (The frames of the UCSD and Mall datasets are 1-indexed, while the remaining datasets are 0-indexed. We retain the indexing used by the original authors to avoid confusion.) Note that training and testing is performed on different sequences using K -fold cross validation (Section 3.3.3)	159
3.5 Summary of the various conditions in the benchmark datasets.	160
3.6 Parameters used for blob subsampling on the benchmark datasets. Any samples whose target falls within the specified range are subsampled at random, and the number of samples selected is specified by the ‘Samples’ column.	164
3.7 Comparison of features on the UCSD dataset	169
3.8 Comparison of features on the PETS 2009 dataset	170
3.9 Comparison of features on the Fudan dataset	171
3.10 Comparison of features on the Mall dataset	172
3.11 Comparison of features on the Grand Central dataset using local features. Holistic features are omitted as training failed due to insufficient data.	173

3.12 Average rank across all datasets , when features are ranked from 1 to 15 on each dataset. Values shown are not actual error rates, but rather an average ranking. (Note that the average rank for <i>holistic</i> features does not include the Grand Central dataset.) . . .	174
3.13 Comparison of regression models on the UCSD dataset	182
3.14 Comparison of regression models on the PETS 2009 dataset . . .	183
3.15 Comparison of regression models on the Fudan dataset	184
3.16 Comparison of regression models on the Mall dataset	185
3.17 Comparison of regression on the Grand Central dataset using local features. Holistic features are omitted as training failed due to insufficient data.	186
3.18 Average rank across all datasets , when regression models are ranked from 1 to 12 on each dataset. Values shown are not actual error rates, but rather an average ranking. (Note that the average rank for <i>holistic</i> features does not include the Grand Central dataset.)	187
3.19 Comparison of local and holistic features on all datasets using GPR. Each row represents a different feature set, and the best result (local or holistic) is indicated in bold, in terms of MAE, MSE and MRE.	189

3.20 Average rank of Kong's system across UCSD, PETS 2009, Fudan and Mall datasets, when regression models are ranked from 1 to 12 on each dataset. Values shown are not actual error rates, but rather an average ranking.	192
3.21 Results of Kong's system across UCSD, PETS 2009, Fudan and Mall datasets. The best-performing regression models are shown (Linear, NN).	192
3.22 Results of various systems on the UCSD dataset, following the experimental protocol of Chan [36]. Frames 600:1399 were designated for training and the remaining frames were set aside for testing (1:599, 1400:2000).	194
3.23 Results of various systems on the PETS 2009 dataset (sequences 13-57, 13-59 and 14-06). The average performance across all frames of these sequences is also reported.	196
3.24 Results of various systems on the Mall dataset, following the experimental protocol of Chen [42]. Frames 1:800 were designated for training and the remaining frames were set aside for testing (801:2000).	197
3.25 Evaluation of the proposed algorithm with and without the tracking module. For each dataset the best result (Tracking vs. No tracking) is indicated in bold.	198
3.26 Long term performance of the proposed algorithm on the UCSD dataset [36]. Performance is stable over the long term compared to textural features (Table 3.1, p. 119).	199

4.1	Summary of the various conditions in the PETS benchmark datasets [148, 149].	220
4.2	Summary of the various conditions in the QUT benchmark datasets.	220
4.3	The benchmark datasets used to evaluate the proposed scene invariant crowd counting algorithm.	221
4.4	Comparison of features on the PETS 2009 datasets, View 1 and View 2 . In each case the system is trained on the six other viewpoints.	225
4.5	Comparison of features on the PETS 2006 datasets, View 3 and View 4 . In each case the system is trained on the six other viewpoints.	226
4.6	Comparison of features on the QUT dataset, Camera 3 . The system is trained on the six other viewpoints.	227
4.7	Comparison of features on the QUT dataset, Camera 5 . The system is trained on the six other viewpoints.	228
4.8	Comparison of features on the QUT dataset, Camera 8 . The system is trained on the six other viewpoints.	229
4.9	Average rank across all seven datasets , when features are ranked from 1 to 15 on each dataset. Values shown are not actual error rates, but rather an average ranking.	230
4.10	Error rates for all datasets when all features are used. The <i>average</i> error rate across all datasets is also shown. Each dataset is weighted equally.	231

4.11 Average rank across all datasets , when regression models are ranked from 1 to 12 on each dataset. Values shown are not actual error rates, but rather an average ranking.	237
5.1 Results of multi camera crowd counting using Views 3 and 4 of the PETS 2006 dataset.	257
5.2 Results of multi camera crowd counting using Views 1 and 2 of the PETS 2009 dataset.	257
5.3 Results of multi camera crowd counting using Views 1, 2, and 3 of the PETS 2009 dataset.	262
6.1 The comparison between crowd counting and virtual gate algorithms presented in this thesis.	274
6.2 The benchmark datasets used to evaluate the proposed virtual gate algorithm. The total number of sequences and their duration is listed, as well as the throughput across all sequences (number of pedestrians passing through the gate).	284
6.3 The benchmark datasets used to evaluate the proposed queue monitoring algorithm.	288
6.4 Comparison of features, histograms and windows on Camera A	294
6.5 Comparison of features, histograms and windows on Camera B	295
6.6 Comparison of features, histograms and windows on Camera C	296
6.7 Comparison of features, histograms and windows on Camera D	297

6.8 Comparison of features, histograms and windows on Camera E	298
6.9 Average rank across all datasets , when all combinations of features, histograms and windows are ranked from 1 to 28 on each dataset. Values shown are not actual error rates, but rather an average ranking.	299
6.10 Comparison of regression models on Camera A	307
6.11 Comparison of regression models on Camera B	307
6.12 Comparison of regression models on Camera C	308
6.13 Comparison of regression models on Camera D	308
6.14 Comparison of regression models on Camera E	309
6.15 Average rank across all datasets , when regression models are ranked from 1 to 12 on each dataset. Values shown are not actual error rates, but rather an average ranking.	309
6.16 Results of queue length estimation using the crowd counting module.	310
6.17 Results of cross validation on the Queue dataset.	312
7.1 Feature vectors used with the proposed system. The notation $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$ represent unit vectors in the i and j dimensions respectively. The width and height of the pedestrian template (w, h) vary linearly as a function of j	335

7.2	The benchmark datasets used to evaluate the proposed anomaly detection algorithm. The total number of sequences is listed, and every frame has been annotated with ground truth (normal or abnormal).	344
7.3	Anomaly detection results of the proposed system on the UCSD anomaly detection datasets [125]. Equal error rate (EER) and area under curve (AUC) of the ROC are reported. Average performance across both datasets is also reported, and the best two results in each column are indicated in bold.	347
7.4	Anomaly detection results with reduced feature sets. In each experiment a different type of feature is omitted. The full feature vector is shown in the bottom row. The uniformity offsets $(\frac{w}{4}, 0), (0, \frac{h}{4})$ were selected, with a GMM classifier.	348
7.5	Anomaly detection results on the UCSD datasets for various algorithms [125]. Equal error rate (EER) and area under curve (AUC) of the ROC are reported.	350
B.1	Comparison of regression on PETS 2009, View 1	370
B.2	Comparison of regression on PETS 2009, View 2	371
B.3	Comparison of regression on PETS 2006, View 3	372
B.4	Comparison of regression on PETS 2006, View 4	373
B.5	Comparison of regression on QUT, Camera 3	374
B.6	Comparison of regression on QUT, Camera 5	375

B.7 Comparison of regression on QUT, Camera 8.	376
--	-----

List of Figures

2.1	General visual surveillance framework.	17
2.2	An example of motion segmentation.	19
2.3	Quadratic and Lorentzian penalty functions.	25
2.4	Pixel pairs are grouped into clusters. The adaptive background model [61] contains a group of clusters, each assigned a weight w_k indicating its likelihood, against which incoming clusters are compared.	29
2.5	Different levels of crowding density. Images from Xiaohua [197].	37
2.6	Three dimensional illustration of the GLCM.	39
2.7	Minkowski fractal dimension is calculated using edge detection followed by dilation using successively larger structuring elements.	41
2.8	Minkowski fractal dimension may not be reliable for crowd counting when the background scene contains strong gradients and effects of perspective.	43

2.9 Tree structure classifier used by Xiaohua [197], comprised of binary SVM classifiers.	44
2.10 Foreground detection on two scenes.	48
2.11 A high-level categorisation of local crowd counting algorithms.	58
2.12 Zhao [203] uses the foreground mask to segment individuals based on a discrete set of pose models.	60
2.13 A synthetic blob shape sampled at 50 perimeter points, and its reconstructed shape using 7 Fourier descriptor coefficients. Images from [63].	62
2.14 Localisation based approaches to crowd counting.	64
2.15 Current approaches to crowd flow monitoring.	69
2.16 Current approaches to queue monitoring.	73
2.17 The analogy between event detection and language processing.	80
2.18 Summary of Kim and Grauman's system. Images from [100].	84
3.1 Block diagram of the proposed system.	102
3.2 Reference lines used by Ma [121] for calculating a density map. P_1P_2 and P_3P_4 correspond to parallel lines on the ground plane. . .	107
3.3 Reference lines and pedestrians used by Chan [36] for calculating a density map.	109

3.4 Pedestrian detections from the PETS 2009 dataset [149] and pedestrian template models.	113
3.5 Comparison of frames 1 and 30 000. A subregion of interest is selected on the walkway and a histogram of pixel brightness for each frame is shown.	120
3.6 Performance of a texture-based holistic crowd counting system over time. These plots indicate a ‘drift’ between estimate and ground truth.	121
3.7 Relationship between crowd size and textural features at different time periods. Trend lines are plotted for each group of points to aid interpretation.	122
3.8 Relationship between crowd size and edge/foreground pixels. . . .	123
3.9 Four directions used for computing perimeter orientation histogram.	127
3.10 Keypoint feature extraction using Speeded-Up Robust Features (SURF) [18]. Keypoints are indicated with red dots.	132
3.11 Relationship between keypoint features and group size using two different weighting strategies: $\sqrt{S(i,j)}$ and $S(i,j)$	134
3.12 Relationship between normalised features and group size on the UCSD dataset [36].	136
3.13 Typical errors in foreground extraction.	138

3.14 The ground truth annotation process. Manual annotations (left) are overlayed on the foreground segmentation results (centre), and the region overlaps are used to automatically determine ground truth counts for each blob (right). Tiny blobs resulting from noise are assigned zero.	139
3.15 Example usage of the notation described in Table 3.3.	141
3.16 Ground truth for the UCSD dataset (frames 1270-1310).	143
3.17 Visualisation of blob tracking results. Groups of constant size are circled.	149
3.18 Images from each of the five benchmark datasets used to evaluate the proposed crowd counting algorithm.	158
3.19 K -fold cross validation procedure (with $K = 5$). The data is divided into K sets, and one set is withheld for testing during each experiment. The training and testing sets are rotated to test the model's capacity for generalisation.	161
3.20 Foreground detection result from frame 60 of the Mall dataset [42]. Note that the foreground includes small instances of noise in addition to the larger blobs which correspond to humans.	164
3.21 Cross validation results on the UCSD dataset	176
3.22 Cross validation results on the PETS 2009 dataset	177
3.23 Cross validation results on the Fudan dataset	178
3.24 Cross validation results on the Mall dataset	179

3.25	Cross validation results on the Grand Central dataset . Sequences are concatenated into a single plot due to their short length.	180
3.26	Visualisation of the proposed algorithm. Local counts are displayed on each blob, and the holistic count is shown at the top of the image.	201
4.1	A scene invariant crowd counting system is trained on a bank of reference viewpoints, so that it may be deployed on any number of novel viewpoints without additional training being required.	207
4.2	The camera calibration model includes the real world's 3D coordinate system, the camera's 3D coordinate system and the 2D sensor plane. Tsai's model also includes radial lens distortion as a subsequent step [189].	208
4.3	Ground truth annotations on the QUT camera network. A camera calibration technique [189] is used to project a human-sized cylindrical object into the scene.	211
4.4	Overhead view of the camera position relative to the centre of the cylindrical pedestrian template. Real world coordinates are shown in the (X_w, Y_w) plane.	212
4.5	Images from each of the seven viewpoints used to assess the performance of the proposed algorithm.	217
4.6	Scene invariant crowd counting results on PETS 2009, View 1 .	232
4.7	Scene invariant crowd counting results on PETS 2009, View 2 .	233

4.8	Scene invariant crowd counting results on PETS 2006, View 3.	233
4.9	Scene invariant crowd counting results on PETS 2006, View 4.	234
4.10	Scene invariant crowd counting results on QUT, Camera 3.	234
4.11	Scene invariant crowd counting results on QUT, Camera 5.	234
4.12	Scene invariant crowd counting results on QUT, Camera 8.	235
5.1	There are not always direct correspondences between groups from different camera angles.	243
5.2	Examples of the overlap map at two points in the PETS 2006 and PETS 2009 datasets.	246
5.3	The PETS 2009 database with ROIs highlighted and a sample of labelled pedestrian annotations. Pedestrians 1-3 indicate unique coverage areas, while pedestrian 4 indicates an overlap region. . .	251
5.4	ROI cutoff for a two-camera setup.	252
5.5	ROI cutoff for a three-camera setup. Views 1 and 3 shown; View 2 remains unchanged from Figure 5.4(b).	252
5.6	Results of multi camera crowd counting using Views 3 and 4 of the PETS 2006 dataset.	256
5.7	Results of multi camera crowd counting using Views 1 and 2 of the PETS 2009 dataset.	259

5.8 Results of multi camera crowd counting using Views 1, 2 and 3 of the PETS 2009 dataset.	261
6.1 Design of the proposed queue monitoring system.	269
6.2 The components of a virtual gate.	276
6.3 Visual representation of frames and windows. Optical flow based features are computed between frame pairs and accumulated across each window.	277
6.4 A feature point passes through the ROI. Optical flow vectors describe its displacement between consecutive frames. The component of the flow in direction d contributes to the <i>aligned</i> flow. These components sum to w approximately.	279
6.5 Relationship between features and number of people passing through the gate for Camera C (as described in Section 6.4.1). The lower histogram bin with range $[0, 0.05]$ is shown in the left hand column; the middle histogram bin with range $[0.05, 0.25]$ is shown in the middle column; and the upper histogram bin with range $[0.25, \infty)$ is shown in the right hand column.	282
6.6 Camera A	285
6.7 Camera B	285
6.8 Camera C	286
6.9 Camera D	286
6.10 Camera E	287

6.11 Queue dataset	289
6.12 Cross validation results on Camera A	300
6.13 Cross validation results on Camera B	301
6.14 Cross validation results on Camera C	302
6.15 Cross validation results on Camera D	303
6.16 Cross validation results on Camera E	304
6.17 Sequence 6, Frame 4200 from Camera D. A higher level of occlusion is observed for this sequence.	305
6.18 Queue length monitoring using the crowd counting module.	311
6.19 Cross validation results on the Queue Dataset using the virtual gate.	313
6.20 Cumulative performance of the virtual gate algorithm across Se- quences A and B. Both arrivals and services are shown.	314
6.21 Results of the proposed queue monitoring algorithm operating on Sequence A . Queue length (Q_t) is estimated using the crowd counting module; arrivals (A_t) and services (S_t) are estimated using the virtual gate modules.	315
6.22 Results of the proposed queue monitoring algorithm operating on Sequence B . Queue length (Q_t) is estimated using the crowd counting module; arrivals (A_t) and services (S_t) are estimated using the virtual gate modules.	316

7.1	Typical abnormalities in crowded pedestrian scenes.	321
7.2	Overview of the anomaly detection system.	327
7.3	Spatio temporal patches may be visualised as a rectangular prism for GMM classification, or as a set of connected slices for HMM classification.	329
7.4	Patches of variable size are used based on the size of the pedestrian template centred at each location in the scene. Various patch sizes are shown on each image.	331
7.5	The integral image can be used to efficiently extract features from spatio temporal patches.	336
7.6	ROC curves for the UCSD anomaly detection datasets.	349
7.7	Detection results for the Ped2 dataset (14 sequences) at the equal error rate. Normal frames are represented by the value 0, abnormal frames are represented by the value 1.	350
7.8	Partial detection of a slow-moving bicycle in sequence 10 of the Ped2 dataset [125].	351
A.1	Least squares and Tukey bisquare penalty functions, their deriva- tives and weight functions [70].	367

Notation

:	MATLAB [132] notation for a set of numbers:
	$a : b = \{a, a + 1, a + 2, \dots, b\}$
	$a : b : c = \{a, a + b, a + 2b, \dots, c\}$
A	Area feature.
A_n	Area of segment n .
B	Set of foreground pixels within the region of interest.
B_n	Set of foreground pixels within the region of interest belonging to segment n .
$B_{t,n}$	Set of foreground pixels belonging to segment n in frame t .
C	Connected component label image. $C(i, j)$ is the index of the segment to which pixel (i, j) belongs.
$C_{p,n}$	Contribution of person p to segment n .
D	Dimensionality.
d	Image of crowding density, $d(i, j)$ refers to pixel (i, j) .
d^v	Image of crowding density in viewpoint v .
E	Edge pixel image. $E(i, j) \in \{0, 1\}$, where 1 represents an edge at (i, j) and 0 represents a non-edge.
\mathbf{f}	Targets during regression.
f_n	The n th target.
F	Foreground detection image. $F(i, j) \in \{0, 1\}$

G_x, G_y	Gradient images computed with horizontal and vertical Sobel kernels, respectively.
G	Gradient image. Each element $G(i, j)$ is a vector with horizontal component $G_x(i, j)$ and vertical component $G_y(i, j)$. The magnitude of the gradient is $ G(i, j) $ and its angle of orientation is $\angle G(i, j)$.
G	Grey level cooccurrence matrix (GLCM).
h	Index, used for histogram bin.
H	Histogram of oriented gradients. $H(h)$ is the value in the h th bin.
H_n	Histogram of oriented gradients of segment n . $H_n(h)$ is the h th bin.
I	Image, containing pixels $I(i, j, k) \in \{1, 2, \dots, 255\}$. The index k is omitted for single plane images.
(i, j)	Image plane coordinate system where i and j denote horizontal and vertical coordinates respectively.
k	Index, used for image plane.
K	Keypoints feature. The superscript indicates the type of feature detector used (e.g. K^{SURF}).
κ	Set of keypoints detected in an image. The superscript indicates the type of feature detector used (e.g. κ^{SURF}).
\mathbf{K}	Covariance matrix in GPR. Also \mathbf{K}^* and \mathbf{K}^{**} .
L	Perimeter length feature.
L_n	Perimeter length of segment n .
M	Set of pixels belonging to region of interest mask.
m	Index, used for blob segments.
n	Index, used for blob segments.
N_i, N_j, N_k	Image dimensions (width, height and planes). $N_k = 1$ for greyscale images and $N_k = 3$ for RGB colour images.

$O_{i,j}^v$	Overlap map for viewpoint v .
$O_t(m, n)$	Overlap of blobs between frames.
\mathbf{p}	Pixel index, for example $\mathbf{p} = (i, j)$ or (i, j, k) depending on image dimensionality. $I(\mathbf{p})$ represents the value of pixel \mathbf{p} in image I .
P	Set of pixels lying on blob perimeters.
P_n	Set of pixels lying on the perimeter of blob n .
Q_p	Quantity of person p within the ROI.
Q_s, Q_h	Holistic ground truth. Subscript s refers to ‘soft’ ground truth, and subscript h refers to ‘hard’ ground truth. $Q_s \in \mathbb{R}$ and $Q_h \in \mathbb{Z}$.
R_p	Pedestrian template for the p th annotated person in a frame. A pedestrian template is a set of pixels which occupies a similar region in the image plane to the pedestrian. The magnitude $ R_p $ represents the <i>area</i> of this template (number of pixels).
$R_{i,j}$	Pedestrian template for a hypothetical person centred at (i, j) in the image plane.
S	Density map used to weight image features.
t	Frame number (time).
T	Threshold value. Subscript differs for each threshold.
u	Superscript used to indicate camera ID (viewpoint). Horizontal component of optical flow.
v	Superscript used to indicate camera ID (viewpoint). Vertical component of optical flow.
W	Width, generic usage
\mathbf{x}	Feature vector.
\mathbf{x}_n	Feature vector for segment n .

- (x, y, z) Ground plane coordinate system in the real world. z represents the vertical component perpendicular to the (x, y) ground plane.

Acronyms & Abbreviations

2D	2-dimensional
3D	3-dimensional
BBN	Bayesian belief network
CCTV	Closed circuit television
DC	Direct current. In signal processing, the DC component/coefficient refers to the zero-frequency component (average value) of a signal.
DCT	Discrete cosine transform
DWT	Discrete wavelet transform
EKF	Extended Kalman filter
FAST	Features from accelerated segment test (corner detection [160])
FPS	Frames per second
GLCM	Grey level cooccurrence matrix
GMM	Gaussian mixture model
GPR	Gaussian process regression
GT	Ground truth
HMM	Hidden Markov model
HOG	Histogram of oriented gradients
ID	Identity
KLT	Kanade-Lucas-Tomasi (feature tracker [174, 186])
MAE	Mean absolute error
Max	Maximum

Min	Minimum
MRE	Mean relative error
MSE	Mean square error
NN	Neural network
ML	Machine learning
PCA	Principal component analysis
PDF	Probability density function
PETS	Performance evaluation of tracking systems (IEEE international workshop [148, 149, 150])
RGB	Red green blue (colour model)
SURF	Speeded-up robust features (feature detection [18])
SVM	Support vector machine
TIOCM	Translation invariant orthonormal Chebyshev moments
XML	Extensible markup language

Certification of Thesis

The work contained in this thesis has not been previously submitted for a degree or diploma at any other higher educational institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

QUT Verified
Signed: _____
Date: 19 Dec 2013

Acknowledgments

I would like to thank my supervisors, Professor Sridha Sridharan, Associate Professor Clinton Fookes and Dr Simon Denman, for their invaluable support throughout my PhD. I would not have been able to complete this thesis without their ongoing feedback and advice, as well as the excellent resources that they have provided in the SAIVT laboratory. I am especially grateful for the many hours of time spent reviewing my articles and research over the last few years. I would also like to thank the members of the SAIVT laboratory for their feedback, advice and friendship during this time.

I would also like to thank my family and friends for their love and support throughout my PhD. Thanks for the many conversations and laughs which helped take the pressure off during stressful times. Finally I would like to thank Tenille for her endless encouragement and love, without which I could not have seen this through to completion.

DAVID ANDREW RYAN

*Queensland University of Technology
December 2013*

Chapter 1

Introduction

1.1 Motivation and Overview

A crowd is a large collection of people within a public space. In public places such as railway stations, airports and shopping centres, it is not possible to monitor every individual person for suspicious behaviour. Instead, the problems posed in crowded environments arise from the crowd's collective properties: congestion, excitement, fighting, rioting and mass panic. As Gustav Le Bon [25] wrote in his classic book on social psychology, *The Crowd: A Study of the Popular Mind*:

Under certain given circumstances, and only under those circumstances, an agglomeration of men presents new characteristics very different from those of the individuals composing it. The sentiments and ideas of all the persons in the gathering take one and the same direction, and their conscious personality vanishes. A collective mind is formed, doubtless transitory, but presenting very clearly defined characteristics. The gathering has thus become what, in the absence

of a better expression, I will call an organized crowd, or, if the term is considered preferable, a psychological crowd. It forms a single being and is subject to the law of the mental unity of crowds.

The infectious nature of human emotions can lead to *collective* behaviours that poses significant threats to safety and security. In short, “a crowd is something other than the sum of its parts” [58].

Closed circuit television (CCTV) provides a means for security personnel to monitor crowds, in order to prevent or minimise these problems. Crowd monitoring is concerned with the holistic properties of pedestrians in a scene. These properties include crowd size, density, growth rate and flow patterns, as well as the detection of abnormal events.

Unfortunately, CCTV provides an incomplete picture of the world, one which is “disjoint and fragmented” [118]. Security systems employing CCTV are managed from a control room containing several monitors, which are observed by a human operator. In public places such as railway stations and airports, the operator is sensitive to specific events, suspicious behaviour and objects. Differentiating such events from routine activities is not easy, and requires constant, focused attention:

In undertaking their work, station staff... do no just passively monitor the general scene for events to occur and then undertake actions, but view the images, in detail, with regard to what may be organisationally appropriate next actions to undertake. [118]

As CCTV becomes ubiquitous, however, it grows increasingly difficult for humans to monitor all of the available data, due to the sheer number of cameras installed.

For example, there are estimated to be between 1.85 million [77] and 4.2 million [142] CCTV cameras installed in the United Kingdom alone.

Furthermore, crowd monitoring may be limited by the “short attention span and lack of adequate training and experience of human operators” [24]. In most cases, security footage is used to investigate events after they occur, rather than to generate real-time alerts during an evolving situation.

In recent years, researchers have turned to computer vision based surveillance technologies to monitor crowds automatically from closed-circuit television footage. Automated visual surveillance is an active field of research, due to the large number of problems which remain unsolved. Existing research into pedestrian tracking and human action recognition is well established [90, 137], however the problems posed by crowded environments are even more challenging due to the presence of occlusions and multiple agents. The techniques proposed in the literature are generally evaluated in controlled environments over a short time frame, whereas most practical applications will require long term surveillance over a wide range of conditions. This introduces additional problems, such as changing lighting conditions and changes in context over time.

Commercial surveillance companies advertise products which can perform pedestrian tracking, detect perimeter intrusions, and count people [5, 85]. However, according to Boghossian [24]:

[T]he consistent and widespread message from end-users of imaging systems has been that this technology is not yet sufficiently robust or generically applicable to reach its full commercial potential. Either systems do not work over a sufficiently wide range of environmental conditions and perturbing factors or they are not applicable in a sufficiently broad range of scenarios without extensive retraining and

customisation to each new application.

There is a trade-off between accuracy and practicality amongst these technologies. A system that is tailored heavily to a specific environment will be more accurate under those controlled conditions, for example, but the system's ability to scale between changing viewpoints and environmental conditions will be compromised, diminishing its practicality. This is largely a symptom of using limited datasets to evaluate algorithms, which can result in overfitting and poor generalisation.

This research is motivated by the need for more practical visual surveillance solutions in crowded environments. This is particularly true in occluded crowds, where traditional object tracking algorithms struggle, or fail, to operate.

1.2 Aims and Objectives

This research aims to expand the capabilities of current intelligent visual surveillance systems to monitor crowds, and to improve their robustness when employed in practical situations. Original contributions are made in the following areas.

1.2.1 Crowd Counting

In large public places, it is often not possible to monitor every person's individual behaviour due to crowd size. Instead, crowd properties can be monitored, such as the distribution of people throughout the space and the total number of people in the scene. Crowd size may be an indicator of security threats such as fighting, rioting, violent protest, mass panic and excitement.

Even in peaceful crowds, size may be an indicator of congestion, delay or other

abnormality. Crowd related information can also be used to provide important business intelligence. For example, the distribution of crowds throughout a shopping complex or large retail store may be used to analyse consumer shopping patterns, while the overall crowd size may be monitored to assess store performance over time. For these reasons, the estimation of crowd size is a very fundamental task within the field of crowd monitoring.

As crowd size is a *holistic* description of the scene, the majority of crowd counting techniques have utilised holistic image features to estimate crowd size. Due to the wide variability in crowd behaviours, distribution, density and overall size, it can be difficult to achieve proper generalisation using holistic approaches. This thesis aims to improve the practicality of existing crowd counting algorithms by incorporating *local* features, which are specific to individuals and groups within an image, to estimate the crowd size and its distribution across a scene. While existing techniques have used similar features such as foreground detection, they are analysed at a holistic level. Local features are proposed to estimate the number of people within each *group*, so that the total crowd estimate is the sum of all group sizes.

Even though large-scale CCTV networks are becoming increasingly common, automated crowd counting is not widely deployed. One of the largest barriers to full deployment of this technology is the requirement to train each camera independently, which is both time-consuming and expensive. Therefore this thesis aims to use camera calibration to scale features between multiple viewpoints, by taking into account the relative sizes of objects in these scenes. It would be highly practical to have a scene invariant crowd counting system which may be trained on one camera and then deployed for counting on another. A system could then be trained on a large bank of data from various cameras, before counting performed on a new viewpoint. In practice, a system which has been pre-trained

on numerous camera viewpoints can operate as a turn-key solution for crowd counting across a wide range of unseen environments.

1.2.2 Crowd Flow

Crowd flow monitoring pertains to the *motion* of a crowd rather than its total size. For example, the number of pedestrians entering or exiting a location is a useful metric for private and public organisations alike.

This thesis presents a crowd flow monitoring algorithm with an application to queue monitoring. A novel *virtual gate* algorithm has been developed with the purpose of detecting pedestrians as they pass through a region of interest. This enables a system to estimate the number of pedestrians entering or exiting a queue, as well as the rates of entrances and exits over time.

1.2.3 Queue Monitoring

Estimating queue parameters is an important part of many business operations, including retail shops, public transport hubs and airports. When these queues are particularly large, as at airport check-in counters with multiple service stations, it is difficult for human operators to quantitatively assess queue parameters such as wait time, throughput rate and overall queue size.

This thesis proposes a combination of crowd counting and virtual gate technologies in order to calculate the queue parameters. These include:

1. **Queue length:** the number of people waiting to be served.
2. **Throughput rate:** the number of people serviced per minute or hour at

the front of the queue. Additionally, the system can monitor the number of people arriving per minute/hour at the end of the queue.

3. **Growth rate:** the speed at which the queue is growing (or shrinking) in size.
4. **Wait time:** the expected time that a person at the end of the queue will wait to be serviced.

Successful queue management requires active monitoring of these properties in order to determine whether additional service stations should be opened or closed, and to plan for future events.

1.2.4 Anomaly Detection

The aforementioned algorithms are based on the assumption that the scene being monitored is occupied by humans under normal circumstances. However when these assumptions are violated the operation of the system may be compromised. This might occur, for example, if a bicycle or vehicle enters a scene designed for pedestrians only. In order to detect violations of these assumptions an anomaly detection algorithm is proposed based on local motion features. A novel representation of motion patterns is proposed, called *textures of optical flow*.

This feature is combined with location and velocity features in order to classify local events as normal or abnormal. The proposed algorithm may also be used for security or review purposes. Typically we seek to detect abnormal events such as pedestrians moving with excessive speed (running or skateboarding), pedestrians in forbidden or restricted areas (spatial anomalies) and non-human objects in a pedestrian walkway (such as vehicles or bicycles).

1.3 Scope of Thesis

The scope of this thesis is restricted to public spaces which are typical of those monitored by security cameras, such as a pedestrian walkway or a shopping mall. It is assumed that the scene being monitored contains pedestrians and not other objects such as vehicles or animals.

The assumption of human-only crowds is applicable to Chapters 3-6, which cover the topics of crowd counting, crowd flow monitoring and queue monitoring. In some instances these assumptions may be violated, and we seek to identify those events by means of an anomaly detection algorithm, as presented in Chapter 7.

Other visual media, such as movies and television, often contain crowds but are beyond the scope of this thesis. Instead, we use data obtained from fixed security cameras in real world environments, as is the intended application of the technology.

This thesis builds upon existing research in the field of visual surveillance. For example, motion detection and optical flow are used extensively in this research. However, the focus of this thesis is not to improve upon the state of the art in these specific fields. Instead, this research makes use of the most popular or best performing implementations of these algorithms, where possible. This thesis focuses on the bigger picture of crowd monitoring and its end results. Future developments in these fields (motion detection and optical flow) can be incorporated into the proposed algorithms by substituting the relevant modules, without any change to the surrounding crowd monitoring framework.

1.4 Original Contributions

This thesis presents a number of original contributions to the field of crowd monitoring using computer vision. The following contributions are made with regards to *crowd counting*:

- A novel approach to crowd counting is presented, which replaces holistic features with localised feature extraction. Instead of estimating a crowd globally, the problem is broken down into groups detected using a foreground detection algorithm, and features are extracted from each segment. (Chapter 3)
- Prior to crowd counting, individuals must be annotated in sample images in order to train a system. This thesis presents a unique method of applying local annotations for each foreground segment, based only on simple ‘dot’ annotations which are easy to perform. A pedestrian template model is used to estimate the region occupied by the person represented by each annotation, and ground truth is apportioned to overlapping foreground segments accordingly. (Chapter 3)
- A novel method of refining the estimates using group tracking is described. Previous approaches have generally ignored the fact that consecutive frames are likely to have similar pedestrian counts, although some have applied smoothing on a holistic level. By identifying and tracking groups as they move through an image, smoothing can be applied on a local level. The confidence of the estimate provided by GPR can also be used to weight each estimate. (Chapter 3)
- A comprehensive analysis of various crowd counting algorithms across five datasets is performed. These capture various crowding properties such as

indoor and outdoor scenes, as well as colour and greyscale images at various resolutions.

- The crowd counting methodology is extended to multiple cameras by normalising features across viewpoints. The system uses camera calibration and real-world reference coordinates to determine this density map. This approach enables *scene invariant crowd counting*, in which a system may be trained on one or more viewpoints and then deployed on one or more other viewpoints for counting, without any additional training required. (Chapter 4)
- Three new benchmark datasets for crowd counting are introduced, with ground truth annotations and camera calibration parameters. These sequences feature challenging reflections, shadows and lighting fluctuations, and are used to demonstrate the efficacy of the proposed methodology. (Chapter 4)
- A comprehensive evaluation on seven calibrated datasets demonstrates the scene invariance of the proposed algorithm. (Chapter 4)
- Two novel methods for compensating for camera overlap when counting crowds in a *multi-camera* environment. These make use of a new concept called the *overlap map*. (Chapter 5)
- The combination of scene invariance and multi camera crowd counting is demonstrated using a three-camera setup. A mean relative error of 6-8% was observed in this environment, using a crowd counting system which was trained on a completely different environment. (Chapter 5)

The following contributions are made with regards to *crowd flow* monitoring:

- A virtual gate algorithm is presented which makes use of feature points passing through a region of interest. The accumulation of optical flow at these points and the introduction of optical flow histograms are both demonstrated to improve accuracy of the algorithm. The use of sub-sequences or *windows* are also shown to improve performance, and this is analogous to the use of local features within a crowd counting system. (Chapter 6)

The following contributions are made with regards to *queue monitoring*:

- The novel combination of crowd counting and virtual gate technologies to monitor queue statistics such as arrival rate, service rate and queue length. (Chapter 6)
- A queue monitoring framework which establishes the relationship between queue statistics and allows these statistics to be derived from others in the absence of adequate video coverage. Derived statistics such as growth rate and wait time are also presented. (Chapter 6)
- An evaluation on real world queue data. Unlike existing crowd datasets which are captured in unordered public spaces such as shopping malls and outdoor walkways, the datasets used in this analysis were captured from six cameras at the Brisbane International Airport. The queue analysis was performed on surveillance footage obtained from the baggage check-in counter. (Chapter 6)

The following contributions are made with regards to *anomaly detection*:

- A novel visual representation called textures of optical flow, which is specifically formulated to detect anomalous motion patterns in crowded scenes.

The proposed representation measures the uniformity of a flow field in order to detect anomalous objects such as bicycles and vehicles; and can be combined with spatial information to detect other forms of abnormality. (Chapter 7)

- An investigation into optimal features and classification models to be used in conjunction with the proposed system. It was demonstrated that the proposed approach outperforms state-of-the-art anomaly detection algorithms on two large, publicly-available datasets. (Chapter 7)
- A number of optimisations were proposed to speed up computation of the proposed algorithm. These included the use of integral images to rapidly calculate patch based features; the use of the K-Means++ algorithm to improve pre-clustering; and foreground masking to reduce the number of data samples without sacrificing relevant motion information. (Chapter 7)

1.5 List of Publications Arising from this PhD Research

The journal articles that have been published as part of this research are as follows:

1. David Ryan, Simon Denman, Clinton Fookes and Sridha Sridharan. Scene invariant multi camera crowd counting. In *Pattern Recognition Letters*. Elsevier, 2013. In Press. [166]
2. Hajananth Nallaivarothayan, David Ryan, Simon Denman, Sridha Sridharan, Clinton Fookes and Andry Rakotonirainy. Detecting anomalous events

at railway level crossings. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit.* 227(5):539-553, September 2013. [140]

The journal articles that have been submitted as part of this research are as follows:

1. **David Ryan**, Simon Denman, Sridha Sridharan and Clinton Fookes. An evaluation of crowd counting methods, features and regression models. Submitted to *Computer Vision and Image Understanding*. Elsevier, 2013.

The book chapters that have been published as part of this research are as follows:

1. **David Ryan**, Simon Denman, Sridha Sridharan and Clinton Fookes. Scene invariant crowd counting and crowd occupancy analysis. In *Video Analytics for Business Intelligence*, pages 161-198. Springer-Verlag, 2012. [168]

The conference articles that have been published as part of this research are as follows:

1. **D. Ryan**, S. Denman, C. Fookes, and S. Sridharan. Crowd counting using multiple local features. In *Digital Image Computing: Techniques and Applications, 2009. DICTA '09.*, pages 81-88, December 2009. [163]
2. **D. Ryan**, S. Denman, C. Fookes, and S. Sridharan. Crowd counting using group tracking and local features. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pages 218-224, September 2010. [164]

3. **D. Ryan**, S. Denman, S. Sridharan, and C. Fookes. Scene invariant crowd counting. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*, pages 237-242, December 2011. [167]
4. **D. Ryan**, S. Denman, C. Fookes, and S. Sridharan. Textures of optical flow for real-time anomaly detection in crowds. In *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, pages 230-235, September 2011. [165]
5. H. Nallaivarothayan, **D. Ryan**, S. Denman, S. Sridharan and C. Fookes. Anomalous event detection using a semi-two dimensional hidden markov model. In *Proceedings of the 2012 International Conference on Digital Image Computing Techniques and Applications (DICTA 12)*, pages 1-7, December 2012. [139]
6. C. Fookes, S. Denman, R. Lakemond, **D. Ryan**, S. Sridharan, and M. Piccardi. Semi-supervised intelligent surveillance system for secure environments. In *Industrial Electronics (ISIE), 2010 IEEE International Symposium on*, pages 2815-2820, July 2010. [69]
7. **D. Ryan**, S. Denman, C. Fookes, and S. Sridharan. Scene invariant crowd counting for real-time surveillance. In *Signal Processing and Communication Systems, 2008. ICSPCS 2008. 2nd International Conference on*, pages 1-7, December 2008. [162]

Chapter 2

Literature Review

This chapter reviews the literature related to crowd monitoring using computer vision. The review begins with a general overview of visual surveillance systems in Section 2.1. A fundamental step in many visual surveillance systems is motion detection, using for example background models or optical flow, and these low-level algorithms which form the basis for many other visual surveillance tasks are discussed in Section 2.2. With this groundwork, subsequent sections review the literature directly related to this thesis: Section 2.3 covers crowd counting, Section 2.4 discusses crowd flow monitoring, Section 2.5 reviews queue analysis, and Section 2.6 discusses anomaly detection.

2.1 Visual Surveillance Overview

Traditional visual surveillance systems are comprised of a number of modules, each performing a unique function. A highly generalised visual surveillance system is depicted in Figure 2.1. This surveillance network consists of multiple

cameras and a sequence of processing steps which are classified broadly into the following categories:

- Environment modelling.
- Feature extraction and/or Object detection.
- Higher-level event and behaviour understanding.
- Fusion of information from multiple cameras.

The information from each camera is fused to acquire operationally meaningful data. The terminology used here is very general: a surveillance system may be designed to track the individual body parts of a single person; or to track the position of multiple individuals in a scene; or to measure the holistic properties of a crowd. Each of the stages in this framework are described briefly in the following sections.

2.1.1 Environment modelling

Prior knowledge about the scene and camera orientation can be exploited using 2D or 3D models. For example, the background of a scene is usually static, with variations in the image occurring due to signal noise and lighting fluctuations over time. In order to model the 2D background image, adaptive approaches are commonly used: each pixel is represented by a probability distribution, such as a Gaussian Mixture Model (GMM), and this is incrementally adjusted over time. The details of such background models are discussed in Section [2.2](#).

Another important component of environment modelling is camera calibration, which describes the relationship between the image plane and the scene (typically

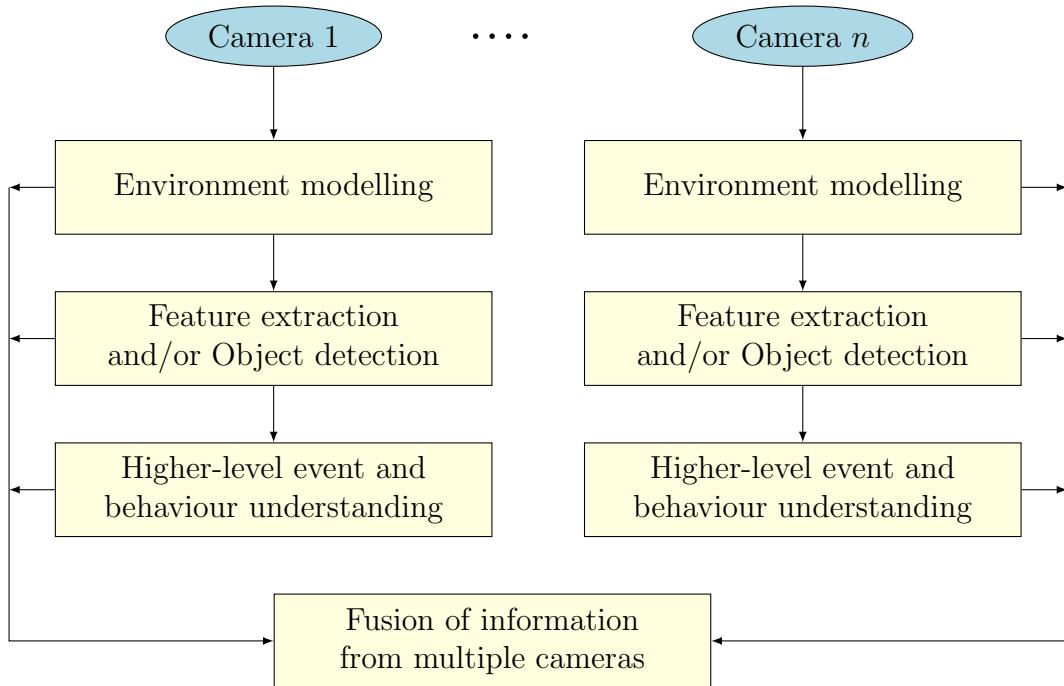


Figure 2.1: General visual surveillance framework.

the ground plane). Homography matrices can be used to describe the transformation between these planes [4], while popular camera calibration techniques such as Tsai's camera model [188, 189] take into account non-linear factors such as radial lens distortion. Camera calibration is discussed in greater detail in Section 4.2.1 (p. 207).

2.1.2 Feature extraction and/or Object detection

Feature extraction is a general term used to describe low level image processing, whereby the pixels of an image are processed into a meaningful set of descriptors. The low level features extracted from the surveillance video are subsequently utilised for higher-level processing and understanding. Depending on the target application, this module might include:

- **Motion segmentation.** The first processing stage in many surveillance applications is motion segmentation, which is used to segment foreground objects from the background. The most commonly used method is Stauffer-Grimson background subtraction [178]. An example of motion segmentation is shown in Figure 2.2.
- **Feature extraction.** The features in an image can be extracted to create a meaningful descriptor or set of descriptors which are then processed at a higher level. For example, holistic image features such as texture have been used for crowd density analysis [128, 153] while local image features extracted from ‘spatio temporal patches’ have been used for anomaly detection [106, 107, 108].
- **Object classification.** Foreground segments are commonly referred to as ‘blobs’ when their underlying object is unknown. The purpose of object classification is to determine the object class to which each segment belongs (typically humans or cars). This may be performed using shape features, such as blob area, or the aspect ratio of the bounding box [51]; or motion features such as periodicity [56] and rigidity [114].
- **Person detection.** Humans can be detected in images using popular approaches such as Dalal’s histogram of oriented gradients [57] and Felzenszwalb’s part-based models [67, 68]. Face detection methods include Viola’s cascade of simple image features [191].
- **Trajectory extraction.** Objects are tracked over time and their trajectories are used for subsequent analysis. A review of existing tracking research is discussed in [199]. Similarly, trajectories of image keypoints can be extracted using methods such as the Kanade-Lucase-Tomasi (KLT) feature tracker [174, 186].



(a) Frame 1100 of the UCSD Pedestrian Database [36, 37].

(b) Corresponding motion segmentation (or foreground detection), obtained using an adaptive background model [59, 60].

Figure 2.2: An example of motion segmentation.

This thesis is concerned with the properties of human crowds. The specific features which have been used for crowd monitoring are discussed in Section 2.3 (crowd counting), Section 2.4 (crowd flow), Section 2.5 (queue monitoring) and Section 2.6 (anomaly detection).

2.1.3 Higher-level event and behaviour understanding

Higher level understanding refers to information relevant to the end user, a human operator. The system uses machine learning tools to perform its intended task, by processing the low level features extracted in the previous modules. The target application may include:

- **Behaviour understanding.** Tracking results are used to analyse trajectories and recognise behaviour of individuals [92, 151, 179].
- **Personal identification.** Height, facial appearance and walking gait are the main biometric features used for personal identification [90].
- **Crowd monitoring.** The focus of this thesis is on monitoring crowded

environments, which includes estimation of crowd size [36, 103], crowd flow analysis [98, 109] and anomaly detection [3, 125].

Machine learning techniques such as support vector machines (SVM), neural networks (NN), hidden Markov models (HMM) and Gaussian mixture models (GMM) are commonly used for classification and regression.

2.1.4 Fusion of information from multiple cameras

Fusion of information from multiple cameras is aided by the use of camera calibration as described in Section 2.1.1 which enables real-world positions to be mapped across viewpoints. Correspondence between objects from multiple viewpoints can improve the system’s understanding of the scene. It may be necessary to account for blind spots or overlaps between the viewpoints.

2.2 Motion Detection

Before reviewing the crowd monitoring literature, this section describes some of the fundamental visual surveillance tasks which underpin the existing research. A fundamental step in most visual surveillance systems is motion detection [90], as this allows subsequent tasks to be performed such as object classification or crowd counting. This section reviews two types of motion detection: Section 2.2.1 discusses optical flow, and Section 2.2.2 discusses foreground detection using adaptive background modelling.

2.2.1 Optical Flow

Optical flow is the “distribution of apparent velocities of movement of brightness patterns in an image” [86]. It refers to the motion occurring at each pixel with respect to the image coordinate plane, and may be thought of as the distance a pixel moves from one frame to the next.

The brightness of a point at coordinate (i, j) at time t is denoted $I(i, j, t)$. The *brightness constraint* is based on an assumption that the brightness of point remains constant when moving by a displacement of $(\delta i, \delta j, \delta t)$:

$$I(i, j, t) = I(i + \delta i, j + \delta j, t + \delta t) \quad (2.1)$$

$$= I(i, j, t) + \delta i \frac{\partial I}{\partial i} + \delta j \frac{\partial I}{\partial j} + \delta t \frac{\partial I}{\partial t} + \epsilon \quad (2.2)$$

where ϵ represents second order and higher order terms. For reasonably small $(\delta i, \delta j, \delta t)$, the value of ϵ is small and can be neglected. Therefore,

$$\frac{\partial I}{\partial i} \frac{\delta i}{\delta t} + \frac{\partial I}{\partial j} \frac{\delta j}{\delta t} + \frac{\partial I}{\partial t} = 0 \quad (2.3)$$

For brevity the left hand side of this equation may be written,

$$\xi_b = I_i u + I_j v + I_t = 0 \quad (2.4)$$

where $u = \frac{\delta i}{\delta t}$ and $v = \frac{\delta j}{\delta t}$ are the unknown values of the optical flow, and I_i , I_j and I_t represent the horizontal, vertical and time derivatives of the image respectively. These gradients are usually estimated using the difference between adjacent pixels. As Equation 2.4 contains two unknown variables it cannot be

solved locally; additional constraints are required. This is referred to as the *aperture problem* because local motion information is ambiguous when viewed through a small window (or aperture).

Optical flow algorithms apply additional constraints to solve this problem, although these require certain assumptions which may or may not be valid in a given circumstance. Lucas and Kanade [117] assume that the disparity (u, v) between two images is constant across a region of interest R , and seek to minimise the L_2 norm measure of difference between the images:

$$\xi_{\text{LK}} = \sum_{(i,j) \in R} \xi_b^2 \quad (2.5)$$

$$= \sum_{(i,j) \in R} [I_i u + I_j v + I_t]^2 \quad (2.6)$$

A minimum value of ξ_{LK} satisfies $\frac{\partial \xi_{\text{LK}}}{\partial u} = 0$ and $\frac{\partial \xi_{\text{LK}}}{\partial v} = 0$, yielding a system of linear equations:

$$\begin{bmatrix} \sum I_i^2 & \sum I_i I_j \\ \sum I_i I_j & \sum I_j^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum I_i I_t \\ -\sum I_j I_t \end{bmatrix} \quad (2.7)$$

Solving this system provides a least squares estimate of the displacement (u, v) across the region R . Once this estimate is obtained, we can then move I by the estimate of (u, v) and repeat the estimation procedure in an iterative manner. The summations in Equation 2.7 are usually taken over a local region, $(i, j) \in R$, and for this reason the Lucas-Kanade method is referred to as a local method [28]. A dense flow field is not generated; rather, only a subset of image pixels are usually selected for optical flow calculation using this method.

Horn and Schunck [86] introduced a global *smoothness constraint* based on the

intuition that neighbouring points have a similar velocity. The smoothness constraint seeks to minimise the square magnitude of the *gradient* of the optical flow field, with respect to i, j . Thus a smoothness term is introduced:

$$\xi_c^2 = u_i^2 + u_j^2 + v_i^2 + v_j^2 \quad (2.8)$$

where the terms on the right hand side are derivatives with respect to their subscripts. The Horn and Schunck algorithm minimises a total error term, ξ_{HS} , across an image:

$$\xi_{\text{HS}} = \iint \xi_b^2 + \alpha^2 \xi_c^2 \, di \, dj \quad (2.9)$$

where α^2 denotes the relative weight of the smoothness term with respect to the data term. Larger values of α lead to smoother flow estimates. The value of the flow velocity (u, v) that minimises ξ_{HS} is found using calculus of variations, yielding two equations and two unknown values [86]. Because the values of (u, v) at each pixel are dependent on their neighbours, a direct solution cannot be efficiently computed. Instead, an iterative scheme is used to repeatedly update the estimates:

$$u^{n+1} = \bar{u}^n - \frac{I_i (I_i \bar{u}^n + I_j \bar{v}^n + I_t)}{\alpha^2 + I_i^2 + I_j^2} \quad (2.10)$$

$$v^{n+1} = \bar{v}^n - \frac{I_j (I_i \bar{u}^n + I_j \bar{v}^n + I_t)}{\alpha^2 + I_i^2 + I_j^2} \quad (2.11)$$

where \bar{u}^n and \bar{v}^n represent local averages in the neighbourhood [86], after the n th iteration step. The number of iterations will depend on the accuracy required, and the quality of the initial guess. For video sequences, the initial guess is typically the optical flow field from the previous frame.

The fundamental assumptions of the Horn and Schunck algorithm are that a flat, moving surface is being imaged under uniform illumination, and that there are no spatial discontinuities in the reflectance of the surface. This latter assumption particularly limits the applicability of the algorithm: “We exclude situations where objects occlude one another, in part, because discontinuities in reflectance are found at object boundaries.” [86]. These assumptions are not suitable for human crowds where pedestrians move in conflicting directions.

Black and Anandan [22] proposed the use of robust estimators to handle occlusions more effectively. They generalised the classical objective function ξ as,

$$\xi = \sum_{(i,j)} \rho_b(I_i u + I_j v + I_t) + \lambda [\rho_c(u_i) + \rho_c(u_j) + \rho_c(v_i) + \rho_c(v_j)] \quad (2.12)$$

where ρ_b and ρ_c denote penalty functions on the data and spatial terms respectively, and λ represents the relative weight given to the spatial terms compared to the data term. In the classical optical flow methods the quadratic penalty term, $\rho(x) = x^2$ was used (Equation 2.9). This penalty function can be highly susceptible to noisy outliers, therefore other functions have been used such as the Lorentzian [22]:

$$\rho(x) = \log\left(1 + \frac{x^2}{2\sigma^2}\right) \quad (2.13)$$

The curve is plotted in Figure 2.3. Larger error values, as displayed on the x -axis, are not penalised as heavily by the Lorentzian function.

The accuracy of various optical flow algorithms is difficult to assess, due to the lack of ground truth available to compare to the algorithm’s output. In the past, synthetic datasets such as the Yosemite sequence, as well as qualitative judge-

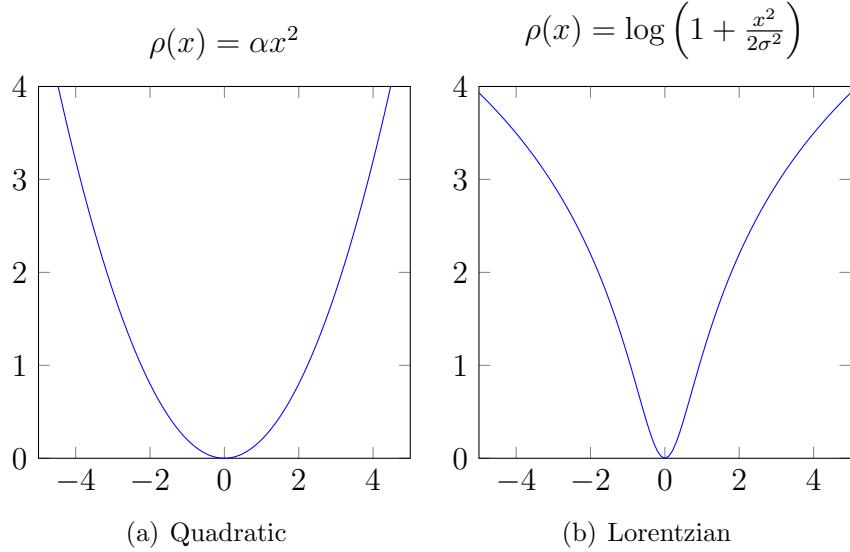


Figure 2.3: Quadratic and Lorentzian penalty functions.

ments, have been used to compare algorithms. The Middlebury database [16] was recently created by painting hidden fluorescent texture on a scene, and using high-resolution imagery to construct the ground truth. The database consists of four real-world sequences annotated in this manner, as well as three synthetic sequences and one stereo sequence. Each sequence is 8 frames in length. Publications commonly report the performance of new algorithms on this database, for example Sun [180] compares multiple methods. The full rankings are published online [16] for an open comparison of optical flow algorithms.

In practice, the performance of high-ranking algorithms on the Middlebury dataset may not necessarily be the best choice for crowd monitoring or other computer vision applications. When applied to an object detection task, for example, Geiger [76] concludes:

Results from state-of-the-art algorithms reveal that methods ranking high on established datasets such as Middlebury perform below average when being moved outside the laboratory to the real world.

Algorithms which are optimised for performance on synthetic and controlled sequences may not be as generally applicable as other methods. Additionally, state of the art algorithms are often very computationally expensive, requiring many seconds or even minutes to process each frame. This is not suitable for real-time surveillance applications.

Improving the performance of established optical flow algorithms is beyond the scope of this thesis, as discussed in Section 1.3. Therefore the research presented in this thesis uses well-established, broadly-applicable and computationally efficient algorithms such as Horn and Schunck [86] and Black and Anandan [22]. Future improvements in optical flow technology and computational efficiency can be incorporated into the algorithms discussed in this thesis by updating the relevant optical flow modules.

2.2.2 Adaptive Background Models and Foreground Detection

While optical flow is an estimate of the motion occurring at each pixel, it is not strictly a segmentation process. The results obtained from such methods are in some cases too detailed, inaccurate in the presence of complex occlusions and variable light sources, and computationally expensive, for real-time surveillance purposes. More commonly, a motion detection algorithm is employed which models the background of a scene in a probabilistic manner, in order to detect anomalies belonging to foreground objects.

Stauffer and Grimson [178, 179] proposed a background subtraction algorithm which has become one of the most popular methods in the literature. The technique models each pixel as a mixture of Gaussian distributions and uses an online

K -means approximation to the Expectation-Maximisation algorithm to update the model. The sequence of values observed at each pixel, $\{\mathbf{x}_t\}$, is considered to be a non-stationary process whose recent history is modelled as a mixture of K (3 to 5) Gaussian distributions. The Gaussian distribution is denoted:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{D}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right) \quad (2.14)$$

where $\boldsymbol{\mu}$ denotes the *mean* value of the distribution and $\boldsymbol{\Sigma}$ denotes the covariance matrix. The dimensionality of \mathbf{x}_t is denoted D , and is typically equal to 1 for greyscale images or 3 for colour images. For an incrementally updated Gaussian Mixture Model (GMM), each Gaussian mixture i at time t is weighted by a coefficient, $\omega_{i,t}$, so that the probability of observing pixel \mathbf{x}_t is:

$$p(\mathbf{x}_t) = \sum_{i=1}^K \omega_{i,t} \mathcal{N}(\mathbf{x}_t | \boldsymbol{\mu}_{i,t}, \boldsymbol{\Sigma}_{i,t}) \quad (2.15)$$

For computational simplicity, a diagonal covariance matrix is used: $\boldsymbol{\Sigma}_{i,t} = \sigma_{i,t}^2 \mathbf{I}$. A pixel, \mathbf{x}_t , is matched to distribution k if its value lies within $d = 2.5$ standard deviations of the k th distribution's mean. If no match is found, the new value replaces the least probable distribution with a new distribution. The weights are updated with learning rate α :

$$\omega_{k,t} = \begin{cases} (1 - \alpha)\omega_{k,t} + \alpha & \text{if pixel } \mathbf{x}_t \text{ belongs to distribution } k \\ (1 - \alpha)\omega_{k,t} & \text{otherwise} \end{cases} \quad (2.16)$$

and then renormalised. The parameters for the matching distribution are also updated to incorporate \mathbf{x}_t :

$$\boldsymbol{\mu}_{k,t} = (1 - \rho)\boldsymbol{\mu}_{k,t-1} + \rho\mathbf{x}_t \quad (2.17)$$

$$\sigma_{k,t}^2 = (1 - \rho)\sigma_{k,t-1}^2 + \rho(\mathbf{x}_t - \boldsymbol{\mu}_{k,t})^T(\mathbf{x}_t - \boldsymbol{\mu}_{k,t}) \quad (2.18)$$

where

$$\rho = \alpha \mathcal{N}(\mathbf{x}_t, \boldsymbol{\mu}_{k,t-1}, \boldsymbol{\Sigma}_{k,t-1}) \quad (2.19)$$

The mixtures are ordered by the ratio $\frac{\omega}{\sigma}$, and the first B distributions in the ordered list are assumed to belong to the background, where:

$$B = \operatorname{argmin}_b \sum_{k=1}^b \omega_k > T \quad (2.20)$$

The threshold T denotes the minimum fraction of the model's data that should be accounted for by background. Pixels that do not belong to these mixtures are classified as foreground pixels.

Zivkovic [207, 209] extended Stauffer's GMM-based background subtraction algorithm, which included a model selection criterion to choose the appropriate number of components for each pixel. A full covariance matrix is used to improve generality. An efficient implementation is available online [208].

A background model similar to the GMM approach was proposed by Butler [30, 31] and Denman [60] in which each pixel is represented by a group of discrete 'clusters', also ordered by weight. Incoming pixels are matched to a cluster using Manhattan distance and a fixed threshold (rather than Euclidean distance and d standard deviations of each distribution). The purpose of this approach is to minimise computation time for real-time operation. Weights are updated and

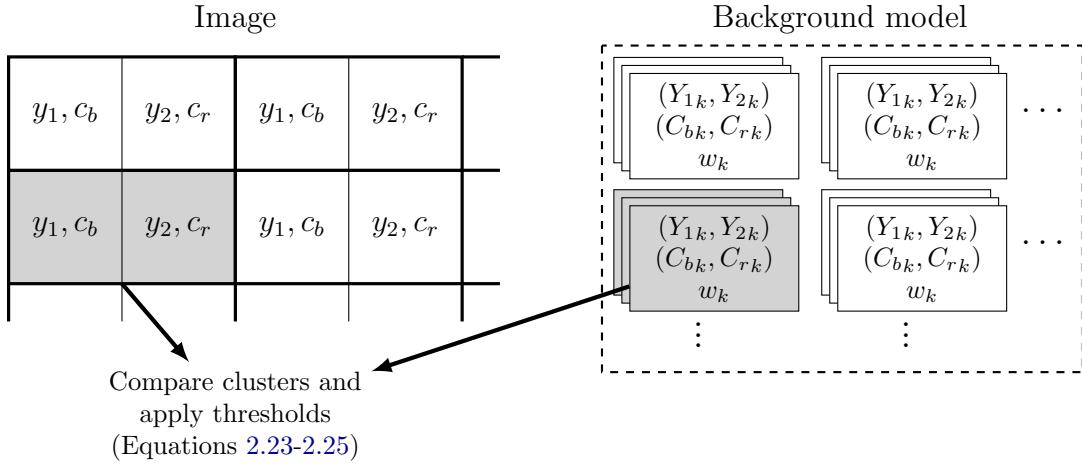


Figure 2.4: Pixel pairs are grouped into clusters. The adaptive background model [61] contains a group of clusters, each assigned a weight w_k indicating its likelihood, against which incoming clusters are compared.

renormalised in a similar manner to Stauffer-Grimson, while the cluster values themselves are integers (which are incremented or decremented when sufficient error accumulates in either direction).

This background segmentation routine operates in the YCbCr 4:2:2 colour space, which provides some invariance to lighting changes through the separation of colour and intensity. Each pixel in the incoming image, I , has two values: a luminance and a single chrominance, which alternates between blue chrominance and red chrominance in the horizontal direction (Figure 2.4). Pixels are paired horizontally so that for each pair there are four values (two luminance, one blue chrominance and one red chrominance):

$$P(i, j, t) = [y_1, y_2, c_b, c_r] \quad (2.21)$$

where $P(i, j, t)$ denotes a pixel pair, or ‘cluster’, formed by grouping the two pixels, $I(2i, j, t)$ and $I(2i + 1, j, t)$. This pairing results in motion detection being effectively performed at half the horizontal resolution of the original image, with

the benefit being increased speed.

A multi-modal background model is then constructed, for each pixel pair, by storing a set of possible modes representing the distribution of colours at that location (Figure 2.4). These are stored as a group of clusters, each accompanied by a weight, w_k , where k is used to denote the mode. The weight describes the likelihood of the colour described by that cluster being observed at that position in the image. Each cluster in the background model is represented by:

$$C(i, j, t, k) = [Y_{1k}, Y_{2k}, C_{bk}, C_{rk}, w_k] \quad (2.22)$$

Clusters in the background model are stored in order of highest to lowest weight. Incoming clusters, $P(i, j, t)$, are compared to all possible modes, $C(i, j, t, k)$, to determine a match. A match is found by finding the highest-weighted mode which satisfies:

$$|Y_{1k} - y_1| + |Y_{2k} - y_2| < T_{Lum} \quad (2.23)$$

$$|C_{bk} - c_b| + |C_{rk} - c_r| < T_{Chr} \quad (2.24)$$

where T_{Lum} and T_{Chr} denote the luminance and chrominance thresholds, respectively. Foreground motion is detected if the probability of the matching mode, m , falls below a threshold, T_{fg} :

$$F(i, j, t) = \begin{cases} 1 & \text{if } \sum_{k=0}^m w_k < T_{fg} \\ 0 & \text{otherwise} \end{cases} \quad (2.25)$$

The matching cluster in the background model is adjusted to reflect the current pixel colour, and the weights of all clusters in the model at this location are adjusted and normalised to reflect the new state [59]. If no match is found, then the

lowest weighted cluster is replaced with a new cluster representing the incoming pixels (and foreground is detected at this location). Clusters are gradually adjusted and removed as required, allowing the system to adapt to slow changes in the background.

Denman [61] also proposed a lighting model which adjusts the thresholds across the scene. In surveillance situations, particularly outdoor scenarios, lighting levels can also change rapidly resulting in large amounts of erroneous motion. When these levels fluctuate, it is the luminance values in an image which undergo significant change, whereas chrominance values remain relatively unchanged. Therefore, to improve performance in real world lighting conditions, Denman adjusted the luminance threshold T_{Lum} . As the luminance change is not constant across a scene, images are divided into several small regions (typically a 5×5 grid of sub-regions), and each is treated separately. Thresholds for detection are varied within each region, according to the lighting conditions in that part of the scene. We define the luminance difference, Δ_{Lum} , at a cluster to be:

$$\Delta_{Lum} = |Y_{1m} - y_1| + |Y_{2m} - y_2| \quad (2.26)$$

where Y_{1m} and Y_{2m} denote the luminance values of the matching mode m , and y_1 and y_2 are the luminance values of the incoming cluster. Attaching coordinate and time information, we use $\Delta_{Lum}(i, j, t)$ to represent the luminance difference at a specific frame and location. Thus the weighted average of luminance changes is calculated across an image region, R :

$$O_{Lum}(R, t) = \frac{\sum_{(i,j) \in R} \Delta_{Lum}(i, j, t) \times w_m(i, j, t)}{\sum_{(i,j) \in R} w_m(i, j, t)} \quad (2.27)$$

The use of weighted sum allows pixels that are only recently created, potentially

under the present lighting conditions, to be weighted less relative to those that have been present longer. An acceptable range for the luminance offset at time t , with respect to the previous frame ($t - 1$), is defined:

$$\chi \leq \frac{O_{Lum}(R, t)}{O_{Lum}(R, t - 1)} \leq \frac{1}{\chi} \quad (2.28)$$

where $\chi \in [0, 1]$ is the change threshold for the luminance offset. If the change in luminance offset falls outside of this acceptable range, a rapid fluctuation in luminance has been detected across the region, and Equation 2.23 is modified as follows:

$$|Y_{1k} - y_1| + |Y_{2k} - y_2| < T_{Lum} + O_{Lum} \quad (2.29)$$

where O_{Lum} is the luminance offset of the region to which the cluster being matched belongs. Loosening the threshold enables improved performance when dealing with both global lighting changes (such as changes in camera gain), or local changes such as variable cloud cover. This approach is robust in various environments, including both indoor and outdoor scenes. The full details of Denman's background model are presented in [59, 60, 61].

Following foreground detection, a morphological closing operation is commonly applied to the binary mask in order to obtain ‘cleaner’ and less fragmented blob segments. The foreground pixels are then grouped into segments or ‘blobs’ using a connected components algorithm.

Recent advancements in background modelling have attempted to incorporate periodic elements into the model, such as escalator motion, flashing warning lights and scrolling advertisements [111]. These are modeled using a Markov process,

which must be initialised from a training sequence.

As with optical flow, the focus of this thesis is not on improving the performance of established foreground detection algorithms (see Section 1.3). Therefore the research presented in this thesis uses popular existing algorithms such as Stauffer-Grimson [178], Zivkovic [207, 209] and Denman [59, 60, 61]. Future improvements in background modelling technology can be incorporated into the algorithms discussed in this thesis by replacing the relevant modules.

2.3 Crowd Counting

The task of crowd counting has been approached from a number of angles. Although the specific techniques differ, these algorithms do share some common elements. They typically involve image feature extraction followed by a classification or regression stage. The first step belongs to the ‘Feature extraction’ module described in Section 2.1.2, while the second is an example of ‘High level understanding’ (Section 2.1.3). The quantity and complexity of the features extracted has a marked impact on the classification stage; specifically, on the complexity of the classifier, on the computation time required, and on the required size of the training data set. As each new feature introduces an additional level of dimensionality (and therefore necessitates a wider set of training data), it is important to use features which correspond accurately to the level of crowding within a scene.

Optical flow, for example, may be considered as a measure of activity and movement within a scene. Computed using algorithms such as Horn and Schunck [86] and Lucas-Kanade [117], an optical flow field is a map of individual pixel velocities between one frame and the next (Section 2.2.1). Although this appears to be

a useful indicator of crowd *motion*, further consideration suggests that velocity is a poor feature for estimating crowd *size*, as optical flow is speed-dependent. Therefore fast-moving pedestrians may contribute substantially to the total velocity present in a scene, leading to over-estimation of crowd size. Conversely, congested scenes might not be detected due to the relative lack of motion caused by over-crowding, which, crucially, is one of the abnormal situations we may seek to detect. Furthermore, the level of detail provided by optical flow is *too specific* for the required task: a floating point value for each pixel in an image is overly precise, and not required. Additionally, it is computationally expensive to obtain such estimates.

By contrast, background subtraction has been used widely [33, 49, 58, 97, 103, 121, 131, 144, 164] to extract a boolean mask of *foreground pixels*, corresponding to objects (such as humans) in a scene, independent of speed. The relationship between the number of foreground pixels and the crowd size has been modelled with varying degrees of complexity, from simple linear fitting [58, 103] to neural network classifiers [49, 103] and Gaussian process regression [36].

A number of other features, such as *textural* information, have been used in conjunction with machine learning techniques to measure crowd density [128, 129, 153, 157, 194]. The term “density” is used in those contexts where no attempt is made to *count* the crowds in question. Rather, a scene may be simply classified as either ‘very low’, ‘low’, ‘moderate’, ‘high’ or ‘very high’ density.

Regardless of the specific features extracted, or the classifier used, the end result is a measure of crowding. This is a holistic description of the scene, and the sole measure of accuracy is how closely the system’s estimate matches the real value. The real value is referred to as the ground truth. Therefore it is logical to design a system based upon those kinds of *holistic* features which are indicative of larger crowds. Fewer attempts have been made to utilise local features which are

specific to individuals within the scene, such as head detection [113]. As stated by Davies [58],

Our objective for the models is that they should not involve actual counting of individuals or tracking of the movements of individuals but should be based on a collective description of crowds (e.g. analogous to the ideal-gas theory which ignores individual molecules).

As with optical flow, specific localised features may have been perceived as too detailed, computationally expensive, or prone to inaccuracy. Therefore holistic features have dominated the research into crowd counting thus far. The following literature review discusses crowd counting techniques in order from high-level (holistic) to low-level (local) features:

1. **Holistic approaches** utilise global image features to obtain an estimate. These can also be described as “mapping-based” approaches, because they map directly between the feature space and the crowd size estimate. This works best for large crowds, in which the analogy to the ideal-gas theory holds. Holistic approaches are discussed in Section 2.3.1.
2. **Local approaches** utilise local image features. These may sometimes be described as “detection-based” approaches, because they detect, track or otherwise classify pedestrians on an individual or group level. Local approaches are discussed in Section 2.3.2.

A summary of the existing research and a discussion of limitations is presented in Section 2.3.3.

Author	Year	Perspective Normalisation	Image Features	Regression Model	Comment
Regazzoni [157]	1993		Edge features	EKF/BBN	
Davies [58]	1995		Foreground pixels (static background)	Linear	
Marana [126, 127, 128, 130]	1997		Textural statistics (GLCM)	NN	Density class only.
Marana [129]	1999		Minkowski fractal dimension	NN	Density class only.
Paragios [144]	2001	✓	Foreground pixels	Linear	Crowding ratio only.
Huang [93]	2002		Foreground pixels	NN	
Ma [121]	2004	✓	Foreground pixels	Linear	
Kong [103, 104]	2005	✓	Blob size histogram, edge histogram	Linear/NN	
Rahmalan [153]	2006		TIOCM	NN	
Xiaohua [197]	2006		Textural statistics (DWT)	SVM tree	Density class only.
Hou [87, 88]	2008	✓	Foreground pixels	NN	
Chan [34, 35, 36, 38]	2008	✓	Textures, fractal dimension, edge features, foreground (bidirectional DTs), perimeter features	GPR	Count each direction.

Table 2.1: High level summary of **holistic** crowd counting systems. See the main text in Section 2.3.1 for a full description of these algorithms and acronyms. The reader may also refer to the Acronyms section on page [xxxvii](#).

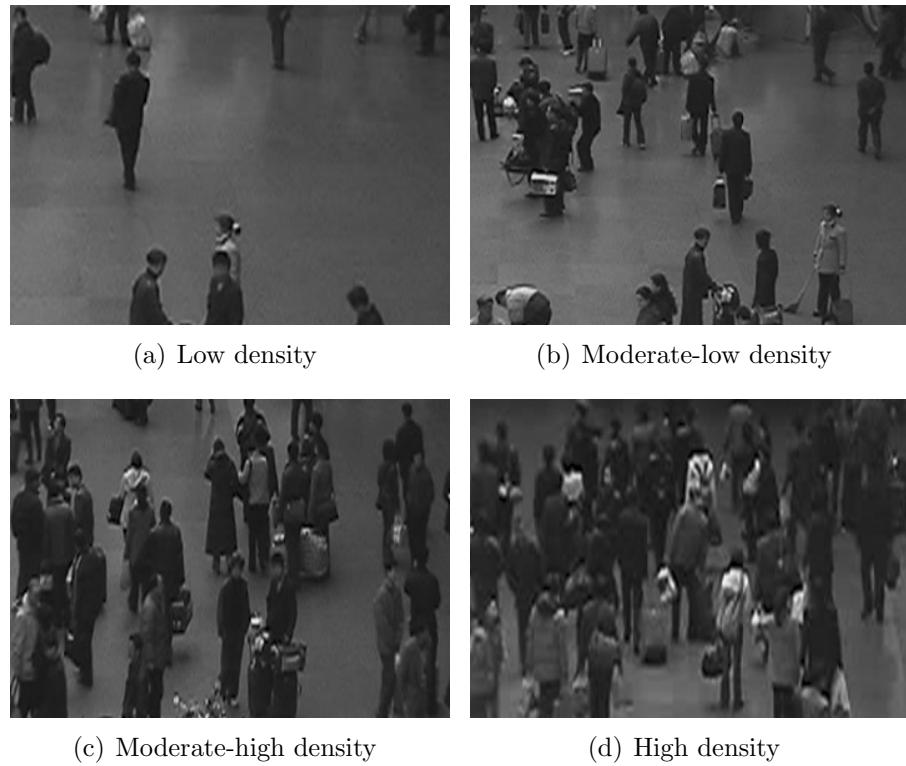


Figure 2.5: Different levels of crowding density. Images from Xiaohua [197].

2.3.1 Holistic Approaches

Holistic crowd counting algorithms use global image features to estimate the size of a crowd. They may also be described as “mapping-based” approaches because they map directly between the feature space and the crowd size estimate. Features used by these systems include textures [128], foreground pixels [58] and gradient features [103], amongst others, while the classification and regression strategies have included linear regression [58] and neural networks [103, 128]. The methods discussed in this section are summarised briefly in Table 2.1.

An early system developed by Marana [126, 128] used holistic image features for crowd density estimation, derived from the textures present in the image. As depicted in Figure 2.5, the textures in images of low density crowds are coarse

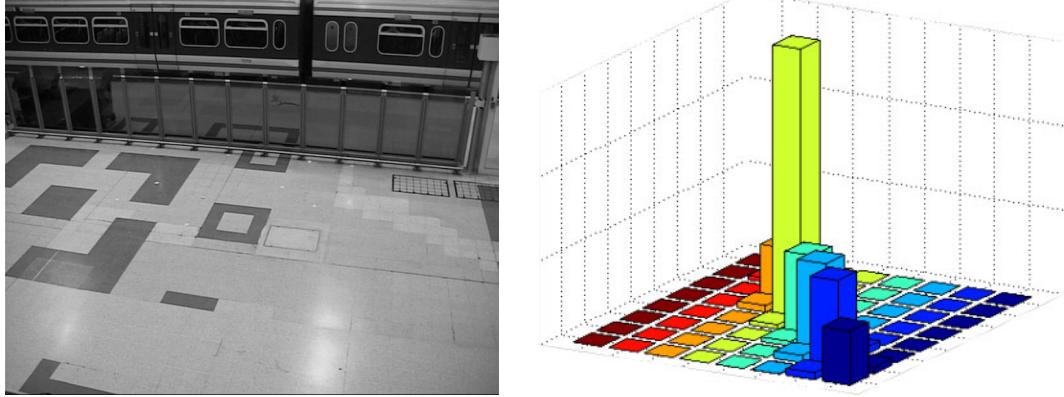
whereas those in high density crowds are fine. For this algorithm, it is assumed that the scene's background is relatively smooth compared to the human textures in the foreground. When considered with this assumption in mind, textures are helpful because the introduction of human crowding will disrupt those textures of the background. While textural information will be altered by the presence of crowding, it does not explicitly segment the foreground from the background.

The textural features used by Marana were first proposed by Haralick [79]. These textural statistics are derived from the Grey Level Cooccurrence Matrix (GLCM), also known as the Grey Level Dependency Matrix (GLDM). Given an image I , whose grey levels are quantized into N levels, the GLCM, G , is an $N \times N$ matrix specifying the quantity of co-occurring pixel values at a given offset $\delta = (\delta_i, \delta_j)$:

$$G(r, c) = \sum_{(i,j) \in I} \begin{cases} 1 & \text{if } I(i, j) = r \text{ and } I(i + \delta_i, j + \delta_j) = c \\ 0 & \text{otherwise} \end{cases} \quad (2.30)$$

That is, element $G(r, c)$ specifies the frequency with which a pixel of grey level r appears at an offset of (δ_i, δ_j) from a pixel of grey level c . The offset δ is adjusted to compute the GLCM horizontally (0°) when $\delta = (1, 0)$; vertically (90°) when $\delta = (0, 1)$; or diagonally (45° and 135°) when $\delta = (1, 1)$ and $\delta = (-1, 1)$. For each angle, a different GLCM is calculated.

A three dimensional illustration of the GLCM is depicted in Figure 2.6. It can be seen here that the GLCM is a histogram of grey level cooccurrences, and that for a low frequency image such as the background in Figure 2.6(a), the histogram bins are greatest along the diagonal because grey levels of equal value frequently occur beside one another. When normalized (by dividing by the number of pixels in the image) the GLCM, G , becomes a second-order joint conditional probability density function, f , such that $f(r, c)$ is the *probability* of the grey levels r and c



(a) Background image from the PETS 2006 database [148].
(b) Three dimensional representation of the corresponding GLCM with $\delta = (1, 0)$, after the image is quantized to $N = 8$ grey levels.

Figure 2.6: Three dimensional illustration of the GLCM.

occurring beside one another (at offset δ). This is calculated by:

$$f(r, c) = \frac{G(r, c)}{\sum_{r=1}^N \sum_{c=1}^N G(r, c)} \quad (2.31)$$

From each of the four normalized Grey Level Cooccurrence Matrices ($0^\circ, 45^\circ, 90^\circ, 135^\circ$), holistic textural properties may be calculated. Those proposed by Haralick [79] include, among others, contrast, homogeneity, energy and entropy:

$$\text{Contrast} = \sum_{r=0}^{N-1} \sum_{c=0}^{N-1} (r - c)^2 f(r, c) \quad (2.32)$$

$$\text{Homogeneity} = \sum_{r=0}^{N-1} \sum_{c=0}^{N-1} \frac{f(r, c)}{1 + (r - c)^2} \quad (2.33)$$

$$\text{Energy} = \sum_{r=0}^{N-1} \sum_{c=0}^{N-1} f(r, c)^2 \quad (2.34)$$

$$\text{Entropy} = - \sum_{r=0}^{N-1} \sum_{c=0}^{N-1} f(r, c) \log f(r, c) \quad (2.35)$$

Marana [128] uses these four properties, for *each* of the four Grey Level Cooccurrence Matrices, providing a total of sixteen holistic textual features describing an image. The mapping from feature space to crowd density was performed using Kohonen’s self organising map (SOM) neural network [102]. Five levels of crowd density were considered, from ‘very low’ to ‘very high’ density. Trained on 151 images taken at a railway station, a correct classification rate of 81.88% was reported on a test set of 149 images.

Other texture-like holistic features include Minkowski fractal dimension [129], wavelet transform features [197] and Translation Invariant Orthonormal Chebyshev Moments [153]. These approaches are discussed below.

The Minkowski fractal dimension is calculated by first performing edge detection on an image, resulting in a binary image, to which morphological dilation is applied using a circular structuring element of various sizes. These dilations are shown in Figure 2.7. The log-log plot of disk size vs number of pixels after dilation is shown in Figure 2.7(e). The fractal dimension is:

$$D = 2 - m \quad (2.36)$$

where m denotes the slope of the linear regression line. The presence of edges in an image is assumed to be indicative of human crowding, with a greater density of edge points increasing the fractal dimension D . A neural network classifier was used to differentiate between images of different class densities on a five point scale [129].

As with the use of textural features, this approach assumes that the background is relatively smooth and free from significant edges. Consider a scene where these assumptions are violated, such as that shown in Figure 2.8. The background



(a) Images from PETS 2009 [149]

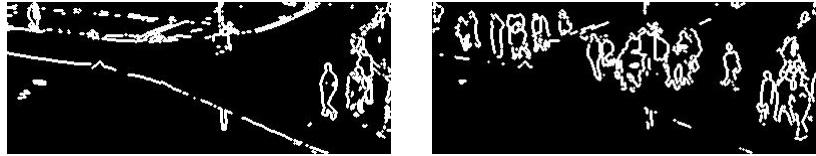
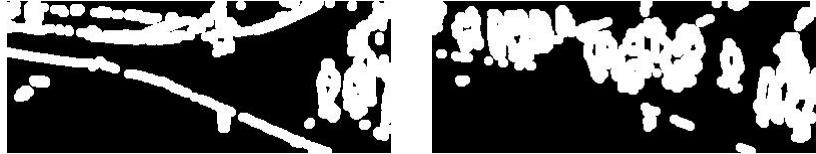
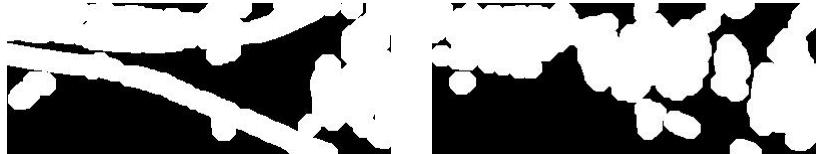
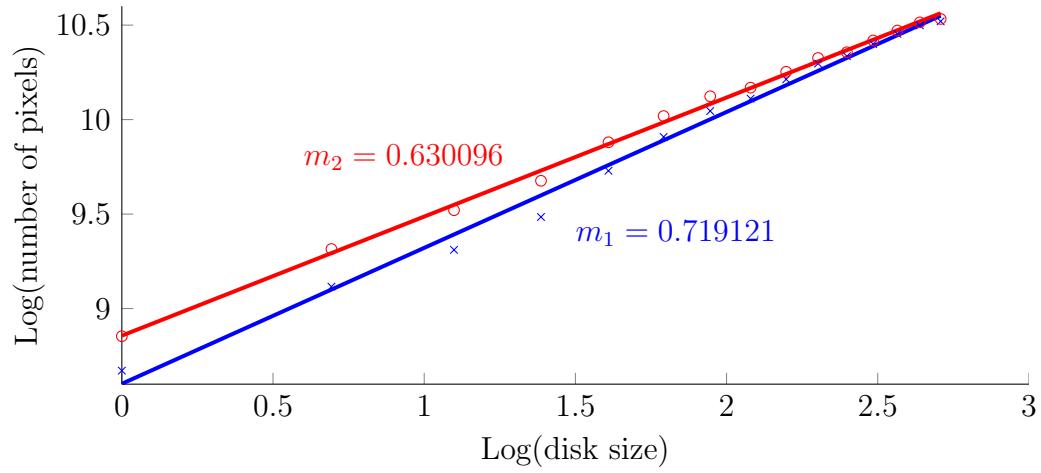
(b) Edge detection followed by dilation with disk ($r = 1$)(c) Edge detection followed by dilation with disk ($r = 5$)(d) Edge detection followed by dilation with disk ($r = 11$)(e) Log-log plot of disk size vs number of pixels after dilation. The fractal dimension is derived from the gradient of the linear regression lines, m_1 and m_2 (Equation 2.36). These lines correspond to the two different images.

Figure 2.7: Minkowski fractal dimension is calculated using edge detection followed by dilation using successively larger structuring elements.

of this scene contains numerous objects such as stalls, seats, plants and floor patterns. Here the density of edge pixels across the image is relatively similar regardless of crowd size (13 vs 53, including partially visible pedestrians). Consequently the slopes of the linear regression lines in Figure 2.8(d) are almost identical. Therefore the fractal dimension is unlikely to provide a robust measurement of crowd size across a wide range of varying conditions. Indeed, Marana [129] found that the Minkowski fractal dimension did not perform as well as the GLCM-based textural features described above.

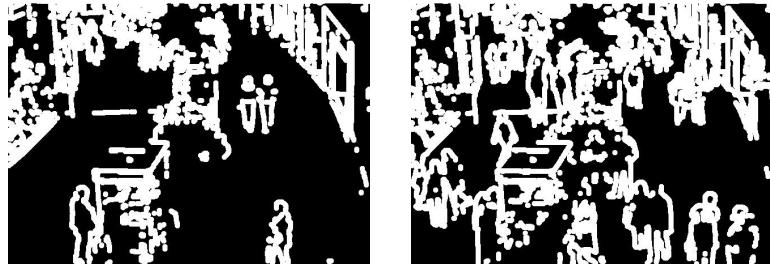
Xiaohua [197] proposed the use of the 2D discrete wavelet transform (DWT) as a basis for extracting textural features. An image is first transformed into a multi-scale format using the wavelet transform, and at each scale the first order and second order statistical features are calculated. A tree structure classifier was used to classify the crowd into four density categories, as shown in Figure 2.9. The tree is comprised of three SVM binary classifiers which divide incoming images into higher or lower density categories. Equal performance was seen when comparing the first order wavelet features to the GLCM features, with 89.3% correct classification. When all textural features were combined into a larger descriptor, an improved classification rate of 95.3% was observed.

Some of the images used by Xiaohua are shown in Figure 2.5. As with Marana's algorithms, Xiaohua's approach is based on the observation that high density crowds "are made up of fine (high frequency) texture patterns" whereas low density crowds generally have "coarse (low frequency) texture patterns" [197]. These assumptions are not guaranteed to be true in all environments, such as the Mall dataset shown in Figure 2.8.

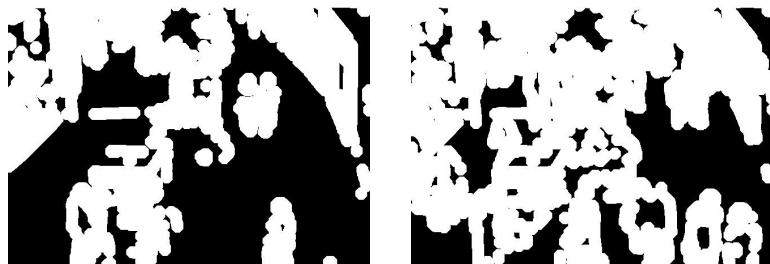
Rahmalan [153, 154] proposed the use of Translation Invariant Orthonormal Chebyshev Moments (TIOCM) to classify crowd density. Given an image I with



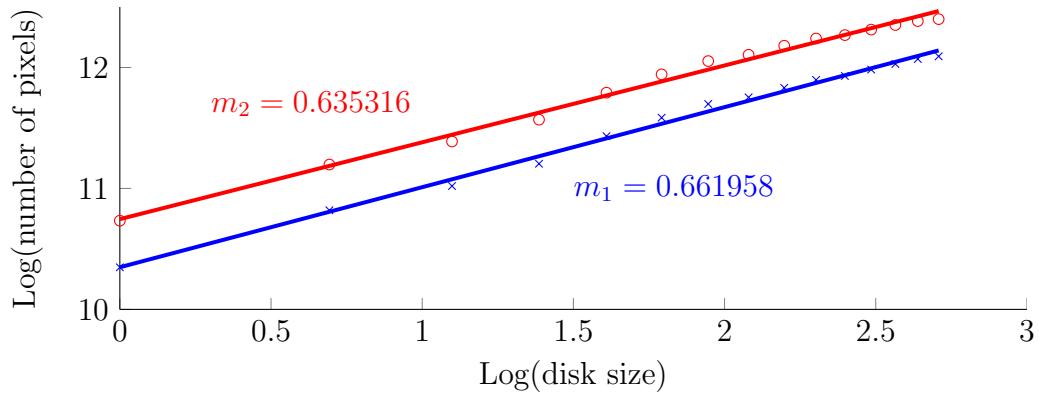
(a) Images from the Mall dataset [42], with crowds of size 13 and 53 (including partially visible pedestrians).



(b) Edge detection followed by dilation with disk ($r = 5$)



(c) Edge detection followed by dilation with disk ($r = 11$)



(d) Log-log plot of disk size vs number of pixels after dilation. The slopes m_1 and m_2 are very similar. These regression lines correspond to the two different images.

Figure 2.8: Minkowski fractal dimension may not be reliable for crowd counting when the background scene contains strong gradients and effects of perspective.

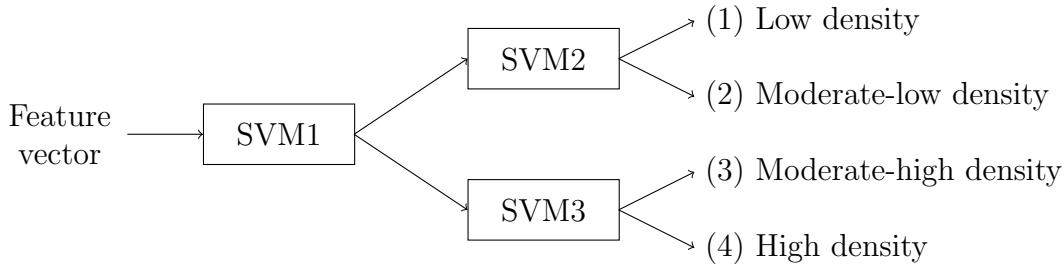


Figure 2.9: Tree structure classifier used by Xiaohua [197], comprised of binary SVM classifiers.

centroid (i_c, j_c) , the TIOCM of order $m + n$ is defined by:

$$C_{mn} = \sum_i \sum_j \hat{t}_m(i - i_c) \hat{t}_n(j - j_c) I(i, j) \quad (2.37)$$

where $\hat{t}_m(\cdot), \hat{t}_n(\cdot)$ are the scaled Chebyshev polynomials, and are defined by the recurrence relation:

$$\hat{t}_m(q) = \alpha_1 q \hat{t}_{m-1}(q) + \alpha_2 \hat{t}_{m-1}(q) + \alpha_3 \hat{t}_{m-2}(q) \quad (2.38)$$

whose polynomial coefficients $\alpha_1, \alpha_2, \alpha_3$ and initial conditions are defined in [153]. The six features used by Rahmalan for crowd density analysis are TIOCM moments of order $(m, n) = (0, 0), (0, 1), (1, 0), (2, 0), (0, 2), (1, 1)$. Rahmalan likens the behaviour of these features to texture:

In this study, moments can also be viewed as a texture descriptor. For example, the zero order moments can be seen to be like the area of the target object. A small area of edge points can be viewed as a coarse texture which represents low crowd density while a huge quantity of edge points can be viewed as a fine texture which indicates a high crowd density.

An SOM neural network classifier is used to distinguish between five categories of crowd density, adhering to the same protocol used by Marana [128, 129]. Rahmalan uses two crowd datasets captured at an outdoor reception: one set is recorded in the morning, and the other in the afternoon. Three feature types were compared: GLCM-based textures, Minkowski fractal dimension and Chebyshev moments (TIOCM). In all cases the TIOCM features outperformed the GLCM and Minkowski features, with a correct classification rate of 70% and 90% for the morning and afternoon datasets respectively. All algorithms performed better on the afternoon dataset, “because the afternoon data has smaller variation of illumination when compared with morning data”. When the datasets were combined to form a larger mixed set, performance of the TIOCM and Minkowski features decreased compared to the afternoon dataset alone, again due to the illumination changes over time.

These results indicate that holistic textural features are sensitive to external changes such as lighting changes and alteration of the scene’s background. Results presented in Section 3.2.3.1 of this thesis (p. 116) provide additional support for this conclusion. Consequently a crowd monitoring system designed for long-term usage would have to be re-trained after any significant changes in the environment took place. For a complex environment, it may be desirable to position a large number of cameras throughout the scene; using holistic textural features would require that each camera be trained independently, perhaps on a hundred frames or more. This would not be a practical solution for long term crowd management over a wide range of conditions and backgrounds.

Holistic textural methods have also been confined to crowd density estimation on a four or five point scale, rather than explicitly counting or estimating the number of people in a scene. There is also ambiguity when the density of people in the scene lies near the border of two neighbouring classes (e.g. ‘very low’

and ‘moderately low’ density classes can be similar). Xiaohua [197] describes the errors encountered in their experiments as follows:

The errors mainly occurred between the neighboring classes. Such errors are excusable since there isn't usually distinct boundary between neighboring two classes.

This lack of a distinct boundary causes the estimates to be inherently imprecise. Although this may be suitable for some applications, it is generally more desirable to have an exact estimate of the number of people visible in a scene. If an approximate density classification is all that is required (for example, using a five point scale), then the crowd size estimate can be used *directly* to perform this as a subsequent step after counting has been performed (e.g. by designing tiers based on pre-defined crowd ranges).

In contrast to those systems which use textural features, more recent crowd counting algorithms have utilised features which are specifically indicative of human (or foreground) crowding. While these features are still considered on a holistic level (for example, the total number of foreground pixels or vertical edges detected in an image), the features themselves are located at points of interest prior to their aggregation.

Regazzoni [157, 158] proposed one of the earliest crowd size monitoring systems which could be found in the available scientific literature. Due to computational constraints, the features were selected based on CPU computation time as well as estimation accuracy. The features used by Regazzoni were:

1. Number of edge points.
2. Vertical edges.

3. The sum of amplitudes of the maxima detected in the histogram shape of edge points.

The vertically oriented edges are used because these are considered “really important for detecting the bodies (i.e. legs and arms)”. Additionally, the histogram maxima are caused primarily by vertical edges [157]. The crowding estimation is based on the Extended Kalman Filter (EKF), which is compared to the results obtained from a Bayesian Belief Network (BBN). The authors found that the EKF provided a smoother and more accurate estimate than the BBN baseline, due to its employment of temporal information. This is in contrast to other methodologies (such as textural methods) that estimate crowding in single images only; in those approaches, “information coming from temporal correlation is not explicitly used inside the system.”

As computing power increased, a number of algorithms have attempted to segment the foreground using background modelling techniques. By accumulating background statistics over time, temporal information is implicitly incorporated into the system. These approaches comprise the majority of holistic crowd counting algorithms in use today. The rationale for this approach is described by Cho [48]:

It is clear that a human observer has absolutely no problem in distinguishing a very dense crowd from the background. It is believed that human brain is well trained and would be likely to use the ratio of “crowd area” to “background area” as an estimate for the crowd density. This idea could be applied quantitatively to computer-based density estimation if the image-pixels corresponding to the crowd could be separated from those of the background.



(a) An image after background subtraction used by Davies [58].



(b) Frame 1280 of the UCSD crowd counting database [36].



(c) Foreground detection on frame 1280 of the UCSD database, using an adaptive background model [59, 60].



(d) Region of interest for UCSD database.

Figure 2.10: Foreground detection on two scenes.

Cho [48, 49, 50], Davies [58] and Huang [93] utilised a static “reference image” of the background of the scene in which crowd levels were to be monitored. The reference image, I_{ref} , was subtracted from each frame, I , before applying a threshold T_{fg} to extract a foreground boolean mask, F :

$$F(i, j) = \begin{cases} 1 & \text{if } |I(i, j) - I_{ref}(i, j)| < T_{fg} \\ 0 & \text{otherwise} \end{cases} \quad (2.39)$$

This would yield a binary foreground image similar to those shown in Figure 2.10. Davies [58] found that the relationship between the number of people in the scene and the total number of foreground pixels was approximately linear, and that this relationship also held for edge pixels (although the correlation was not as strong). The crowd estimate was therefore obtained by using linear regression to model the relationship between foreground pixels and crowd size, and filtering the output over time with a Kalman filter. The mean relative error of this approach was less than 8% when applied to railway station footage. Cho [48, 49, 50] also used edge and foreground pixel counts and proposed a fast training algorithm for feedforward neural networks. Huang [93] calculated the percentage of foreground pixels in each sub-region of the image, and these values were used to populate a feature vector which served as inputs to a neural network for regression.

For the purposes of indoor crowd estimation over a short period of time, these approach are successful. However, the use of a static background image means that the system is sensitive to lighting changes over longer periods of time, whether sudden or gradual. Adaptive background models such as Stauffer-Grimson [178], Zivkovic [207, 209] and Denman [59, 60, 61] (see Section 2.2.2) are robust against such changes, and have been adopted in more recent crowd counting applications.

The analyses of Davies [58], Cho [48, 49, 50] and Huang [93] were based on scenes with a relatively high camera angle (e.g. Figure 7.8(a)), in which the effects of perspective were not apparent. When perspective distortion is significant, as seen in Figures 2.8(a) and 7.8(b), the total number of foreground pixels is less likely to be a reliable indicator of crowding, because objects in the distance appear smaller and therefore contribute fewer pixels to the foreground mask. The effects of perspective are also problematic for those algorithms that employ texture, fractal dimension and image moments (Marana [128, 129], Rahmalan [153] and Xiaohua [197], respectively).

Paragios [144] introduced the use of a density estimator to account for perspective in an image. Given a railway setting, the authors observe:

[T]he platform is planar and we can obtain quasi-calibration information by using the images of the trains. This quasi-calibration information that is sought is about the relative size variation of the projection height and widths of a rigid object as the object translates in depth. This size variation is a function of the row and column coordinates.

Therefore a pair of functions, $H(r)$ and $V(r)$, are employed to describe the relative scale of widths and heights of a “unit box in the world projected to a given row r in the image”. Consequently, the relative scale of an object at an image location (i, j) is represented by the product, $G(i, j) = H(j) \times V(j)$, and each foreground pixel is weighted by the inverse of this scale to compensate for perspective. The weighted sum of foreground pixels is used by Paragios to obtain a “crowdedness measure,” C , between 0 (representing an empty scene) and 1.0 (fully occupied).

Ma [121] also considered perspective in a similar manner, by computing a ‘density map’ weighting each pixel according to the area it represents on the ground plane. It was calculated based on the coordinates of four points in an image, corresponding to two parallel lines in the real world, such as the edges of a roadway. The camera is assumed to be oriented horizontally, so that the weight assigned to every pixel in any given row of the image is equal. A *reference row* is selected in the image, to which a density of 1.0 is assigned, and all other rows in the image are scaled with respect to this. The approach is described in greater detail in Section 3.2.2. The system described by Ma used a threshold to detect excessive crowding from the weighted sum of pixels in the foreground mask.

Ma [121] and Celik [33] also investigated “whether the geometric correction derived for the ground plane can be applied to human objects standing upright to

the plane” [121]. That is, if weights are applied ‘blindly’ to a foreground mask, those pixels corresponding to the human upper body and head will be weighted as if they were located on the ground plane behind them. Consequently, those pixels will be weighted by a higher value than they perhaps should be. One solution to this problem is to only weight a foreground segment (or blob) according to the pixel at its base, as this is the part of the blob which makes contact with the ground plane (i.e., those pixels corresponding to the human’s feet). Although applying the weights in this manner will accurately compensate for perspective on an *individual* person, the same cannot be said for larger blobs corresponding to multiple people, because these people may not be standing near the same point on the ground plane. An example of this is shown in Figures 2.2 and 2.10(c), in which the largest blobs correspond to several widely-spaced pedestrians.

Celik [33] compared a number of pixel weighting strategies in practice, including the following:

1. “[A]ll pixels of a blob are weighted equally, using the weight of the bottom pixel in the blob.” (This pixel is assumed to correspond to the feet in contact with the ground plane.)
2. A ‘blind’ pixel weighting approach using the density map, as proposed by Paragios and Ma, in which they “assign linearly different weights to different pixel rows in the blob.”
3. A combination of strategy 1 and 2, in which “blobs longer than the human model... are linearly weighted. Pixels of other blobs are weighted equally”.

Celik’s results indicate that the first strategy performs slightly better than the second, whereas the third strategy performs significantly worse than all others. However, the evaluation was performed on a limited dataset containing crowds of

size 0 to 10. The first weighting strategy would be unlikely to work accurately on substantially larger crowds, in which a single blob may span a larger percentage of the frame. The authors noted this, stating that it “will fail in the case of a large blob containing several individuals positioned at different ordinates and occluding each other in part”. Furthermore, when modelling a human as rectangular, Ma assessed the ‘blind’ weighting (strategy 2) and derived mathematically [121]:

[I]f the scale found for the ground plane is applied blindly to foreground pixels from human bodies, the difference between this and the correct size is a constant, regardless of the location of the human bodies in the image.

This is an important result because it allows foreground pixels to be weighted directly using a density map, without the need for blob segmentation or any consideration for the location of the feet.

Hou [87, 88] utilised a similar approach to Ma [121], using the density map to accumulate a ‘blind’ weighted foreground pixel count, and performing regression with a neural network to estimate the crowd size. Hussain [94] also drew upon earlier work to build a holistic crowd counting algorithm which used weighted foreground and edge pixels to count crowds using a neural network. After explicit counting had been performed, the crowd density was determined on a five point scale, based on the number of people present and the area of the ground plane.

Kong [103, 104] proposed the use of *histograms* to describe image features on a holistic level. The blob size histogram and edge orientation histogram were used to capture the range of object sizes and their appearance in a scene. With each pixel weighted by its value in a density map, Kong calculates the size A_n of each blob, enumerated by n . The blob size histogram is constructed as follows: if the

value in the k th histogram bin is denoted $H(k)$, and the blob size for that bin is lower-bounded by a_k , then:

$$H(k) = \sum_n \begin{cases} A_n & \text{if } a_k \leq A_n < a_{k+1} \\ 0 & \text{otherwise} \end{cases} \quad (2.40)$$

That is, each histogram bin accumulates the weighted sum of pixels belonging to those blobs whose size falls within the predefined range established for that bin. Kong uses six histogram bins ($k \in [0, 5]$) of width 500, such that:

$$a_k = \begin{cases} 500k & \text{if } k < 6 \\ \infty & \text{if } k = 6 \end{cases} \quad (2.41)$$

In general, the widths and number of these bins will have to be altered according to the anticipated size and range of the blobs. This will also depend upon the resolution of the images captured.

The blob size histogram serves to separate the blobs present in an image and to place them into predefined categories. It would be expected that noise contributes to the smallest histogram bin, while individual pedestrians and small groups contribute to the second or third bins, respectively (for example). The exact nature of the relationship shall be learned by the regression model, but the use of blob size histogram bins as image features will enable it to distinguish between groups of people and individuals.

It is necessary to distinguish between groups and individuals due to the occlusion which occurs in a group setting, when one pedestrian partially blocks another from view. Kong describes the reasoning for using histograms [104]:

This histogram serves two purposes. Firstly, it can model noise re-

sulted from background subtraction. Secondly, the difference between individual pedestrian and group of pedestrians can be captured. For example, a single pedestrian may occupy 100 pixels. When two pedestrians are together, due to occlusion, there will be less than 200 pixels. In other words, each pixel contributes less to the final counts when the blob size is large.

This approach is based on the idea that the relationship between the number of pedestrians and foreground pixels is a function of the group size itself: the larger the group, the more occlusion is likely to be present. A crowd of twenty people could be distributed as sparse individuals or as one large group, for example, but the number of foreground pixels will be fewer in the latter scenario. By separating the blobs into different classes and constructing a blob size histogram, Kong compensates for occlusion while still performing regression at a holistic level.

Kong also used the Canny edge detector [32] to extract edge pixels and their angle of orientation. These pixels are masked by the foreground so that those edges in the background are ignored. An edge angle histogram is constructed with eight bins between 0° and 180° . The edge orientation histogram “can distinguish edges caused by pedestrians, which are usually vertical, with other scene structures such as noise, shadows and cars” [103]. There is support for this statement in other visual surveillance research. For example, Dalal [57] described a similar descriptor called the histogram of oriented gradients (HOG), which uses edge detection to populate a histogram of edge orientations for the explicit purpose of human detection.

The edge angle histogram is computed as follows. Canny edge detection produces a binary image C whose elements are denoted $C(i, j) \in [0, 1]$, where 1 represents

an edge. The orientation of this edge is denoted $\angle C(i, j)$. If the value in the k th histogram bin is denoted $E(k)$, and the edge angle for that bin is lower-bounded by θ_k , then for $k \in [0, 7]$:

$$E(k) = \sum_{(i,j) \in C} \begin{cases} \sqrt{S(i,j)} & \text{if } C(i,j) = 1 \text{ and } \theta_k \leq \angle C(i,j) < \theta_{k+1} \\ 0 & \text{otherwise} \end{cases} \quad (2.42)$$

where S represents the density map used to normalise for perspective. The square root of S is used to normalise one dimensional features such as edges, whereas two dimensional features (such as foreground area) are normalised using the density map S directly.

The feature vector \mathbf{x} used by Kong to represent an image is the concatenation of the blob size histogram and the edge angle histogram:

$$\mathbf{x} = [H(0), H(1), \dots, H(5), E(0), E(1), \dots, E(7)] \quad (2.43)$$

Both linear regression and neural network regression were used to model the crowd size as a function of \mathbf{x} . Equal performance was observed on one dataset while the neural network performed better on the second.

Chan [34, 35, 36, 38] proposed a holistic algorithm which extracted a very large number of features from each image in order to account for occlusion and other non-linearities such as segmentation error. Rather than use traditional background modelling techniques, a unique segmentation algorithm was used by Chan [37] to identify foreground motion. The segmentation is based on dynamic textures (an extension of textures into the temporal domain), defined by Soatto [177] as “sequences of images of moving scenes that exhibit certain stationarity properties in time”. Examples of dynamic textures include fire, smoke and wave

ripples. For the purpose of crowd counting, Chan considers an outdoor pathway on which pedestrians were classified as walking either toward or away from the camera. Treated as different instances of dynamic textures, these two classes of pedestrian motion were segmented from the background and from one another. This effectively yielded two “foreground” masks; one for each direction.

This approach requires some additional training to learn the dynamic textures, however there is a clear advantage of using this method in some surveillance applications. For example, an airport walkway in which the flow of traffic should only be permitted in one direction would benefit from detecting abnormal motion classes. Because the textures are considered over many frames using 3D spatio-temporal ‘patches’, segmentation is achieved more accurately and with greater efficiency than optical flow-based techniques [37].

Chan extracts a large number of holistic image features from the foreground mask for each direction, including foreground area, perimeter pixel count, edge orientation histogram and textural features. In total, 30 features are extracted and Gaussian process regression (GPR) is used to predict the number of pedestrians walking in each direction. The rationale for using this many features is described by Chan [34]:

Ideally, features such as segmentation area or number of edges should vary linearly with the number of people in the scene. However, local non-linearities in the regression function arise from a variety of factors, including occlusion, segmentation errors, and pedestrian configuration (e.g. spacing within a segment). To model these non-linearities, we extract a total of 30 features from each crowd segment.

The accuracy of this approach comes at the expense of additional training data requirements. As stated previously, the quantity of features will affect the clas-

sification or regression stage, including the required size of the training dataset. The implementation described by Chan utilised 800 frames of training data, which were manually annotated with ground truth (the number of pedestrians moving in each direction). This would be a burdensome task to perform for every camera in a large facility where crowd size monitoring was required. Additionally, dynamic textures can only segment *moving* pedestrians, and not those who have stopped in the middle of the scene. Pedestrians stop frequently in surveillance footage, and this can even be caused by excessive congestion, which is often what we primarily seek to detect. An adaptive background model such as Denman [59] can continue to detect stationary objects for some time after they have come to a stop. However, it cannot distinguish between pedestrians passing either direction.

Hsu [89] proposed the use of the Discrete Cosine Transform (DCT) to extract frequency information from a downsampled foreground image, and uses SVMs to classify the crowd density into five categories. The DCT contains frequency information about the image, as well as a DC coefficient which represents the average value of the image. This DC component is a measure of total foreground pixels detected, while the frequency information behaves more like a gradient or textural feature.

Dimensionality reduction techniques have been used by some authors. Zhang [200] proposes high dimensionality holistic features followed by dimensionality reduction using principal component analysis (PCA) and kernel dimension reduction (KDR). Tan [181] automatically selects a subset of 129 holistic features and uses semi-supervised elastic net (SSEN) regression on this reduced feature set.

In summary, holistic approaches are based on the intuition that a global metric (crowd size) is best estimated from global image properties (holistic features). The major algorithms discussed in this section have been summarised briefly in

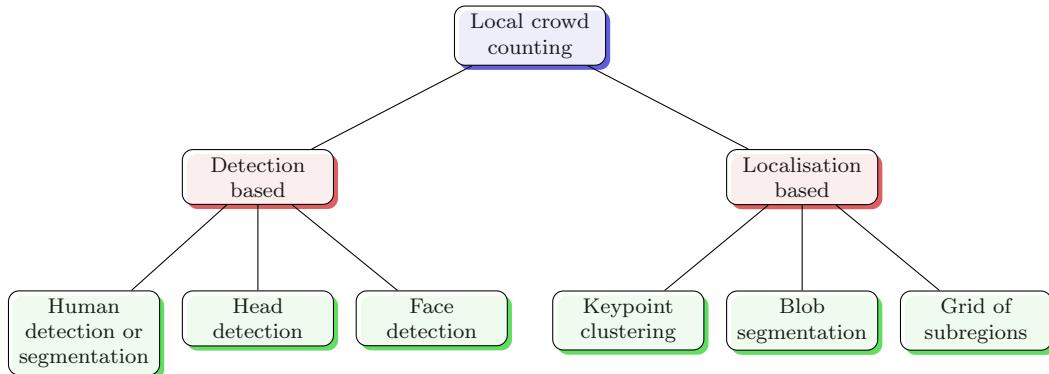


Figure 2.11: A high-level categorisation of **local** crowd counting algorithms.

Table 2.1. Crowd size is difficult to monitor due to the high variation in crowd behaviours, distribution and density. In contrast to holistic techniques, local approaches attempt to estimate crowd size by dividing the problem into a set of smaller tasks: explicitly detecting and counting pedestrians, for example; or dividing an image into a grid of sub-images. These methods are discussed in the following section.

2.3.2 Local Approaches

Local approaches to crowd counting utilise detectors or features which are specific to individuals or groups of people within an image. These groups are independently analysed, so that the total crowd estimate is the sum of its parts. Generally these methods can be categorised as follows (Figure 2.11):

1. **Detection based** approaches utilise head, face or human detectors and/or segmentation algorithms to obtain the approximate location of each individual within the scene. Crowd counting is then performed as a subsequent step.
2. **Localisation based** methods divide an image into a number of subregions

and then apply counting techniques locally.

In sparse crowds it is more appropriate to use individual pedestrian detection. For example, Dalal [57] introduced the histogram of oriented gradients (HOG) to represent images, using an SVM classifier to detect humans. Felzenszwalb [67, 68] used deformable part models to detect a variety of object classes including humans and vehicles. These kind of approaches are best suited for sparse environments in which the detected object is fully visible. As this thesis is concerned with crowded and occluded environments, these methods are not discussed in detail here. A survey of existing pedestrian detection methods can be found in [62, 65].

An alternative to pedestrian detection is crowd segmentation into larger groups. This methodology attempts to explain the observed image features by estimating not just the number of people, but also their (approximate) spatial arrangement in the scene. Zhao [203] suggests that this information can be inferred from the foreground mask alone (Figure 2.12), and proposes a method for human segmentation within a model-based Bayesian framework. The human 3D model consists of four ellipsoids with adjustable parameters, including pose (Figure 2.12(c)), position, height and fatness. These parameters are denoted $M_i = \{l_i, x_i, y_i, h_i, f_i\}$ for person i , and the overall solution includes the number of people n with their associated parameters: $\theta = \{n, \{M_1, M_2, \dots, M_N\}\}$. In order to find the optimal parameter set θ^* , the problem is formulated as a maximum a posteriori (MAP) estimation:

$$\theta^* = \operatorname{argmax}_{\theta} P(\theta|F) \quad (2.44)$$

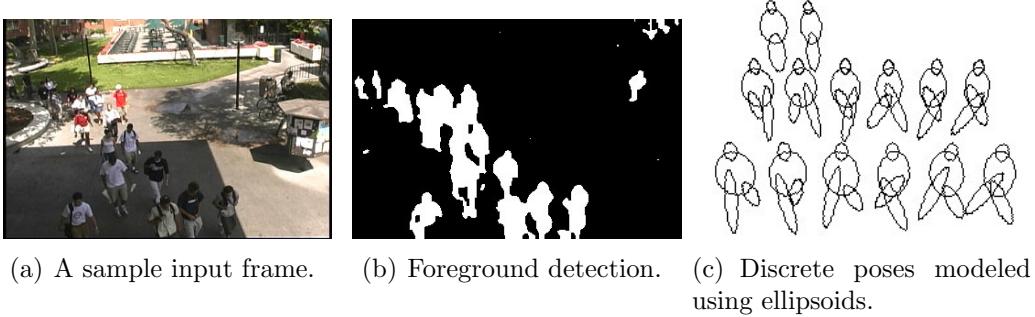


Figure 2.12: Zhao [203] uses the foreground mask to segment individuals based on a discrete set of pose models.

where F is the foreground mask. By Bayes' rule,

$$P(\theta|F) \propto P(F|\theta)P(\theta) \quad (2.45)$$

where $P(F|\theta)$ is the likelihood function of the observed foreground, given the parameter set θ , with prior probability $P(\theta)$. The prior probability term $P(\theta)$ is formulated as the product of the priors on all individuals in the scene, which penalises excessively large or small models, assumes a Gaussian distribution of height and fatness, and uniform distribution of spatial position in the scene. The likelihood term $P(F|\theta)$ is the probability of the observed foreground mask given the parameter set θ . This is determined by generating an expected foreground mask using the ellipsoid models and penalising incorrectly classified pixels.

Computing the optimal θ^* is difficult due to the numerous permutations of possible solutions. The posterior probability contains many local extrema, and the dimensionality of the solution space varies with the number of people present. In order to efficiently arrive at a near-optimal solution, Zhao employs the Metropolis-Hastings algorithm [82, 135], a popular Markov Chain Monte Carlo (MCMC) method, to sample from the posterior probability distribution so as to search

for the maximum. Each iteration of the algorithm utilises a Markov chain to arrive at the next sample, by performing either a jump (adding or subtracting a pedestrian from the proposed solution) or diffusion (stochastically altering a pedestrian’s parameters). These jump-diffusion dynamics allow the MCMC algorithm to traverse the solution space non-exhaustively, within regions of high probability.

A weakness of this technique is the inability to perform real-time segmentation in crowded situations containing more than 10-15 occupants, due to the higher dimensionality of the solution space and its many possible permutations. The utility of various pose models is also questionable in larger crowds where such information is likely to be occluded from view, and in any event, blobs rarely resemble the ellipsoid models of Figure 2.12(c). Although the MCMC approach is faster than an exhaustive search, it still requires hundreds or thousands of iterations to adequately sample the posterior probability distribution.

Reversible jump MCMC (RJMCMC) has also been used by other authors to perform crowd segmentation. For example, Ge [71, 73, 74, 75] proposes an example-based approach, by constructing a mixture model of Bernoulli shapes to represent foreground humans from a training dataset. Using these 2D models, RJMCMC is employed in a similar manner to Zhao to estimate the crowd configuration which best explains the foreground. Ge extends this to a multi-camera framework in [72, 74].

Instead of using explicit shape models, Dong [63] utilised an example-based approach in which shape descriptors were used to represent blobs in compact form. A blob’s approximate shape is encoded using Fourier descriptors, discarding high frequency coefficients as these contribute little to the overall blob shape. It was found that seven Fourier coefficients provided a sufficient representation (Figure 2.13). The training dataset contained groups of pedestrians arranged in various

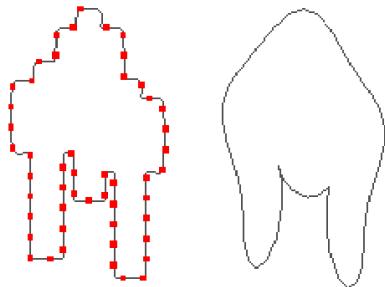


Figure 2.13: A synthetic blob shape sampled at 50 perimeter points, and its reconstructed shape using 7 Fourier descriptor coefficients. Images from [63].

configurations, so that new data can be assessed by interpolating from the examples using K Nearest Neighbours (KNN) regression. On unseen test data, KNN takes the sample mean among the closest K sample points (based on Euclidean distance). Unfortunately this approach is limited by the amount of training data available and cannot scale to arbitrarily large crowds. As group size increases, it becomes increasingly difficult to obtain sufficient training data for all of the various pedestrian configurations, and the example-based approach becomes insufficient. Dong’s approach was demonstrated on groups of size 1-6.

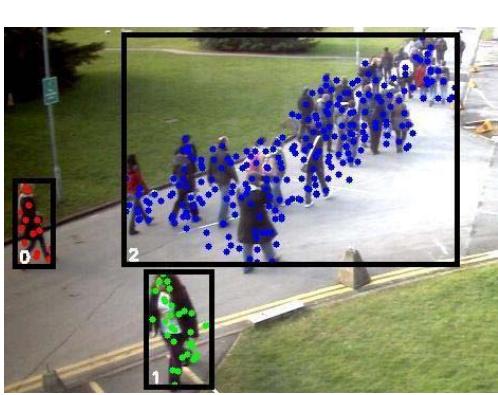
Chen [43] detected regions of symmetry to identify humans, and used iterative fitting to find human models in the foreground mask. It is only demonstrated on sparse scenes with 6-8 people, and is not designed for crowded environments. Rabaud [151] used a parallelised KLT tracker [174, 186] to determine partial tracks of interesting feature points across an image. These trajectories are then conditioned and clustered in order to estimate the number of pedestrians walking in a scene. Masoud [131] applied object tracking to perform pedestrian counting in relatively uncrowded environments.

A number of other crowd segmentation approaches have been proposed, using for example 2D models and expectation maximisation [115, 159], however they are not designed to operate in arbitrarily large crowds.

Overhead cameras have also been proposed to simplify the counting problem. Schofield [170, 171] utilised the foreground detection from an overhead camera in a lift to perform a direct object count by region search. An iterative procedure was used to eliminate human-sized regions from the foreground mask, incrementing the person count with each removal, until no objects remained to be counted. The approach is constrained to environments utilising an overhead camera. Teixeira [183] proposed a similar approach for low power sensor nodes. Frame differencing is used from an overhead camera to detect regions of motion. Person-sized motion histogram bins are then used to compute a density estimation; local maxima are computed to detect histogram peaks which correspond to humans.

Head detection has been proposed by a number of authors. Zhang [201] proposed head detection from an overhead camera using a Model-specified Directional Filter (MDF). Lin [113] used the Haar wavelet transform (HWT) to extract features of the head contour, which is processed by an SVM classifier. Merad [15, 134] segmented the human body by means of a skeleton graph into head, torso and limbs. From this decomposition head pose estimation was performed and the number of individuals counted. Patzold [147] combined shape and motion information to perform head localisation. Dalal's histogram of oriented gradients (HOG) was adapted to detect head contours only (rather than the entire body) and optical flow is used to track these head detections as pedestrians pass through a scene.

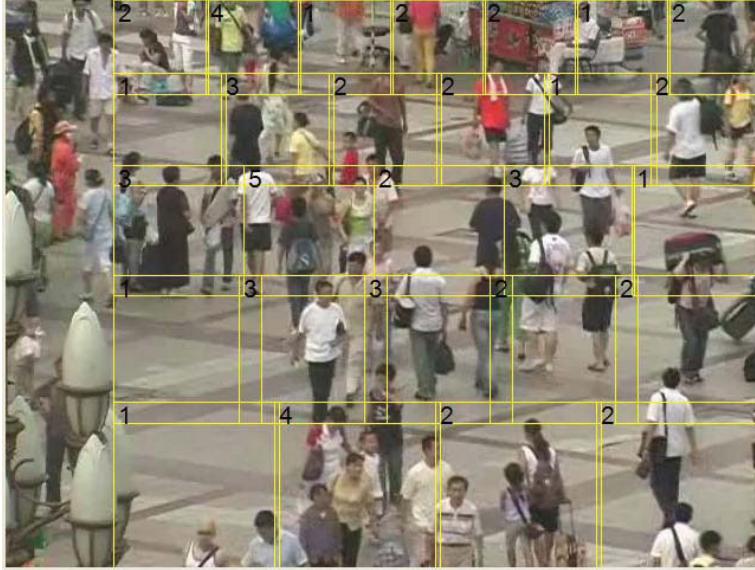
Similarly, Chen [40, 41] proposed the use of face detection using SVMs to count the number of people watching an electronic billboard advertisement. This setup required subjects to be looking toward the camera and was demonstrated on crowds of size 1-10 people. Zu [210] also uses face detection in a subway corridor as people pass toward the camera. These approaches are useful in crowds where the face of each individual is always visible to the camera, although they do not provide a general solution to the crowd counting problem.



(a) Moving feature point clustering, used by Conte [53].



(b) Foreground localisation with elliptical modelling, used by Kilambi [97].



(c) Grid of multi-resolution cells, used by Ma [123].

Figure 2.14: Localisation based approaches to crowd counting.

The aforementioned approaches are **detection based** algorithms and are generally based on the assumption of low crowd density or specific camera placement. By contrast, **localisation based** strategies divide the image into a number of subregions and attempt to count groups within the crowd locally. These approaches are categorised generally in Figure 2.11 and depicted visually in Figure 2.14.

Conte [52, 53, 54, 55] proposed moving keypoint clustering to perform group localisation. In this approach, the Speeded-Up Robust Features (SURF) algorithm proposed by Bay [18] was used to detect keypoints within an image. These points are then masked by optical flow so that stationary points are ignored. The remaining moving points are clustered into groups using the K -means algorithm, from which localised group size estimation is performed using ϵ -support vector regression (ϵ -SVR). The localisation strategy is depicted in Figure 2.14(a). The approach is built upon the work of Albiol [6], in which Harris corners [81] were employed in conjunction with linear regression on a holistic level. Acampora [2] extended the work of Conte by using adaptive neuro-fuzzy inference systems (ANFIS) for regression. The authors concluded that “the neuro-fuzzy based estimator obtains good performance when scenes are characterized by a high density crowd, whereas the approach based on ϵ -SVR works in a better way when the scenes are characterized by a low density crowd.” These approaches are limited to *moving* pedestrians because the keypoints are masked by the optical flow field.

Foreground detection has also been used by a number of authors. Celik [33] proposed a blob based algorithm which does not require training. It assumes a direct linear relationship between the number of pixels within a blob segment and the number of people represented by that segment, in order to obtain an estimate for each group. The approach is tested in relatively unoccluded environments and is based on an untrained linear model rather than machine learning approaches. Kilambi [96, 97] and Fehr [66] modelled a group of pedestrians as an elliptical cylinder, assuming a constant spacing between people within the group. Tracking a large blob over several frames increases the robustness of the group size estimate. However, the application to complex crowds in which blobs regularly split and merge may be challenging. It is unclear how a cylinder model [66, 96, 97] or a linear model [33] would hold up in larger crowds under various configurations, such as that shown in Figure 2.10(c) (p. 48).

A number of authors have used a grid of subregions, whereby an image is divided into a number of smaller cells and analysed locally, or even on a pixelwise basis. These approaches have been used to detect local abnormality with binary classifiers [194], to classify discrete density levels [64, 122, 123, 124] or to explicitly count crowds within in each cell [42, 110].

Wu [194] applied texture analysis to a number of small ‘multi-resolution density cells’, which are spaced across an image at various locations. The resolution of the cell is dependent upon the location in the image, to compensate for the effects of perspective. Within each cell, the textural features used previously by Marana [128] were calculated from the GLCM, enabling the system to estimate crowding for that particular region of the scene. The purpose of this system was to detect local abnormality due to overcrowding or undercrowding. These conditions were detected using a support vector machine (SVM) classifier, with binary output.

Dong [64] also used multi-scale patches within an image to extract texture features from the grey-gradient dependence matrix (GGDM). The GGDM is similar to Haralick’s GLCM [79], however it incorporates both grey levels and gradients. Similarly, Ma performed local texture analysis based on the GLCM [123] or local binary patterns (LBP) [122, 124]. The localisation strategy is shown in Figure 2.14(c). In these approaches, density estimation was performed using four classes, similar to the approach of Xiaohua (Figure 2.5, p. 37) and others.

Chen [42] also used local feature mining to count crowds directly (rather than to perform density level estimation). Features are extracted from equally sized cells in a rectangular grid. Perspective normalisation is performed using a density map, as introduced by Paragios [144] and Ma [121]. Multiple output ridge regression is used to capture both global and local trends in the image. Lempitsky [110] takes this a step further and estimates the fractional crowd density at each *pixel*, so that integrating the density over any region would yield the number of people

in that region. Each pixel \mathbf{p} is represented by a feature vector $\mathbf{x}_\mathbf{p}$, containing local foreground and gradient information. A linear model is used to obtain the density at each pixel,

$$F(\mathbf{p}) = \mathbf{w}^T \mathbf{x}_\mathbf{p} \quad (2.46)$$

so that the count is obtained across a region of interest R by integrating over F as follows:

$$\sum_{\mathbf{p} \in R} F(\mathbf{p}) \quad (2.47)$$

In this approach ground truth annotation is performed by assigning fractional targets to each pixel. Each pedestrian in the training dataset is annotated with a central ‘dot’ and this is dispersed across a neighbourhood using a Gaussian kernel whose elements sum to 1. Each frame in the training dataset contains a very large number of samples (one for each pixel), which provides much more training data than holistic methods (one sample per frame), minimising the number of training frames required. A sample for every pixel could make training problematic for large datasets, however, due to the very large quantity of samples.

2.3.3 Summary

Existing crowd counting algorithms are predominantly holistic in nature, employing machine learning techniques to perform regression between image features and crowd size. In recent years a number of local systems have been proposed, although many of these algorithms are detection based and rely on assumptions about camera placement or visibility of human features such as head, face or

body parts. Other local approaches are restricted by assumptions about crowd configuration, such as the elliptical model or linear model, or rely upon optical flow which can only detect *moving* pedestrians.

Moreover, a lack of testing across multiple datasets has made it difficult to compare and contrast different methodologies. Popular datasets include the PETS 2009 database [149] and the UCSD dataset [36]. In recent years a number of newer datasets have also been introduced, and these are discussed in Section 3.3.2 and Section 4.3.1. A comprehensive analysis across multiple datasets is required to compare local and holistic methods, and to compare image features such as foreground pixels and edges.

Existing methods are largely restricted to single camera viewpoints: the crowd counting system must be trained using the same scene in which it will be deployed. This necessitates that the system be re-trained for every new viewpoint. The exceptions to this rule are those algorithms using pedestrian detection or head/face detection. Existing methods are also unable to count crowds across multiple overlapping viewpoints and to compensate for this overlap. These limitations motivate the design of a new algorithm which incorporates camera parameters to improve the generality and practicality of crowd counting systems.

2.4 Crowd Flow Monitoring

Crowd flow monitoring is related to the problem of crowd counting. Crowd counting (Section 2.3) is concerned with pedestrian occupancy over a wide region of interest (at any instant in time). By contrast, crowd flow monitoring is concerned with crowd movement over a wide period of time at a specific location in the scene. For example, the number of people entering or exiting a doorway is a

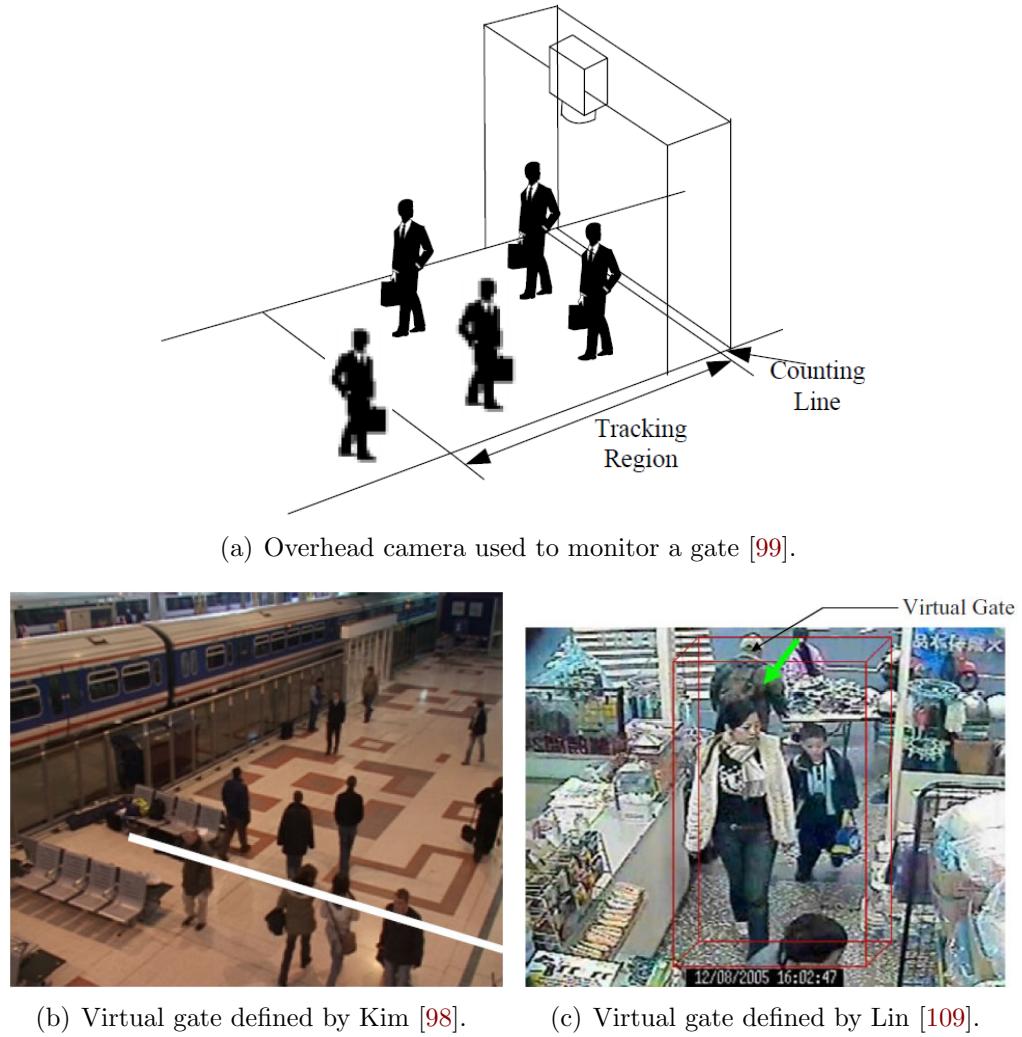


Figure 2.15: Current approaches to crowd flow monitoring.

measure of crowd flow.

Many techniques in the literature have employed overhead cameras [6, 17, 39, 44, 45, 46, 99, 184, 190] as depicted in Figure 2.15(a). Zenithal camera angles are advantageous because occlusion between pedestrians is much less severe than with oblique mounted cameras. This makes object and feature tracking much easier to perform. However, this kind of solution is not very flexible as it cannot be employed for the vast majority of security cameras already in place.

For more general camera angles, a variety of methods have been proposed. Existing algorithms have typically fallen into one of two categories:

1. **Detection and tracking.** These approaches utilise object or feature tracking [7, 39, 173, 198]. In crowded scenes where the camera is sufficiently close to the scene, face detection has been used to track incoming pedestrians [47, 205] as well as head and shoulder contours [175].
2. **Regression of optical flow.** These approaches adopt a statistical approach more suitable for crowded scenes. They typically calculate the optical flow fields within a gate and accumulate the motion vectors over time [20, 27, 98]. A regression function is learned to estimate the number of people entering based on the optical flow.

For example, Kim [98, 109] has approached the problem by defining a ‘virtual gate’ in the scene as a single line (Figure 2.15(b)). In order to estimate the pedestrian count, optical flow perpendicular to the gate is integrated over time, scaled by a learned coefficient, and rounded to the nearest integer during stationary periods. By contrast, Lin [112] presents a method for segmenting entry and exit events of pedestrians using a 2D virtual gate rather than a single line (Figure 2.15(c)). Motion is detected using optical flow and events are segmented in time. As such, the definition of a virtual gate is somewhat flexible, as it may refer to either a counting line or a region of interest.

These existing methods are currently evaluated on different datasets, making comparison between algorithms difficult or impossible. Current crowd datasets, such as the PETS 2009 [149] database, do not contain sufficient data to adequately test and compare pedestrian flow algorithms. This is due to the fact that, unlike crowd counting (which is performed on individual frames), crowd *flow* monitoring is performed across an entire *sequence*. In order to evaluate performance over

multiple sequences, a very large quantity of annotated crowd data is needed. A direct comparison of features and system parameters is currently unavailable for these reasons, and this motivates the research presented in this thesis.

2.5 Queue Analysis

A queue is a specific type of crowd arranged in an orderly manner, and is common in public spaces such as airports and shopping centres. It is characterised by its spatial arrangement, length, arrival rate, service rate and wait time. Successful queue management requires active monitoring of these properties, in order to determine whether additional service stations should be opened or closed, and to plan for queues in the future.

There is a substantial body of literature on queueing theory. A queueing system can be fully described by six characteristics [78]:

1. **Arrival pattern.** Typically the arrival process is stochastic, and a probability distribution describes the time between successive arrivals.
2. **Service pattern.** The service pattern may be a stochastic or deterministic process depending on the queueing system.
3. **Queue discipline.** In human queues this is typically first-in-first-out (FIFO).
4. **System capacity.** This pertains to the physical limitations of the queueing space.
5. **Number of service channels.** The number of parallel service stations in operation at the front of the queue.

6. **Stages of service.** The number of stages through which a customer must progress while being serviced (typically one counter).

Common practice is to describe a queue by the arrivals process, service process and the number of service channels. For example, the notation M/D/1 is commonly used to describe a queueing system with a Poisson arrival process¹, a deterministic (fixed) service time and 1 service counter. Gross [78] describes three responses of interest of the system:

1. **Wait time.** The time spent in the queue by a customer.
2. **Customer accumulation.** The number of customers in the queue.
3. **Idle time of the servers.** The percentage of time that any server is idle, or alternatively the entire system is devoid of customers.

Of these, “what we most often desire in solving queueing models is to find the probability distribution for the total number of customers in the system” [78, p. 10]. A direct observation of queue length, arrival rate and service rate will therefore assist the queueing analyst “to determine the optimum number of channels to maintain and the service rates at which to operate these channels” [78, p. 9].

Computer vision provides a possible avenue for observing these queue parameters directly. Currently, there are very few methods in the computer vision literature for detecting or analysing queues. Aubert [13] proposed a method for estimating queue length in single images based on the level lines image representation. The algorithm was applied to images of a single subway station, as shown in Figure 2.16(a). Queue length was assessed in terms of pixels, rather than the number of people, and the system did not measure throughput rate or growth rate.

¹The M stands for Markovian, or memoryless [78].



(a) Queue length measured in terms of pixels by Aubert [13].

(b) Queue and counter zones used by Parameswaran [145].

Figure 2.16: Current approaches to queue monitoring.

Parameswaran [145] has proposed a queue monitoring system that incorporates recent advances in crowd counting. The approach is based on adaptive background modelling techniques to detect foreground pixels belonging to people in the scene. It is assumed that the number of foreground pixels is related to the number of people in the scene. Their crowd counting algorithm is based on the work of Paragios [144].

Parameswaran's system also makes use of 'counter zones' which are established at each service desk in the scene. These are depicted in Figure 2.16(b). Pedestrian detection techniques [63, 204] are used to detect whether or not a service desk is occupied. The length of time taken to be serviced at each desk is recorded by the system, and the average service time is denoted \bar{S}_N . When the queue is comprised of N people, the estimated wait time is:

$$T_{avg} = N \bar{S}_N \quad (2.48)$$

The system does not estimate throughput rates explicitly at either end of the

queue, although these could be derived at the front of the queue from the various counter zones.

No other approaches to queue analysis in human crowds were found in the current literature. Despite a lack of studies on queue analysis specifically, there has been substantial research into crowd counting (Section 2.3) and crowd flow analysis (Section 2.4) in recent years, both of which are relevant and can be adapted to queues, as described in Chapter 6.

2.6 Anomaly Detection

Anomalous events are those which do not fit the statistical pattern of motions within a scene, or those events which violate our assumptions about the scene. For example, the presence of vehicles in a pedestrian space (or vice versa) indicate an abnormality which ought to be detected and flagged by the visual surveillance system. Anomaly detection algorithms can generally be organised into two categories:

1. **Object detection and trajectory analysis.** These systems operate in relatively uncrowded scenes by tracking individuals, detecting object categories and/or detecting abnormal trajectories.
2. **Motion analysis.** These systems analyse motion patterns in a video sequence rather than attempting to distinguish objects. These approaches are better suited for crowded scenes.

Although the focus of this research is on crowded environments, the techniques used to monitor individuals are of interest because they may be applicable or

extendible to situations involving groups of people and crowds. This section reviews both types of anomaly detection: object detection and trajectory analysis is discussed in Section 2.6.1 and motion analysis is discussed in Section 2.6.2.

2.6.1 Object Detection and Trajectory Analysis

Analysis of single person behaviours can be performed in low-density crowds. In a typical application, a tracking algorithm is used to obtain a person’s trajectory, which may be classified as normal or abnormal.

An early system was presented by Collins [51] called ‘Video Surveillance and Monitoring’ (VSAM). Foreground objects are classified into object classes such as human or truck using a neural network. Humans are skeletonized to form a ‘star’ with five points (4 limbs and head). Frequency information obtained from the legs can be used to distinguish running from walking.

Stauffer and Grimson [179] tracked individuals in a scene, recording the silhouette’s position, velocity and size over the full trajectory. Vector quantisation was applied to a very large training database of trajectories to generate a codebook of prototypes. Co-occurrence statistics were then used to classify the trajectories.

Haritaoglu [80] presented the W4 surveillance system in which humans are tracked individually and as groups. The system estimates posture of individuals (as either standing, sitting, bending or lying down) using the horizontal and vertical projection histograms of the foreground blob or silhouette in question. Persons carrying luggage are detected by analysing silhouette symmetry. Groups of people were counted by attempting to locate the heads of each person, using corner detection and the vertical projection histogram.

Ayers [14] used a Finite State Machine (FSM) to monitor behaviour in an office environment. People are detected using skin colour and tracked by their head. A state model is used to follow their activities, however it is highly specific to the scene. (States include ‘near cabinet’ and ‘near phone’, for example). This system requires significant prior knowledge of the scene, and only recognises a small number of hardcoded activities.

Lou [116] analysed trajectories in training data and clusters them into distinct classes. Trajectories are compared using a spatial similarity measure similar to the Hausdorff distance. Activities are classified online by dividing the trajectory into smaller segments, with 4 basic types of action: ‘move forward’, ‘turn right’, ‘turn left’, and ‘stop’. A Bayesian classifier is implemented to do the classification, and if no existing class provides a match, the activity is detected as abnormal.

Nair [138] used Hidden Markov Models to recognise an individual person’s activity. The trajectory of a person in a corridor is obtained from a tracking module, which is utilised by the HMMs, each trained to recognise one activity. The activity is classified depending on which HMM returns the highest likelihood. If the activity cannot be recognised by any model, it is treated as suspicious. Behaviours tested included ‘enter room’ and ‘exit room’. The Abstract HMM (AHMM) is employed in [29] to model more complicated behaviours. Hu [92] uses a self-organising neural network model to assess normal and abnormal trajectories.

Nguyen [141] uses the Hierarchical Hidden Markov Model (HHMM) for activity recognition. This is an extension of the HMM, allowing sub-HMM’s with specific activities. In this example, the activity ‘have snack’ consists of sub-behaviours such as enter, go to fridge, go to chair (to eat), then leave room. These behaviours in turn consist of state transitions as the person moves about the space.

Jung [95] proposed a system for detecting abnormal trajectories using two cues:

‘spatial occupancy’ and ‘trajectory consistency’. Spatial occupancy assesses whether the locations of a pedestrian are consistent with normal behaviour. Even if the pedestrian is within the normal regions, trajectory consistency then assesses whether the trajectory itself is abnormal. Based on training data, a ‘spatial occupancy map’ is calculated which specifies the probability of a pedestrian occupying each region in an area under normal circumstances. This map is used to calculate the spatial occupancy and trajectory consistency as a pedestrian passes through a scene.

Xiang and Gong [196] proposed an unsupervised approach to behaviour modelling and abnormality detection. The system learns the behavioural models incrementally and adaptively given a small training data set. Each foreground blob segment is represented by a feature vector (including centroid, bounding box dimensions, and direction of motion). The feature vector is assumed to belong to one of K_e event classes, and a Gaussian Mixture Model (GMM) is used to calculate the probability of each class. Thus each frame is represented by a K_e -dimensional vector of probabilities, and a sequence of frames constitutes a *behaviour*.

The training data set is split into a number of small video segments, each modeled by a Dynamic Bayesian Network (DBN) such as a Multi-Observation Hidden Markov Model (MOHMM). The Multi-Observation HMM is used because each event frame is represented by a K_e -dimensional vector (i.e. multiple observations). Each training segment’s MOHMM is compared to one another by computing an affinity matrix. This is used to cluster the behaviour patterns into K_c similar groups using a spectral clustering algorithm. Those clusters containing fewer members are deemed abnormal. Two models, M_n and M_a (normal and abnormal), are constructed as mixtures of MOHMMs.

Online video segments are classified using the Likelihood Ratio Test (LRT) to

determine whether they belong to M_n or M_a . This is done for behaviour P_{new} by comparing the ratio $\frac{P(P_{new}|M_n)}{P(P_{new}|M_a)}$ to a predetermined threshold. P_{new} is then incorporated into either M_n or M_a accordingly. Testing was performed on a corridor scene featuring 1-2 people.

Interactions between *multiple* persons have also been analysed using pedestrian trajectories. Oliver [143] compared the HMM and the Coupled HMM (CHMM) for detecting three types of human interactions (between two people): ‘meet and continue together’, ‘meet and split’, and ‘follow’. A feature vector is computed for each pair of nearby persons, containing their relative distance, the velocity of each person, and the degree of trajectory alignment. The CHMM is an extension of the HMM, incorporating multiple state variables. The CHMM used in [143] contains 2 state variables, corresponding to each person.

Park [146] proposed a system for the recognition of two-person interactions using a hierarchical Bayesian Network (BN). Body-part features are extracted and represented by ellipses and convex hulls. The Bayesian network accurately estimates body poses and recognises actions such as pointing, punching and pushing. However the system requires a side-on view and can only analyse two interacting people.

Wu and Ou [195] detect abnormal poses of individuals. A blob is tracked over several frames. The distance from the centroid to each point on the silhouette border is calculated and the Discrete Fourier Transform (DFT) is applied to the sequence. Twenty primary DFT coefficients from four consecutive blobs (80 total) are selected. Principal component analysis (PCA) is applied to reduce dimensionality. A support vector machine is used to classify the blob sequence as normal or abnormal. Abnormal behaviours such as ‘carrying a bar’, ‘running’ and ‘bending down’ were detected (as opposed to the normal behaviour, walking).

These approaches are generally based on foreground detection and object tracking, with subsequent analysis being performed on the foreground blobs or their trajectories. The most popular method for detecting behaviour classes is the Hidden Markov Model, which provides a statistical model of a temporal sequence of observations. The HMM is discussed in greater detail in Section 7.3.4.2 (p. 340). Although pedestrian detection and trajectory analysis is not feasible for arbitrarily large crowds, the use of statistical models such as GMMs and HMMs have informed the design of motion analysis algorithms.

2.6.2 Motion Analysis

Motion analysis is typically based on the optical flow patterns extracted from a video sequence, which are subsequently classified as normal or abnormal. The statistical models used to detect such anomalies include bag of words [133, 193], hidden Markov models [11, 106], Markov random fields [100] and other models as discussed below.

2.6.2.1 Bag of Words

Wang [193] used hierarchical Bayesian models to perform anomaly detection in crowded scenes. The approach uses three levels, as depicted in Figure 2.17. Activities are modelled as probability distributions over low-level motion features, while interactions are modelled as distributions over these atomic activities.

Their approach builds upon two hierarchical models, Latent Dirichlet Allocation (LDA) [23] and the Hierarchical Dirichlet Process (HDP) [182]. LDA is a generative probabilistic model composed of *words* from a discrete, finite vocabulary; latent *topics*, modeled as probability distributions over words; *documents*,

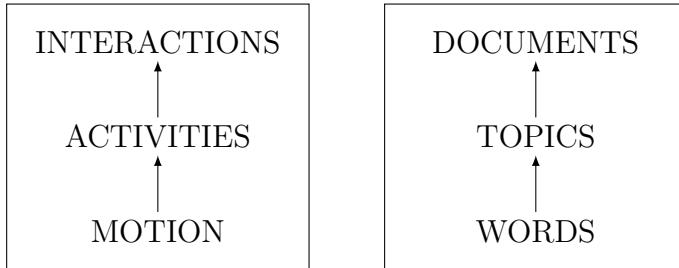


Figure 2.17: The analogy between event detection and language processing.

containing a sequence of words; and a corpus of such documents.

As with other document modeling and clustering techniques, LDA assumes that the words contained within a document are *exchangeable* (their order is unimportant). A document under this assumption is often described as a “bag of words”.

In a visual surveillance context, low-level pixel motion corresponds to the “words” within a video clip, the “document”. The latent topics may describe atomic activities such as “car turning left” or “pedestrian crossing the street”, although these are automatically learned by the system and might not be defined so explicitly.

As the standard LDA model does not model clusters of documents, Wang extends this model so that each document belongs to a cluster c . As such, each cluster represents an *interaction* containing similar sets of atomic activities (topics). These might represent “vehicles from road g make a right turn to road a while there is not much other traffic” or “pedestrians walk on the crosswalks while there is not much traffic” [193].

Wang also proposes the use of HDP, which automatically decides the number of topics, and proposes a dual-HDP model which also decides the number of document clusters automatically.

The low-level features are extracted using simple image processing techniques.

Basic frame differencing is used to detect moving pixels, and the features of these pixels constitute the ‘words’ in the document. Two kinds of information are extracted from each moving pixel: location and motion direction. The motion direction is obtained from Lucas-Kanade optical flow [117].

Because the Bayesian model is based on words from a discrete, finite vocabulary, these low-level features must be quantised according to a codebook. The location information is quantised by assigning each pixel to a 10×10 cell, and direction information is quantised into four directions. Therefore moving pixels are assigned a word from the codebook based on position and motion direction. This results in a substantial loss of information, although this representation is ideal for modeling scenes with hierarchical Bayesian models.

This approach is most useful for detecting events which are anomalous within a given spatiotemporal context, for example pedestrians crossing the road outside of the designated crossing, or jaywalking. However, due to quantisation in the feature extraction process, this approach cannot be used for detecting anomalous low-level motion patterns themselves. This is acknowledged by the authors:

For example, if a pedestrian is walking along the path of vehicles, just based on positions and moving detections, his motions cannot be distinguished as abnormality. If a car drives extremely fast, it will not be detected as abnormal either.

The “bag of words” approach is suitable for detecting abnormal combinations of normal events, particularly in complicated scenes containing various classes of objects and events, such as an outdoor traffic scene. However, in a scene containing only pedestrians, Wang observes that the system did not appear to detect any “interesting interactions”.

Mehran [133] also used LDA to model normal behaviour with a different underlying visual representation, the social force model [83]. The social force model was used to model human movement as a function of forces, comprised of a ‘personal desire’ force as well as interaction forces imposed by the surrounding environment. The magnitude of the interaction force is extracted at each pixel, based on the detected optical flow fields, and behaviour is modelled as a collection of patches (bag of words) within an image.

2.6.2.2 Hidden Markov Models

A Markov model is a stochastic model consisting of a series of states which are visited in a sequential order. In a *hidden* Markov model, the states themselves are not observed, and instead a sequence of outputs corresponding to each state are observed. Typically these outputs belong to a statistical distribution such as a Gaussian function or a GMM. Section 7.3.4.2 describes HMMs in greater detail. HMMs have been used by several authors for anomaly detection in crowded scenes [8, 9, 10, 11, 106, 107, 108].

Andrade proposed a number of methods using HMMs for anomaly detection and event detection [8, 9, 10, 11]. In [8], Andrade used a HMM to model optical flow patterns within each local block of an image. The algorithm was used to detect a falling person in simulated test sequences. In [9, 10, 11], PCA was used to reduce the dimensionality of the optical flow field on a holistic level, and this reduced feature vector was modelled by a HMM. The video segments within a training dataset were grouped using spectral clustering, and each cluster is used to train a different HMM. This bank of HMMs was then used to detect sequences of various classes, enabling event classification as well as anomaly detection when the likelihood of a sequence falls below the detection threshold.

Kratz [106, 107] extracted information from local spatio-temporal patches in a video sequence. Each ‘patch’ is a 3D volume, represented by a distribution of gradients in (i, j, t) . For each pixel p in patch P ,

$$\nabla P_p = \left[\frac{\partial P}{\partial i} \quad \frac{\partial P}{\partial j} \quad \frac{\partial P}{\partial t} \right]^T \quad (2.49)$$

The distribution of these spatio-temporal gradients is modeled as a 3D Gaussian function, with:

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{p \in P} \nabla P_p \quad (2.50)$$

$$\boldsymbol{\Sigma} = \frac{1}{N} \sum_{p \in P} (\nabla P_p - \boldsymbol{\mu})(\nabla P_p - \boldsymbol{\mu})^T \quad (2.51)$$

where N denotes the number of pixels in the patch. To discriminate between patch representations, the symmetric Kullback-Leibler (KL) divergence is used. Generally, the KL-divergence between the probability density functions $f(x)$ and $g(x)$ is defined:

$$D_{\text{KL}}(f||g) = \int_x f(x) \log \frac{f(x)}{g(x)} dx \quad (2.52)$$

For two Gaussians, it has the closed-form expression:

$$D_{\text{KL}}(f||g) = \frac{1}{2} \left[\log \frac{|\boldsymbol{\Sigma}_g|}{|\boldsymbol{\Sigma}_f|} + \text{Tr}(\boldsymbol{\Sigma}_g^{-1} \boldsymbol{\Sigma}_f) - d + (\boldsymbol{\mu}_f - \boldsymbol{\mu}_g)^T \boldsymbol{\Sigma}_g^{-1} (\boldsymbol{\mu}_f - \boldsymbol{\mu}_g) \right] \quad (2.53)$$

The symmetric KL-divergence is $D_{\text{KL}}(f||g) + D_{\text{KL}}(g||f)$. Kratz associates similar patches with a small distance between them in an online manner. Each patch is

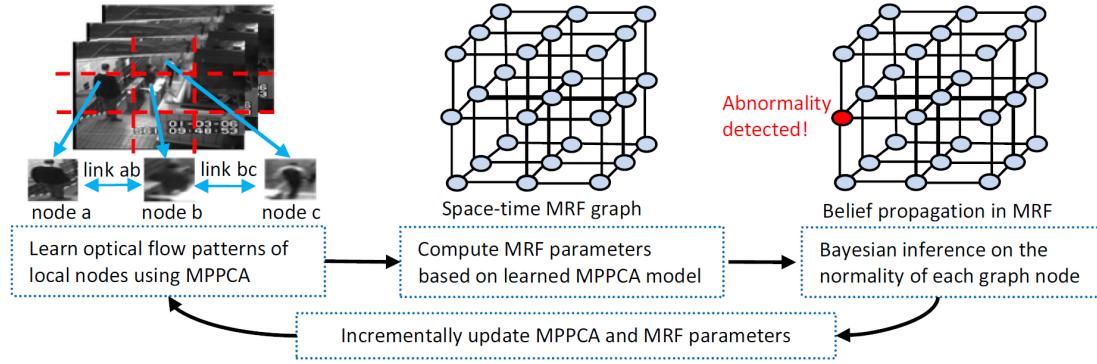


Figure 2.18: Summary of Kim and Grauman’s system. Images from [100].

compared to a bank of prototypes $\{P_s | s = 1 \dots S\}$ and if the distance is greater than a threshold $\forall s$ then a new prototype is formed. Otherwise the prototypes are updated as in [106]. A different set of prototypes is used for each region in the scene.

In order to capture the relationship between motion occurrences, a HMM is created for each location in the scene. Each state is associated with a prototype S . A coupled HMM structure is used to model neighbouring patches simultaneously and to capture their relationships.

2.6.2.3 Markov Random Fields

The Markov random field (MRF) is used by Kim [100] to obtain a *labelling* at each site in a grid. Each location in an image is classified as normal or abnormal, as depicted in Figure 2.18.

Their approach extracts optical flow using a multi-scale block matching algorithm. The region associated with each MRF node is broken down into a $u \times v$ grid of sub-regions, from each of which a 9D optical flow vector is computed (8 orientations and 1 speed). These feature vectors are concatenated to form a $9uv$ -dimensional descriptor of the region.

The dimensionality of the feature vector is reduced using the mixture of probabilistic principal component analysers (MPPCA). A common dimensionality reduction technique is Principal Components Analysis (PCA), and this has been applied to large optical flow previously for crowded event detection [11]. However, a limitation of PCA is that it defines only a single linear projection of the data into the lower-dimensional space. Tipping [185] proposed MPPCA, which models a high-dimensional observation $\mathbf{t} \in \mathbb{R}^d$ as a noisy function of a latent variable $\mathbf{x} \in \mathbb{R}^q$ ($q < d$) using a generative model:

$$\mathbf{t} = y(\mathbf{x}) + \epsilon \quad (2.54)$$

where ϵ represents the Gaussian noise model, $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$, and y is a linear function of \mathbf{x} :

$$y(\mathbf{x}) = \mathbf{W}\mathbf{x} + \boldsymbol{\mu} \quad (2.55)$$

This implies a probability:

$$p(\mathbf{t}|\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp(-\frac{1}{2\sigma^2} \|\mathbf{t} - \mathbf{W}\mathbf{x} - \boldsymbol{\mu}\|^2) \quad (2.56)$$

And assuming a Gaussian prior over the latent variables, namely,

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{q/2}} \exp(-\frac{1}{2} \mathbf{x}^T \mathbf{x}) \quad (2.57)$$

then the marginal distribution is:

$$p(\mathbf{t}) = \int_{\mathbf{x}} p(\mathbf{t}|\mathbf{x})p(\mathbf{x}) d\mathbf{x} \quad (2.58)$$

$$= \frac{1}{(2\pi)^{d/2} |\mathbf{C}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{t} - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{t} - \boldsymbol{\mu})) \quad (2.59)$$

where $\mathbf{C} = \sigma^2 \mathbf{I} + \mathbf{W}\mathbf{W}^T$. The maximum-likelihood estimate of $\boldsymbol{\mu}$ is $\boldsymbol{\mu}_{\text{ML}} = \frac{1}{N} \sum_n \mathbf{t}_n$, while an iterative EM algorithm for obtaining the ML estimates of the other parameters (\mathbf{W} and σ^2) is given in [185].

A *mixture* of M probabilistic principal component analysers then gives rise to the distribution,

$$p(\mathbf{t}) = \sum_{i=1}^M \pi_i p(\mathbf{t}|i) \quad (2.60)$$

where the mixing coefficients $\{\pi_i\}$, and each mixture's parameters $\{\boldsymbol{\mu}_i, \mathbf{W}_i, \sigma_i\}$ are estimated via the EM algorithm in [185]. This expresses a higher-dimensional feature vector as a mixture probability density over the latent subspace. Finally, $p(\ell|\mathbf{t})$ denotes the posterior probability that MPPCA component ℓ generated observation t :

$$p(\ell|\mathbf{t}) = \frac{\pi_\ell p(\mathbf{t}|\ell)}{\sum_{i=1}^M \pi_i p(\mathbf{t}|i)} \quad (2.61)$$

Letting $\mathbf{t}_{i,k}$ denote the $9uv$ -dimensional vector at node i in the k th frame, Kim defines the frequency and co-occurrence histograms over a sequence, $k = 1, 2, \dots, n$, respectively:

$$H_i(\ell) = \sum_{k=1}^n p(\ell|\mathbf{t}_{i,k}) \quad (2.62)$$

$$H_{i,j}(\ell, m) = \sum_{k=1}^n p(\ell|\mathbf{t}_{i,k})p(m|\mathbf{t}_{j,k}) \quad (2.63)$$

A node evidence function, $n(x_i)$, and a pair-wise potential function, $\rho(x_i, j_i)$, are defined in terms of the histograms $H_i(\ell)$ and $H_{i,j}(\ell, m)$ (as described in [100]). These are used to formulate the MRF as follows. Each node i is labeled normal ($x_i = 0$) or abnormal ($x_i = 1$). The objective is to infer the labeling which maximises:

$$E(\mathbf{x}) = \lambda \sum_i n(x_i) + \sum_{(i,j) \in \text{neighbour}} \rho(x_i, x_j) \quad (2.64)$$

where λ is a relative weighting. Given the MRF parameters in the node evidence and pairwise potential functions, MAP inference is used to maximise $E(\mathbf{x})$, by labelling each node as normal or abnormal.

2.6.2.4 Other Models

Adam [3] extracted optical flow from fixed spatial locations in an image in a similar manner to block-matching, however rather than obtaining a point estimate for the flow, a probability distribution is generated by considering the sum of squared differences (SSD) error corresponding to various discrete shifts in a window surrounding a pixel. These flow probabilities are aggregated into magnitude-based and angle-based histogram bins. A cyclic buffer is used to determine the likelihood of new observations, and when this likelihood falls below a threshold the scene is deemed to be abnormal.

Mahadevan [125] proposed a system which does not make use of optical flow or gradients, but rather a mixture of dynamic textures [37]. Existing methods

“focus uniquely on motion information, ignoring abnormality information due to variations of object appearance”. Mahadevan jointly models appearance and dynamics using a generative model for each component in the mixture. Patches of low likelihood are detected as outliers. Their approach outperformed the visual representations used by Mehran [133], Kim [100] and Adam [3] on the UCSD anomaly detection dataset [125].

2.6.2.5 Summary

Existing approaches to anomaly detection using motion analysis generally require the computation of dense optical flow at full resolution, which is difficult to perform in real time with substantial accuracy. They also do not capture the motion *patterns* present in a scene, but instead reduce the optical flow in some way (through quantisation, dimensionality reduction, or histogram binning). For example, the “bag of words” approach requires quantisation of features according to a codebook [133, 193], and other methods have reduced the optical flow field using PCA [9, 10, 11] or other dimensionality reduction techniques such as MPPCA [100]. In the system proposed by Adam [3], optical flow is aggregated into histograms.

These type of approaches may not always capture interesting interactions from a surveillance perspective, and novelty detection is performed as a subsequent process on this reduced data. However, if a visual representation is insufficient, an abnormal event will not be detected at a later stage of the algorithm, regardless of the statistical model used. This observation motivates the design of new visual representations to capture the meaningful properties of motion patterns in crowded environments.

2.7 Machine Learning

Computer vision algorithms make use of a variety of machine learning techniques. In this section three specific methods are reviewed, as they are used in subsequent chapters of this thesis. Section 2.7.1 reviews Gaussian mixture models; Section 2.7.2 review hidden Markov models; and Section 2.7.3 reviews Gaussian process regression.

2.7.1 Gaussian Mixture Model

The Gaussian mixture model (GMM) is a probabilistic model obtained from a mixture of Gaussian distributions. The Gaussian distribution has the density function:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{D}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right) \quad (2.65)$$

where $\boldsymbol{\mu}$ denotes the *mean* value of the distribution, $\boldsymbol{\Sigma}$ denotes the covariance matrix and D represents the dimensionality of the feature vector \mathbf{x} . A mixture model is therefore constructed as a mixture of K such distributions:

$$p(\mathbf{x}|\Theta) = \sum_{k=1}^K \alpha_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (2.66)$$

where Θ represents the entire set of parameters; α_k , $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ denote the mixture weight, mean and covariance of component k respectively. For brevity Θ is omitted from the notation henceforth.

Let us suppose that a set of N data samples, $\{\mathbf{x}_i\}_{i=1}^N$, are observed from an

unknown distribution and we seek to build a GMM to model the data. The parameters of the GMM are learned using the expectation-maximisation (EM) algorithm [21]. The procedure is summarised briefly below.

First, the probability of data sample \mathbf{x}_i having been generated by mixture component ℓ is calculated using Bayes' rule:

$$p(\ell|\mathbf{x}_i) = \frac{p(\ell)p(\mathbf{x}_i|\ell)}{p(\mathbf{x}_i)} \quad (2.67)$$

$$= \frac{\alpha_\ell \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell)}{\sum_{k=1}^K \alpha_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \quad (2.68)$$

The model parameters $\{\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ are then updated using the re-estimation equations:

$$\alpha_\ell = \frac{1}{N} \sum_{i=1}^N p(\ell|\mathbf{x}_i) \quad (2.69)$$

$$\boldsymbol{\mu}_\ell = \frac{\sum_{i=1}^N \mathbf{x}_i p(\ell|\mathbf{x}_i)}{\sum_{i=1}^N p(\ell|\mathbf{x}_i)} \quad (2.70)$$

$$\boldsymbol{\Sigma}_\ell = \frac{\sum_{i=1}^N p(\ell|\mathbf{x}_i) (\mathbf{x}_i - \boldsymbol{\mu}_\ell)(\mathbf{x}_i - \boldsymbol{\mu}_\ell)^T}{\sum_{i=1}^N p(\ell|\mathbf{x}_i)} \quad (2.71)$$

Training a GMM requires alternately computing Equation 2.68 and Equations 2.69-2.71. The procedure may be initialised randomly, or by using a pre-clustering algorithm such as K -means.

2.7.2 Hidden Markov Model

A Markov model is a stochastic model in which a series of states are visited in a sequential order. In a *hidden* Markov model the state sequence is not known

directly, and instead, a sequence of outputs are observed [21, 152]. A Markov chain is the simplest type of Markov model, consisting of states $q_t \in [1, N]$ which are visited in the sequence $Q = \{q_t\}_{t=1}^T$ for which the probability of observing a state is dependent only on the previous state:

$$a_{i,j} = p(q_{t+1} = j | q_t = i) \quad (2.72)$$

where the transition probability $a_{i,j}$ is a constant (independent of t). The *initial* probability of observing state i is denoted:

$$\pi_i = p(q_1 = i) \quad (2.73)$$

In a hidden Markov model, the state sequence is hidden and the set of observations $O = \{\mathbf{o}_t\}_{t=1}^T$ is a probabilistic function of the hidden states, such as a Gaussian mixture model:

$$p(\mathbf{o}_t | q_t = j) = \sum_{\ell=1}^M c_{j\ell} \mathcal{N}(\mathbf{o}_t | \boldsymbol{\mu}_{j\ell}, \boldsymbol{\Sigma}_{j\ell}) \quad (2.74)$$

where $c_{j\ell}$, $\boldsymbol{\mu}_{j\ell}$, $\boldsymbol{\Sigma}_{j\ell}$ denote the mixture coefficient, mean and covariance of mixture ℓ in state j , respectively.

The full set of parameters for the HMM is therefore:

$$\lambda = \{\pi, A, B\} \quad (2.75)$$

$$= \{\{\pi_i\}, \{a_{ij}\}, \{c_{j\ell}, \boldsymbol{\mu}_{j\ell}, \boldsymbol{\Sigma}_{j\ell}\}\} \quad (2.76)$$

The likelihood of an observation given the current state, j , is denoted:

$$b_j(t) = p(\mathbf{o}_t | q_t = j) \quad (2.77)$$

$$= \sum_{\ell=1}^M c_{j\ell} b_{j\ell}(t) \quad (2.78)$$

where

$$b_{j\ell}(t) = p(\mathbf{o}_t | q_t = j, X_t = \ell) \quad (2.79)$$

$$= \mathcal{N}(\mathbf{o}_t | \boldsymbol{\mu}_{j\ell}, \boldsymbol{\Sigma}_{j\ell}) \quad (2.80)$$

where X_t denotes the mixture component that generated observation \mathbf{o}_t . The likelihood of observing an output sequence is:

$$p(O|\lambda) = \sum_{\text{all } Q} \left(\pi_{q_1} b_{q_1}(1) \prod_{t=2}^T a_{q_{t-1} q_t} b_{q_t}(t) \right) \quad (2.81)$$

where $\sum_{\text{all } Q}$ denotes summation across all possible state sequences. This is computationally intractable, so a forward procedure is used to compute $p(O|\lambda)$ more efficiently using the forward variable:

$$\alpha_i(t) = p(\{\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_t\}, q_t = i | \lambda) \quad (2.82)$$

This is the probability of observing the partial output sequence up until time t , with $q_t = i$. This is computed by way of induction:

1. Initialisation:

$$\alpha_i(1) = \pi_i b_i(1) \quad (2.83)$$

2. Induction:

$$\alpha_j(t+1) = \left[\sum_{i=1}^N \alpha_i(t) a_{ij} \right] b_j(t+1) \quad (2.84)$$

3. Termination:

$$p(O|\lambda) = \sum_{i=1}^N \alpha_i(T) \quad (2.85)$$

A similar backward procedure is defined using the backward variable:

$$\beta_i(t) = p(\{\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, \dots, \mathbf{o}_T\} | q_t = i, \lambda) \quad (2.86)$$

1. Initialisation:

$$\beta_i(T) = 1 \quad (2.87)$$

2. Induction:

$$\beta_i(t) = \sum_{j=1}^N a_{ij} b_j(t+1) \beta_j(t+1) \quad (2.88)$$

3. Termination:

$$p(O|\lambda) = \sum_{i=1}^N \beta_i(1) \pi_i b_i(1) \quad (2.89)$$

In order to train a HMM, we wish to choose the parameters λ so as to maximise the likelihood $p(O|\lambda)$ of the observed sequence². The Baum-Welch procedure for training a HMM is a specific example of expectation-maximisation (EM), in which $p(O|\lambda)$ is locally maximised by re-estimating the parameters in an iterative manner [21, 152]. This re-estimation makes use of the following variables.

- The probability of being in state i at time t is denoted:

²In practice, the training dataset will comprise *multiple* observation sequences. Appendix C.2 describes how the training procedure described in this section is extended to multiple observation sequences.

$$\gamma_i(t) = p(q_t = i | O, \lambda) \quad (2.90)$$

$$= \frac{p(q_t = i, O | \lambda)}{p(O | \lambda)} \quad (2.91)$$

$$= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^N \alpha_j(t)\beta_j(t)} \quad (2.92)$$

- The probability of being in state i at time t , *and* of the observation o_t having been generated by mixture component ℓ , is denoted:

$$\gamma_{i\ell}(t) = p(q_t = i, X_t = \ell | O, \lambda) \quad (2.93)$$

$$= \gamma_i(t) \frac{c_{i\ell} b_{i\ell}(t)}{b_i(t)} \quad (2.94)$$

- The probability of being in states i and j at times t and $t+1$ is denoted:

$$\xi_{i,j}(t) = p(q_t = i, q_{t+1} = j | O, \lambda) \quad (2.95)$$

$$= \frac{\alpha_i(t)a_{ij}b_j(t+1)\beta_j(t+1)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_i(t)a_{ij}b_j(t+1)\beta_j(t+1)} \quad (2.96)$$

From these we can deduce that $\sum_{t=1}^{T-1} \gamma_i(t)$ is the expected number of time steps spent in state i , and thus the number of time transitions from state i (including self transitions); and $\sum_{t=1}^{T-1} \xi_{i,j}(t)$ is the expected number of transitions from state i to state j . With this in mind the Baum-Welch algorithm's re-estimation equations are:

$$\pi_i = \gamma_i(1) \quad (2.97)$$

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{i,j}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \quad (2.98)$$

$$c_{i\ell} = \frac{\sum_{t=1}^T \gamma_{i\ell}(t)}{\sum_{t=1}^T \gamma_i(t)} \quad (2.99)$$

$$\boldsymbol{\mu}_{i\ell} = \frac{\sum_{t=1}^T \gamma_{i\ell}(t) \mathbf{o}_t}{\sum_{t=1}^T \gamma_{i\ell}(t)} \quad (2.100)$$

$$\Sigma_{i\ell} = \frac{\sum_{t=1}^T \gamma_{i\ell}(t) (\mathbf{o}_t - \boldsymbol{\mu}_{i\ell})(\mathbf{o}_t - \boldsymbol{\mu}_{i\ell})^T}{\sum_{t=1}^T \gamma_{i\ell}(t)} \quad (2.101)$$

Training a HMM requires alternately computing Equations (2.92, 2.94, 2.96) and Equations 2.97-2.101.

In practice the computation of the forward and reverse variables α, β tend exponentially toward zero, and for long sequences this can exceed the machine's precision. Appendix C.1 describes how to maintain numerical stability in this calculation. Furthermore, Appendix C.2 describes how to estimate the HMM parameters when there are multiple observation sequences in the training dataset; it is a straightforward extension of the procedure described above.

2.7.3 Gaussian Process Regression

The Gaussian Process is defined as a collection of random variables, any finite subset of which have a joint Gaussian distribution. In regression problems, we observe N samples from a training set, consisting of feature vectors $\mathbf{X} = \{\mathbf{x}_n\}$ and targets $\mathbf{f} = \{f_n\}$. In Gaussian process regression (GPR) these targets are imagined as a sample from some multivariate Gaussian distribution, whose mean is typically taken to be zero:

$$\mathbf{f}|\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}) \quad (2.102)$$

The covariance matrix, $\mathbf{K} \in \mathbb{R}^{N \times N}$, is obtained from the covariance function

$k(\mathbf{x}_m, \mathbf{x}_n)$, such that

$$\mathbf{K}_{mn} = k(\mathbf{x}_m, \mathbf{x}_n) \quad (2.103)$$

A Gaussian process is fully specified by its covariance function, $k(\mathbf{x}_m, \mathbf{x}_n)$, and the mean function $m(\mathbf{x}) = 0$. The covariance function expresses the covariance of outputs as a function of inputs. For example, a typical covariance function is the squared exponential:

$$k_{\text{SE}}(\mathbf{x}_m, \mathbf{x}_n) = \sigma_{\text{SE}}^2 \exp\left(-\frac{1}{2\ell^2} |\mathbf{x}_m - \mathbf{x}_n|^2\right) \quad (2.104)$$

The closer the inputs, \mathbf{x}_m and \mathbf{x}_n , are to one another, the more correlated their outputs will be. The hyperparameter ℓ is a characteristic length scale which represents the distance one would expect to move in the input space to produce a significant change in the output space.

Given N^* test inputs, $\mathbf{X}^* = \{\mathbf{x}_n^*\}$, we wish to obtain the predictive outputs $\mathbf{f}^* = \{f_n^*\}$. Let \mathbf{K}^* denote the $N \times N^*$ train-test covariance matrix, whose elements are calculated by:

$$\mathbf{K}_{mn}^* = k(\mathbf{x}_m, \mathbf{x}_n^*) \quad (2.105)$$

Similarly, let \mathbf{K}^{**} denote the $N^* \times N^*$ test set covariance, with:

$$\mathbf{K}_{mn}^{**} = k(\mathbf{x}_m^*, \mathbf{x}_n^*) \quad (2.106)$$

As all variables in a Gaussian Process are normally distributed, the training and

testing data sets are jointly Gaussian [156], with the following distribution:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{K}^* \\ \mathbf{K}^{*T} & \mathbf{K}^{**} \end{bmatrix}\right) \quad (2.107)$$

Each subset of these random variables also follows a joint Gaussian distribution:

$$\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}) \quad (2.108)$$

$$\mathbf{f}^* \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{**}) \quad (2.109)$$

Prediction using Gaussian process regression is performed by conditioning the predictive outputs on the training data, with the following posterior distribution obtained for \mathbf{f}^* :

$$\mathbf{f}^* | \mathbf{f}, \mathbf{X}, \mathbf{X}^{**} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad (2.110)$$

where:

$$\boldsymbol{\mu} = \mathbf{K}^{*T} \mathbf{K}^{-1} \mathbf{f} \quad (2.111)$$

$$\boldsymbol{\Sigma} = \mathbf{K}^{**} - \mathbf{K}^{*T} \mathbf{K}^{-1} \mathbf{K}^* \quad (2.112)$$

This method provides not only point estimates, $\boldsymbol{\mu}$, but also a matrix $\boldsymbol{\Sigma}$ of covariances for the test outputs. The diagonal elements of $\boldsymbol{\Sigma}$ can thus be used to obtain pointwise error bars.

$$\sigma_n^2 = \boldsymbol{\Sigma}_{nn} \quad (2.113)$$

For example, setting a 95% confidence interval, the estimate for test sample n would be $\mu_n \pm 1.96\sigma_n$. If we only wish to use the diagonal elements of Σ it is not necessary to calculate the full covariance matrix as described in Equation 2.112. Instead, let \mathbf{K}_n^* denote the n th column of \mathbf{K}^* , and the diagonal elements of Σ are evaluated by:

$$\Sigma_{nn} = \mathbf{K}_{nn}^{**} - \mathbf{K}_n^{*T} \mathbf{K}^{-1} \mathbf{K}_n^* \quad (2.114)$$

The GPR model is ‘trained’ by choosing the hyperparameters in the covariance function (e.g. σ_{SE} and ℓ in Equation 2.104) so as to maximise the likelihood of the observed training data $p(\mathbf{f}|\mathbf{X})$. (Good predictive performance can still be achieved with reasonable estimates for these hyperparameters.) Following from Equation 2.102,

$$\log p(\mathbf{f}|\mathbf{X}) = -\frac{1}{2}\mathbf{f}^T \mathbf{K}^{-1} \mathbf{f} - \frac{1}{2} \log |\mathbf{K}| - \frac{N}{2} \log 2\pi \quad (2.115)$$

This term is maximised using an optimisation algorithm such as conjugate-gradients, provided that $k(\mathbf{x}_m, \mathbf{x}_n)$ is differentiable with respect to each of the hyperparameters. Once optimised, prediction is then performed using Equations 2.110-2.112.

Chapter 3

Crowd Counting using Local Features

3.1 Introduction

Crowd size may be an indicator of security threats such as rioting, violent protest and mass panic. In some circumstances overcrowding may pose a threat in and of itself, as occurred at a German music festival in 2010 [84]:

An extreme and fluctuating pressure builds up, when the densities become so high that they cause contact forces between bodies to add up. This ultimately implies a sudden onset of ‘crowd turbulence’. Under such conditions, the sizes and directions of forces acting on the bodies of visitors move them around in an uncontrolled way, and people have difficulties keeping their balance; when people stumble and fall, this can be the nucleus of a crowd disaster.

Crowd size can also indicate congestion and other abnormal events within peaceful crowds. Crowd information can also be used to provide operational analytics for business intelligence. Therefore automated crowd counting has become an active field of computer vision research in recent years.

Crowd size is a *holistic* description of a scene. Therefore the majority of crowd counting techniques have utilised holistic image features to estimate crowd size. However, due to the wide variability in crowd behaviours, distribution, density and overall size, holistic features may not necessarily be the best approach for monitoring the crowd size. A major drawback of the holistic approach is the large amount of training data required to train a system; it becomes necessary to annotate a very large number of frames in order to achieve proper generalisation across these various conditions. In a facility containing numerous cameras, for example, it would not be practical to supply hundreds of frames of ground truth for every viewpoint.

In this chapter, a novel algorithm is proposed that uses *local* features, specific to individuals and groups within an image, to estimate the crowd size and its distribution across a scene. While existing techniques have used similar features such as foreground pixels, they are analysed at a holistic level. In this chapter, local features are used to estimate the number of people within each *group*, so that the total crowd estimate is the sum of all group sizes. Because local features are used, training data must also be annotated with local information. A unique method of localised ground truth annotation is therefore proposed, which allows the system to be trained rapidly, and greatly reduces the required training data.

The rationale for using local features is that crowding arrangement can alter the holistic features of an image, despite the total crowd size remaining constant. As Kong [104] observes, “a single pedestrian may occupy 100 pixels. When two pedestrians are together, due to occlusion, there will be less than 200 pixels.” In

other words, the relationship between crowd size and holistic features is modulated by the crowd’s arrangement, and in particular, the size of its groups. The larger a group, the more occlusion is likely to be present, compared to a sparsely populated scene of individuals. Viewed from a holistic perspective, these various crowd distributions can give rise to vastly different image features. Existing techniques cope by extracting a larger quantity of holistic features (for example, 29 features are used by Chan [36] and 129 features are used by Tan [181]), necessitating more training data and more complicated classification strategies. However, it will be shown in this chapter that the relationship between image features and group size is more reliable and consistent when analysed on a local scale. Results comparing the local and holistic approaches are presented in Section 3.3.4.

A localised approach also enables the estimation of crowd densities at different locations *within* the scene (unlike holistic systems, which can only provide a density for the whole scene), and allows for a direct extension to a multi-camera environment as described in Chapter 5. The ability to determine local crowd densities greatly improves the systems ability to detect abnormalities in a scene. While the overall number of people in a scene may be considered normal, there may be an abnormally high concentration of people in a small area. Holistic systems are unable to detect such an abnormality, however the proposed local approach can detect such an occurrence.

The proposed system is tested on five datasets: UCSD [36], PETS 2009 [149], the Mall Dataset [42], the Fudan dataset [181] and the New York Grand Central Station dataset [206]. These datasets feature crowds of size 1 to 245 people, and capture a wide variation in scene properties. The proposed algorithm is compared to several holistic techniques, and is shown to outperform holistic techniques in terms of accuracy, scalability and practicality. The system is shown to be highly scalable, as it is capable of extrapolating to crowd sizes which are smaller or

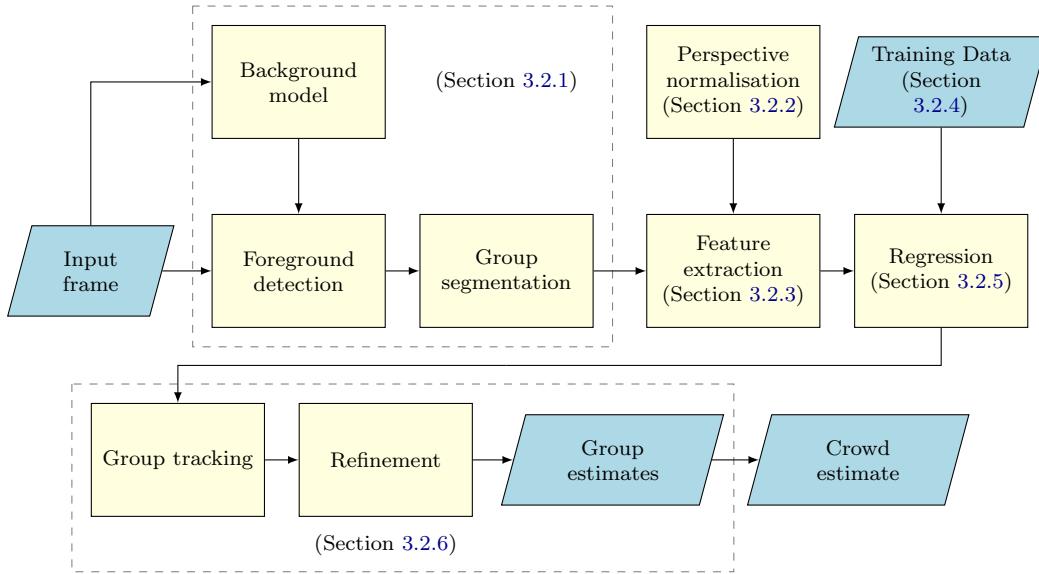


Figure 3.1: Block diagram of the proposed system.

larger than those encountered during training; and highly practical, as it can count crowds when trained on as few as 1 to 10 frames of training data.

The remainder of the chapter is structured as follows: Section 3.2 outlines the proposed algorithm, Section 3.3 presents the datasets and experimental results, Section 3.4 describes the visualisation strategy used to present results to the end-user of the system, and Section 3.5 presents conclusions of this research.

3.2 Crowd Counting using Local Features

The algorithm presented in this chapter is comprised of several modules. These are shown in the block diagram of Figure 3.1. Section 3.2.1 describes the background model and group segmentation algorithm; Section 3.2.2 discusses perspective normalisation; Section 3.2.3 describes the feature extraction process; Section 3.2.4 presents a ground truth annotation strategy that greatly simplifies the training process; Section 3.2.5 discusses crowd modelling and the chosen re-

gression model; and Section 3.2.6 proposes an extension to the system which uses group tracking to refine and improve the group size estimates.

3.2.1 Foreground Detection and Group Segmentation

Background modelling is a fundamental step in many surveillance systems, and it forms the backbone of the proposed algorithm. The background model allows subsequent tasks to be performed, such as foreground detection, group segmentation and feature extraction.

Denman’s cluster based algorithm [59, 60, 61] is chosen for this purpose because it incorporates a lighting model to compensate for illumination changes in a scene. It is an extension of the work of Butler [30, 31], which is an efficient discretised version of Stauffer and Grimson’s popular multimodal background model [178, 179]. The details of these algorithms are discussed in Section 2.2.2.

Following foreground detection, a morphological closing operation is applied to the binary mask in order to obtain cleaner and less fragmented blob segments. Closing consists of a morphological dilation operation, followed by erosion. A 3×3 structuring element kernel is used for these operations on small images (up to 320×240 pixels) while a larger 6×6 structuring element is used for larger images. The use of morphology is motivated by the fact that the proposed algorithm uses these segmentation results as a basis for local feature extraction. A connected components algorithm is run to identify the locations of each blob in an image.

In the proposed crowd counting algorithm, a crowd estimate is obtained for each blob in an image, so that the total estimate for the scene is the sum of the estimates for each individual blob. In order to train the system, ground truth

annotation is performed *after* the first stage of image processing, once the foreground is extracted. The group size is explicitly labeled for each blob in an image, therefore each frame provides several instances of ground truth. The details of this annotation process are discussed in Section 3.2.4.

The advantages of foreground localisation include:

1. **Relevant localisation.** The proposed approach is based on foreground detection, which is directly relevant to the objects of interest in the scene.
2. **Inclusion of stationary pedestrians.** Background models can continue to detect foreground pixels corresponding to pedestrians even after the pedestrian has come to a halt for some time. By contrast, many other localisation strategies only detect pedestrians in motion. For example, these strategies include clusters of moving SURF points [52], KLT tracking of moving corners [151], and dynamic textures [36] which are based on active motion.
3. **Small training requirements.** The quantity of training data required is relatively small: each frame is likely to provide 1-30 foreground blobs with corresponding annotations, depending upon crowd distribution and noise. Holistic methods only yield one training sample per frame, necessitating a larger number of frames to be annotated. By contrast, other localisation strategies may produce excessively large training datasets which can be problematic. For example, the pixel-level localisation strategy used by Lempitsky [110] and the block-based approach of Chen [42] produce a very large quantity of training data (one sample per pixel or block) for every frame. Samples taken from empty parts of the scene may contain redundant information whereas foreground segmentation provides relevant localisation.

Unlike other foreground localisation strategies, the algorithm proposed in this thesis does not place any assumptions on the foreground blob segments, such as an elliptical model [66, 96, 97] or an untrained linear model [33] which would be best suited for sparse crowds. Instead, the machine learning techniques which have typically been used on a holistic level are applied here at the local level, using foreground segmentation as the basis for localisation.

For the remainder of this chapter, the following notation is used in regards to foreground detection and segmentation. The foreground image F , of the same dimensions as the input image I , is defined as follows:

$$F(i, j) = \begin{cases} 0 & \text{if pixel } (i, j) \text{ belongs to the background} \\ 1 & \text{if pixel } (i, j) \text{ belongs to the foreground} \end{cases} \quad (3.1)$$

The *set* of foreground pixels in an image is denoted B :

$$B = \{(i, j) : F(i, j) = 1\} \quad (3.2)$$

The letter B is used as it represents the set of all *blobs* in the foreground mask. A connected components label image, C , is used to label each pixel, $C(i, j)$, with the ID of the blob to which it belongs. The variable n is used to enumerate the blobs, so that the set of pixels belonging to the n th foreground segment is denoted B_n :

$$B_n = \{(i, j) : C(i, j) = n\} \quad (3.3)$$

In set terminology, the collection of blobs $\{B_n\}$ is a *partition* of the set B . This means that the blobs are collectively exhaustive, $B = \cup_n B_n$, and pairwise disjoint:

$$B_n \cap B_m = \emptyset, \forall n \neq m.$$

Before image features can be extracted from the blobs, each pixel is normalised in order to compensate for the effects of perspective. This is discussed in Section 3.2.2.

3.2.2 Perspective Normalisation

As shown in Figure 3.1, feature extraction is performed as a subsequent step to group segmentation, and perspective normalisation is used to compensate for the effects of perspective in the image plane. This is necessary because distant objects appear smaller in an image and thus contribute fewer features than closer objects.

It is therefore necessary that features are normalised appropriately so that the trained system can count objects across the scene accurately. A typical method for perspective normalisation is to construct a *density map*, S , which assigns to each pixel a relative weight: pixels corresponding to distant objects are given a higher weight. This weighting is applied to any features extracted at that location, such as foreground pixels.

Ma [121] introduced a density map which weighted each pixel according to the area it represented on the ground plane. It was calculated based on the coordinates of four points in an image, corresponding to two parallel lines in the real world. Figure 3.2 illustrates the scenario considered by Ma. A roadway is comprised of two lines, P_1P_2 and P_3P_4 , parallel in the real world, which converge on the vanishing point P_v in the image plane at (i_v, j_v) .

The camera is assumed to be oriented horizontally, so that any horizontal line

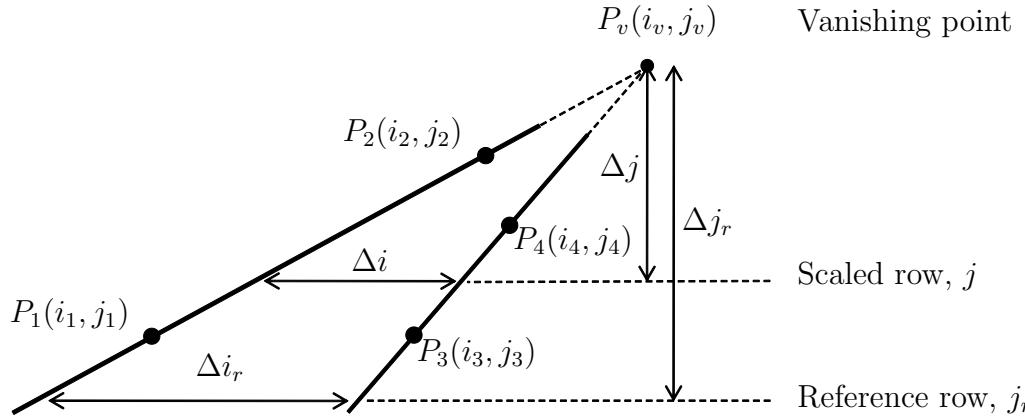


Figure 3.2: Reference lines used by Ma [121] for calculating a density map. P_1P_2 and P_3P_4 correspond to parallel lines on the ground plane.

in the image plane intersects the reference lines at right angles on the ground plane. Therefore the weight assigned to every pixel in any given row of the image is equal.

A *reference row* is selected at j_r , such as the bottom row in the image, to which a density of 1.0 is assigned. The width of the roadway along this reference row is denoted Δi_r , as shown in Figure 3.2, and the vertical distance from the vanishing point is denoted Δj_r .

All other rows of pixels in the image are scaled with respect to this reference row. Consider generally the row j , which is described as a ‘scaled row’ in Figure 3.2. The width of the roadway along this row is denoted Δi , and the vertical distance from the vanishing point is Δj . The scale used by Ma to compensate for perspective is the ratio of the roadway width at the reference row to that of the current row, namely:

$$s(j) = \frac{\Delta i_r}{\Delta i} \quad (3.4)$$

Therefore an object positioned at row j_r appears to be $s(j)$ times wider in the

image plane than an equivalent object positioned at row j . By triangular similarity,

$$s(j) = \frac{\Delta i_r}{\Delta i} \quad (3.5)$$

$$= \frac{\Delta j_r}{\Delta j} \quad (3.6)$$

$$= \frac{j_r - j_v}{j - j_v} \quad (3.7)$$

This scaling factor is used to compensate for perspective in both dimensions, horizontal and vertical, therefore the full scaling factor is:

$$S(i, j) = s(j)^2 = \left(\frac{j_r - j_v}{j - j_v} \right)^2 \quad (3.8)$$

The value of $S(i, j)$ is calculated once for each row in the image, and is used to weight pixels belonging to that row. The calculation of $S(i, j)$ is dependent on the vanishing point's vertical coordinate, j_v . The position of this vanishing point can be calculated from the parallel lines, P_1P_2 and P_3P_4 , annotated by the user during initialisation of the system. The calculation of j_v is straightforward and as described in Appendix A.1.

A similar methodology is employed by Chan [36], who also approximates a density map by linearly interpolating object sizes in the j dimension of the image plane. The width of a walkway, $w(j)$, is interpolated as a function of j , as is the height of a reference person, $h(j)$.

Two horizontal reference lines, ab and cd , at a height of $j = j_1$ and $j = j_2$ respectively, are chosen in the image plane (Figure 3.3), with reference widths $w_1 = |ab|$ and $w_2 = |cd|$. These lines correspond to an equal distance in the real world (in this case, the width of a walkway). At any height j in the image plane,

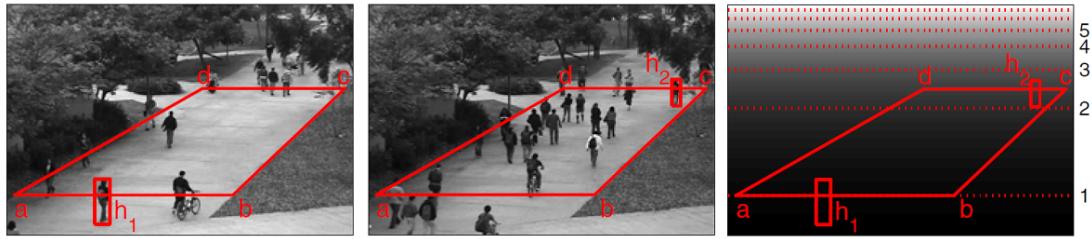


Figure 3.3: Reference lines and pedestrians used by Chan [36] for calculating a density map.

the width of the walkway is denoted $w(j)$, and is linearly interpolated from w_1 and w_2 .

Pixels along ab are given a reference density of 1.0, and all other pixels are weighted relative to this. The scaling factor for object width is therefore defined:

$$s_w(j) = \frac{w_1}{w(j)} \quad (3.9)$$

Thus an object positioned at row j_1 appears to be $s_w(j)$ times wider in the image plane than an equivalent object positioned at row j .

Similarly, a reference pedestrian is selected, with heights h_1 and h_2 when the person is at ab and cd (j_1 and j_2), respectively. The height of the reference person at any height j is denoted $h(j)$, and is linearly interpolated from h_1 and h_2 . As before, pixels along ab are given a reference density of 1.0. The scaling factor for object height is therefore defined:

$$s_h(j) = \frac{h_1}{h(j)} \quad (3.10)$$

Thus an object positioned at row y_1 appears to be $s_h(j)$ times higher in the image plane than an equivalent object positioned at row j .

This scaling factor is used to compensate for perspective in two dimensions, both horizontal and vertical, so that the full scaling factor using Chan's method [36] is:

$$S(i, j) = s_h(j)s_w(j) \quad (3.11)$$

$$= \frac{h_1 w_1}{h(j)w(j)} \quad (3.12)$$

(Note that this is an alternative to the density map proposed by Ma [121], as described in Equation 3.8.) This calculation of $S(i, j)$ is dependent on $w(j)$ and $h(j)$, which is the interpolated width of the walkway and interpolated height of the reference pedestrian. These values can be easily calculated from the user-annotated reference values, w_1 , w_2 , h_1 and h_2 , as described in Appendix A.2.

Both Ma and Chan use linear interpolation in the j coordinate to approximate object sizes, so that the weight assigned to a pixel is inversely proportional to the estimated object size at this location. These approaches rest on the assumption that the camera is oriented horizontally, and are based on simple annotations such as pedestrian height, which can vary from person to person or even frame to frame depending on that person's position, posture and orientation. Additionally, the use of a road or walkway with parallel edges may not be available in all scenes.

Consider for example the PETS 2009 dataset introduced at [149], as shown in Figure 3.4. There are no parallel lines in this scene of sufficient length to calculate a calibration. There are, however, a large number of pedestrian instances in the dataset which could be detected automatically using a pedestrian detector such as Dalal's histogram of oriented gradients [57] or Felzenszwalb's part-based models [67, 68]. Although not all humans will be detected with this approach, especially when the scene becomes crowded, a subsample of pedestrian annotations is sufficient to ascertain the typical variation of object sizes in the image. This

could then be used to calculate an density map, in a manner similar to Ma or Chan, using the dimensions of the pedestrian detections as a basis. Additionally, this approach does not require any manual annotations to be performed by the user or for any real-world measurements to be taken from the scene.

An alternative method for computing a density map is proposed based on these automatic pedestrian detections. A pedestrian *template*, $R_{i,j}$, is used to represent the set of pixels occupied by a pedestrian centered at pixel (i, j) . The simplest model of a pedestrian is a rectangle in the image plane, however alternative templates can also be used such as a 3D cylinder model.

Pedestrians are detected automatically from a set of training images using discriminatively trained part-based models [68]. These detections are each represented by a rectangular bounding box, which we can use as our template. The p th detection is of width w_p and height h_p , centered around pixel (i_p, j_p) . A large set of detections from the PETS 2009 dataset [149] are overlaid on a single image in Figure 3.4(a).

A linear model is used to model the dimensions of the pedestrian template, $w \times h$, as a function of its centroid's position in the image plane (i, j) . The set of detections enumerated by p , namely $\{i_p, j_p, w_p, h_p\}$, is used as training data for this model. Due to the approximate nature of an object detector's output, a robust linear fitting algorithm is used to reduce the influence of outliers. The iteratively reweighted least squares (IRLS) regression algorithm is used to fit the linear model, as described in Appendix A.3. This model is described by:

$$w = b_w j + c_w \quad (3.13)$$

$$h = b_h j + c_h \quad (3.14)$$

where $\{b_w, c_w, b_h, c_h\}$ denote the linear weights, j is the vertical coordinate of the point around which the template model is to be positioned, and $\{w, h\}$ denote the dimensions of the template model (as a function of j). Note that this model is independent of i , in keeping with the assumptions of Chan and Ma. Figure 3.4(b) depicts the pedestrian templates positioned at various locations in the image plane, for illustrative purposes, which are projected using Equations 3.13 and 3.14, at fixed intervals.

Alternatively, we may relax the assumption that the camera is positioned horizontally, and instead use a linear model dependent upon both i and j :

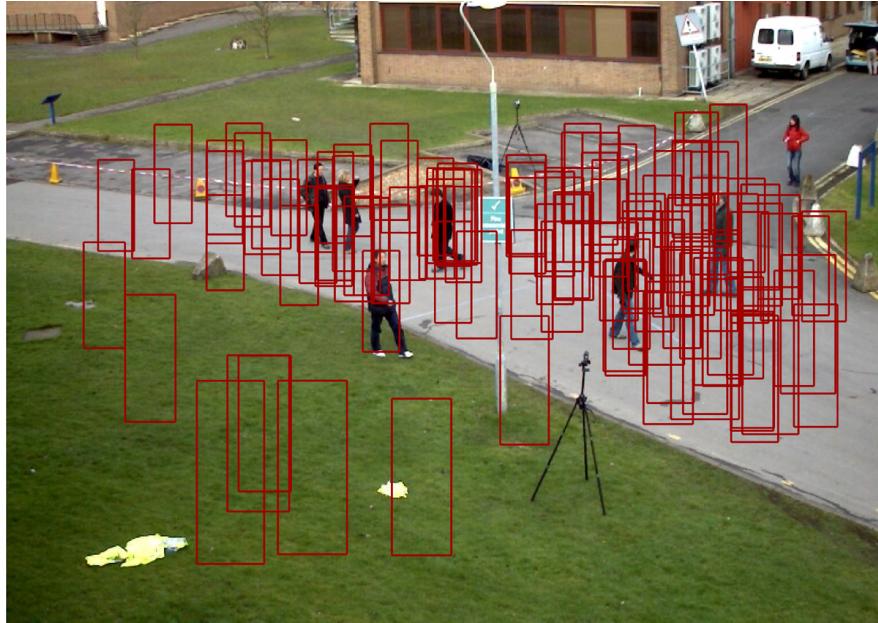
$$w = a_w i + b_w j + c_w \quad (3.15)$$

$$h = a_h i + b_h j + c_h \quad (3.16)$$

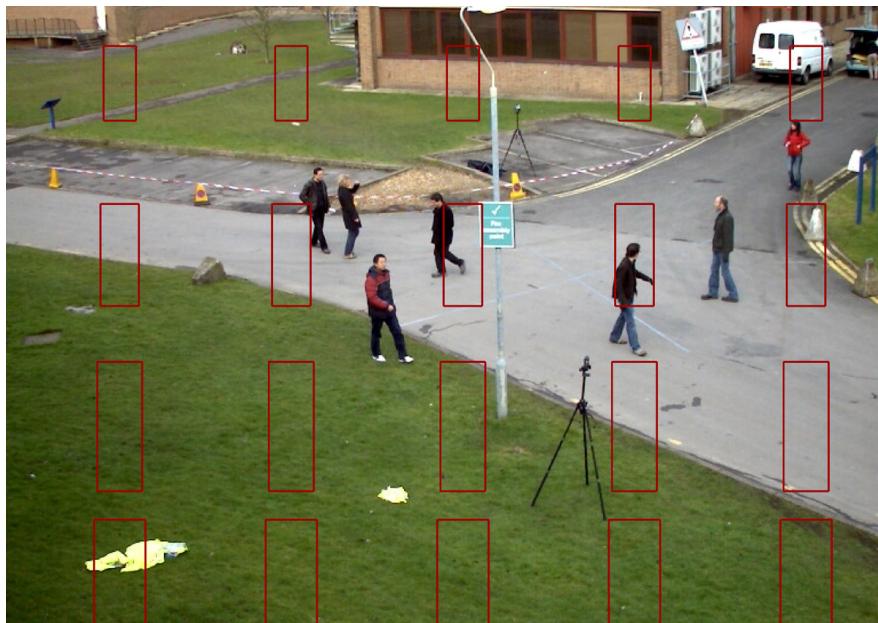
Given that most pedestrian detection algorithms assume that pedestrians in an image appear vertical, however, this model is most suitable when a camera is oriented on only a slight angle.

Note that this approach is based on pedestrian detection, which requires at least a few sparse (unoccluded) individuals to be present in a video sequence. Even in crowded environments there are usually a number of individuals who appear unoccluded at various points in the sequence, and this is sufficient for detection. Alternatively, it is straightforward to manually annotate a set of pedestrian bounding boxes and use this set of $\{i_p, j_p, w_p, h_p\}$ instead.

Another pedestrian template that can be considered is the 3D cylinder model [34]. This template is a 2D projection of a cylinder onto the image plane, with the center of the model coinciding roughly with the center of a pedestrian standing at that location. This approach requires camera calibration parameters to be



(a) All pedestrian detection bounding boxes from a video sequence overlaid on a single image from the PETS 2009 dataset [149].



(b) Projected pedestrian templates overlaid at fixed intervals on a single image using IRLS linear regression.

Figure 3.4: Pedestrian detections from the PETS 2009 dataset [149] and pedestrian template models.

known or estimated from real world measurements, and is discussed further in Section 4.2.2.

Given the pedestrian template $R_{i,j}$ centered around pixel (i, j) , the area of the template is denoted:

$$A = |R_{i,j}| \quad (3.17)$$

which is equal to $w \times h$ in the case of a rectangular model. The values in the density map can then be computed in a similar manner as Chan [36]. A reference pixel (i_r, j_r) is selected near the bottom of the image, whose pedestrian template is of area $A_r = w_r h_r$. The value of the density map at any location in the image is then taken to be inversely proportional to the area of the template:

$$S(i, j) = \frac{A_r}{A} \quad (3.18)$$

which, for a rectangular model, is equal to:

$$S(i, j) = \frac{w_r h_r}{w(j) h(j)} \quad (3.19)$$

The value of the density map at the reference pixel is 1.0 (although this designation is arbitrary).

The proposed approach is similar to Ma [121] and Chan [36], although the following differences are noted:

1. The proposed method is automated and does not require any manual annotations from the user.

2. This method does not require a roadway or a set of parallel lines to be readily apparent in the image.
3. The estimation of a pedestrian template $R_{i,j}$, and its size at any position (i, j) in the image, is based on the aggregate of a large number of pedestrian bounding boxes, which are obtained using a pedestrian detector [68], rather than a single instance of a pedestrian moving through the scene who might be subject to incidental variations in size from one frame to the next due to limb movement and orientation.

Furthermore, the pedestrian template $R_{i,j}$ is also used for ground truth annotation and system training, as described in Section 3.2.4. In fact, the use of the pedestrian template is an important part of the annotation process and simplifies training greatly. It is therefore intuitive to use it for constructing the density map as well.

3.2.3 Feature Selection

The density map described in Section 3.2.2 is used to weight the features extracted at each pixel in an image. The features may include foreground pixels, edge pixels and other keypoints of interest which may be indicative of crowding.

The number and quality of features extracted has a direct impact on the subsequent regression or classification stage. For example, higher dimensionality of the feature vector leads to a more complex classifier and an increased training data requirement. Depending on the classifier or regression model, computation time may also be increased. Consequently it is important to use accurate features which capture the level of crowding within a scene.

In this section, image feature selection is discussed and a range of features are proposed for crowd counting. The empirical performance of these features is assessed quantitatively in Section 3.3.3. In general, the features used for crowd counting can be categorised under the following headings:

1. **Texture** refers to the holistic descriptors of an image such as contrast and homogeneity. (Section 3.2.3.1).
2. **Size** refers to the magnitude of any interesting segments extracted from an image which are deemed to be relevant, such as the foreground pixel count. (Section 3.2.3.2).
3. **Shape** pertains to the orientation and shape descriptors of these areas, segments or objects detected in an image. (Section 3.2.3.3).
4. **Edge** refers to the relative change in pixel intensities across an image, and this is typically measured by means of a binary edge detector. (Section 3.2.3.4).
5. **Keypoints** are any other points of interest, such as corners, that are detected in an image. (Section 3.2.3.5).

These features and their value for predicting crowd size are discussed in subsequent sections. Finally, Section 3.2.3.6 discusses the *full* proposed feature vectors to be used in this system.

3.2.3.1 Texture

Texture is a holistic description of a scene, and therefore cannot intuitively be weighted pixelwise by a density map to compensate for perspective. Textural features which have been used in the literature include:

1. GLCM based features (Marana [128]).
2. Minkowski fractal dimension (Marana [129]).
3. Image moments (Rahmalan [153]).
4. Wavelet transform based features (Xiaohua [197]).

These features are primarily used for density *classification*, where crowd densities are measured using a four or five point scale. Less commonly, these features have also been used for direct counting via *regression*. For example, Chan [36] has proposed the use of holistic image features, including textures, for regression. However, there are two important considerations to take into account with regard to this approach:

1. Chan proposed a total of 29 features, including motion segmentation and edge detection, in addition to the textures. These additional features have been shown by numerous studies to play a significant role in the success of crowd counting systems [58, 103, 121]. The system proposed by Chan did not exclusively use textures and it is therefore unclear what role they played in the counting accuracy.
2. The testing protocol used by Chan did not evaluate the *long term* performance of the algorithm over a significant period of time containing a wide range of conditions, such as illumination changes, which have been shown to impact texture-based crowd counting negatively (Section 2.3.1; [153]).

This latter point is significant because holistic textures can be affected by lighting and environmental changes, whereas an adaptive background model can incorporate such changes, within reason, in order to achieve accurate foreground segmentation.

The UCSD dataset introduced by Chan [36] consisted of 33000 video frames, the first 2000 of which are annotated with ground truth information (i.e. the location of each pedestrian and the total count for each frame); and Chan’s analysis was performed exclusively on these 2000 frames. At a frame rate of 10 fps, these annotated frames cover only 200 seconds of crowd movements and fluctuations. This is unlikely to be of sufficient length to observe small environmental changes in the scene.

However, the full UCSD dataset covers a total period of almost one hour. As such, the full dataset may be used to quantitatively analyse the performance of textural features over time by using frames at each extreme end of the dataset. In this section, long term performance of textural features are evaluated using the UCSD dataset. More information about this dataset is provided in Section 3.3.2.

Frames 601-1400 are used for training, and testing is performed on three frame ranges at various times as specified in Table 3.1. For these testing ranges, a subset of frames were selected and annotated with ground truth (crowd size). The following textural features were considered:

1. **Contrast** at 0° , 45° , 90° and 135° (Equation 2.32, p. 39).
2. **Homogeneity** at 0° , 45° , 90° and 135° (Equation 2.33, p. 39).
3. **Energy** at 0° , 45° , 90° and 135° (Equation 2.34, p. 39).

These twelve features provide holistic description of the textures present in an image. A linear regression model is used to learn the relationship between this feature space and holistic crowd size, and performance is evaluated on each of the testing ranges. Table 3.1 shows the accuracy of this approach in terms of

Frame range	Description	MAE	MSE	MRE
601 : 1400	Training range	1.60	4.80	6.78%
1401 : 2000	Testing range 1	2.88	11.59	10.97%
23000 : 24000	Testing range 2	4.79	25.24	62.70%
30000 : 32000	Testing range 3	3.65	15.35	136.03%

Table 3.1: Frame ranges used in this analysis, on the UCSD dataset [36], and corresponding performance using **holistic textures** only.

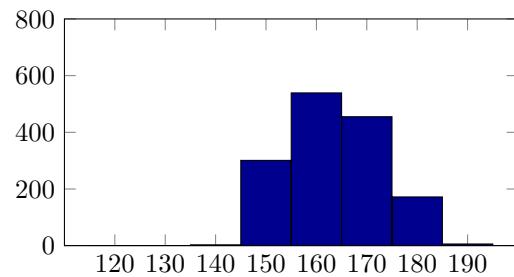
mean absolute error ('MAE'), mean square error ('MSE') and mean relative error ('MRE') for each time period. (More information about these metrics can be found in Section 3.3.1). This approach performs well when tested on the frames in the training range, as one would expect. It also performs reasonably well on 'Testing range 1' which immediately follows the training range in the video sequence. However, testing ranges 2 and 3 occur some duration after the training range, after which sufficient time has passed for subtle changes in the environment to affect the relationship between crowd size and image texture. Consequently the accuracy of the method is poor in these ranges.

In particular, the brightness of the image is decreased slightly between frames 1 and 30000 (Figure 3.5). Although difficult to perceive to the human eye, these subtle environmental changes are often significant and alter the statistical properties of the image. For example, Figure 3.5(b) depicts the histogram of pixel intensities of a small subregion in Frame 1 of the UCSD dataset. The mean and variance of the intensity is $\mu = 163.8$ and $\sigma^2 = 82.5$, respectively. The equivalent subregion in frame 30000 has mean $\mu = 152.1$ and variance $\sigma^2 = 43.2$ (Figure 3.5(d)). These statistical alterations are indicative of an environmental change.

Figure 3.6 plots the estimated crowd size against the ground truth for each frame range. These plots indicate that the estimate has 'drifted' dramatically from the ground truth after some time has passed (testing ranges 2-3).



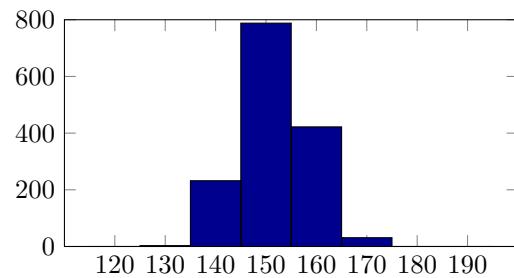
(a) Frame 1 with subregion indicated



(b) Histogram of pixel intensities within subregion (Frame 1)



(c) Frame 30000 with subregion indicated



(d) Histogram of pixel intensities within subregion (Frame 30000)

Figure 3.5: Comparison of frames 1 and 30 000. A subregion of interest is selected on the walkway and a histogram of pixel brightness for each frame is shown.

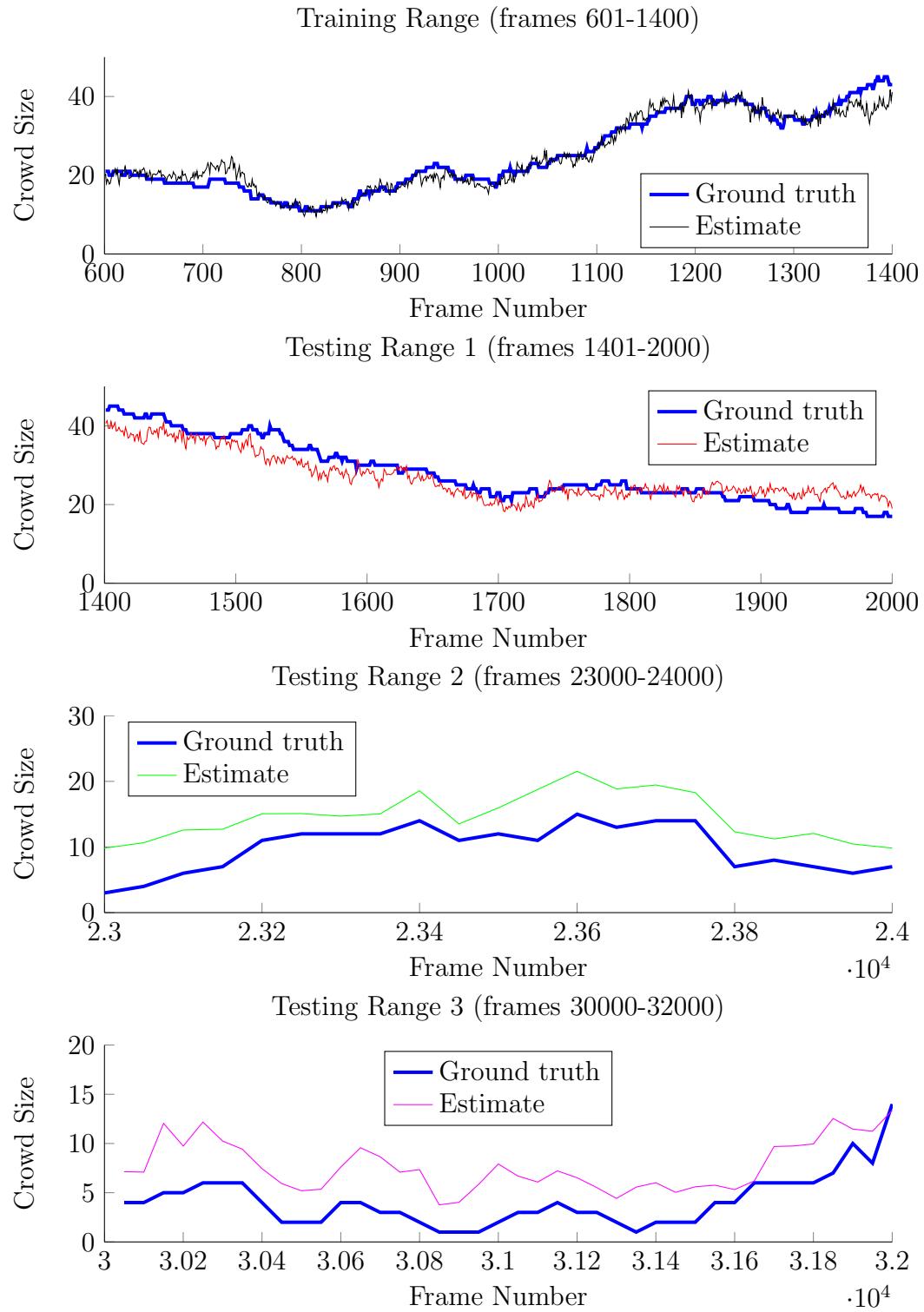


Figure 3.6: Performance of a texture-based holistic crowd counting system over time. These plots indicate a ‘drift’ between estimate and ground truth.

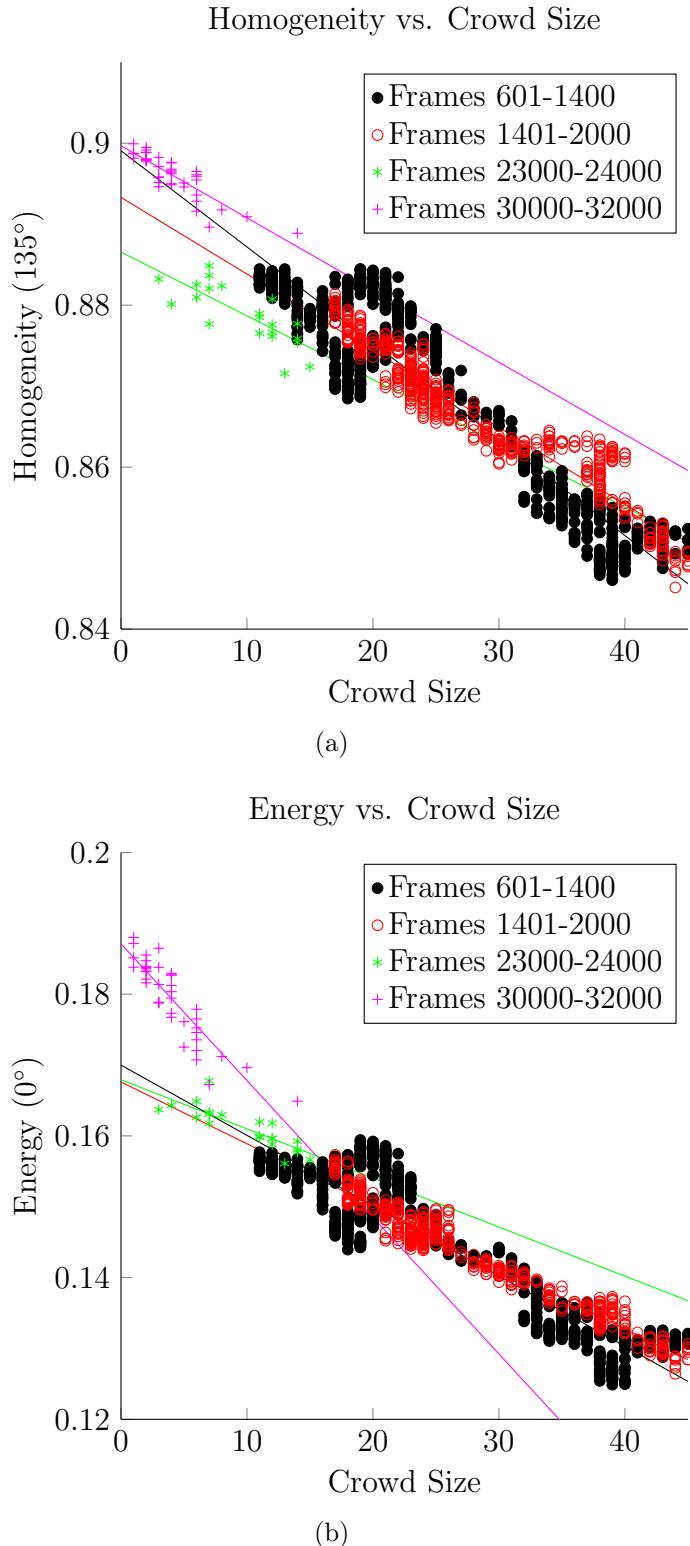


Figure 3.7: Relationship between crowd size and textural features at different time periods. Trend lines are plotted for each group of points to aid interpretation.

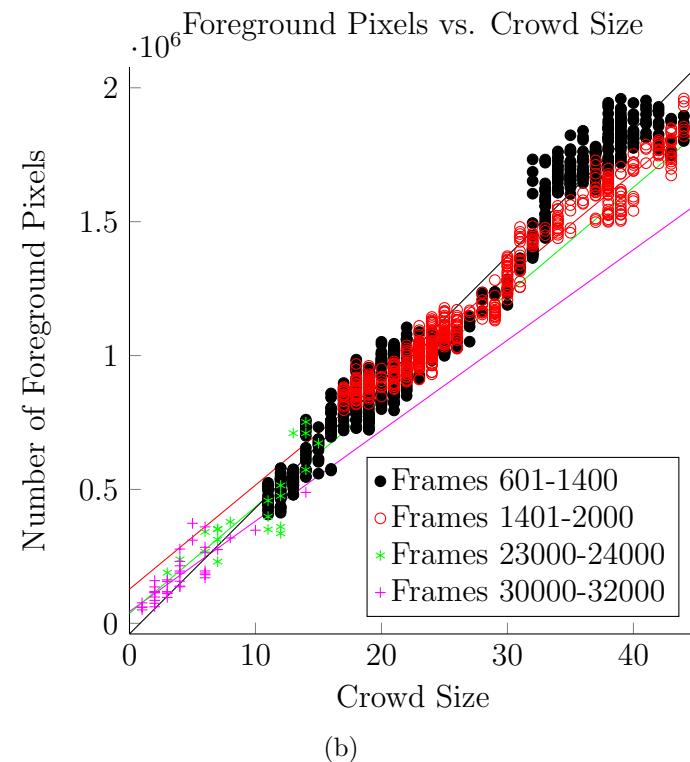
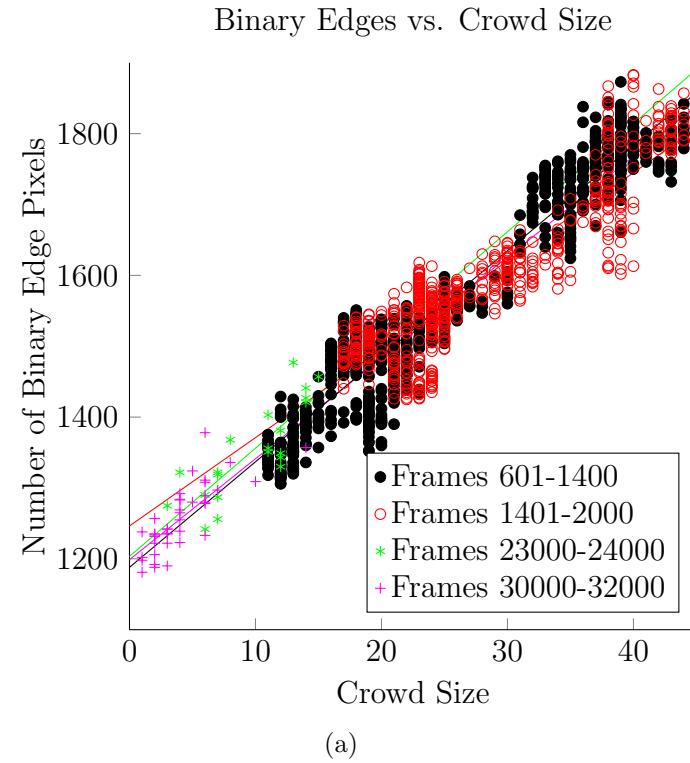


Figure 3.8: Relationship between crowd size and edge/foreground pixels.

The relationship between crowd size and textural features are shown in Figure 3.7. The points on these scatter plots are grouped by time period to provide a clear indication of how the relationship has changed over time. For example, the black and red points in Figure 3.7(a), corresponding to frames 601-1400 and 1401-2000 respectively, occupy a similar region of the plot. This indicates that the relationship between crowd size and homogeneity remains relatively unchanged after a short period of time. It is also noteworthy that the relationship appears to be approximately linear, validating the use of a linear regression model in this analysis. However, the green points corresponding to frames 23000-24000 indicate that relationship has changed over time. This is also true for the magenta points in Figure 3.7(b) which correspond to frames 30000-32000, indicating that the relationship between crowd size and energy has changed (in this case, quite significantly).

For comparison, Figure 3.8 displays the equivalent scatter plots for binary edge detection using a Sobel operator and foreground detection using Denman's adaptive background model [59, 60, 61]. These plots suggest that the relationship remains relatively unchanged over time for edge and foreground pixel counts.

We conclude that the use of image textures is unlikely to be reliable for crowd counting over the long term due to subtle environmental fluctuations.

3.2.3.2 Size

Size refers to the magnitude of any detected regions, such as motion segments, in an image. Davies [58] proposed the use of the foreground pixel count as a measure of the holistic crowd size, while Ma [121] introduced the density map to weight each pixel to compensate for perspective.

The set of foreground pixels within the region of interest is denoted B , such that the number of foreground pixels is $|B|$. The *weighted area* of the foreground is denoted A . This is calculated using the density map, S , as defined in Section 3.2.2:

$$A = \sum_{(i,j) \in B} S(i,j) \quad (3.20)$$

This area directly captures the size of the foreground normalised for perspective. In practice, the presence of occlusions will lead to non-linearities in the relationship between crowd size and weighted foreground. This is shown in Figure 3.8: while the relationship is approximately linear, there are deviations from the regression line when the foreground is more or less than expected. This is usually due to occlusion or segmentation errors. Other features such as gradients (Section 3.2.3.4) and keypoints (Section 3.2.3.5) are also employed to improve prediction in the presence of occlusion, as these features are more prominent in regions of object overlap.

Another strategy proposed in this thesis is to segment the foreground into a set of connected components, which are individually labelled, and enumerated by n . The notation B_n is used to represent the set of pixels which belong to the n th blob. In set terminology, the collection of blobs $\{B_n\}$ is a *partition* of the set B . The weighted area of each blob, A_n , is:

$$A_n = \sum_{(i,j) \in B_n} S(i,j) \quad (3.21)$$

Note that $A = \sum_n A_n$ because the holistic foreground area is the sum of its segmented parts. The proposed advantages of local features were described in the introduction to this chapter (Section 3.1), and with regards to blob size

specifically, localisation enables the system to differentiate between small and large groups, as well as noise, whereas the holistic area A merely aggregates the group sizes.

Another size feature is perimeter length. The set of perimeter pixels P_n is obtained by tracing along the boundary of the n th blob, and the set of all perimeter pixels in an image is denoted $P = \cup_n P_n$. Perimeter pixels are a one-dimensional feature, and are thus weighted using the square root of the density map S . The weighted perimeter of the n th blob segment is therefore calculated as follows:

$$L_n = \sum_{(i,j) \in P_n} \sqrt{S(i,j)} \quad (3.22)$$

And the holistic perimeter length feature is:

$$L = \sum_{(i,j) \in P} \sqrt{S(i,j)} \quad (3.23)$$

$$= \sum_n L_n \quad (3.24)$$

The perimeter length may provide valuable size information when the foreground segments erroneously contains ‘holes’. It also supplements the area feature to provide a more complete description of group size.

3.2.3.3 Shape

Perimeter pixels provide valuable shape information about an object. Aside from the perimeter length, which measures the object *size*, the orientation of the perimeter pixels also contain important shape information.

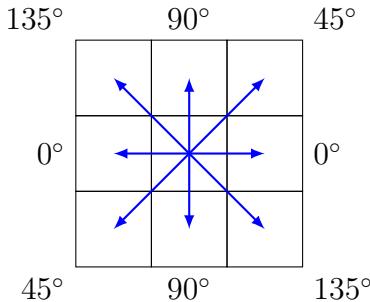


Figure 3.9: Four directions used for computing perimeter orientation histogram.

For example, Dong [63] encoded a blob's approximate shape using Fourier descriptors, discarding high frequency coefficients as these contribute little to the overall blob shape (Figure 2.13, p. 62). Dong found that $M = 7$ Fourier coefficients provided a sufficient representation of object shape, and these seven coefficients formed the feature vector describing the blob. K Nearest Neighbours (KNN) regression was used to estimate the group size, based on a training dataset of examples; however, this approach quickly became impractical for groups larger than 1-6 people.

Chan [35, 36] uses several features including a perimeter orientation histogram with 6 bins. The perimeter orientation is determined by filtering the image with six Gaussian kernels (oriented at evenly spaced angles), and selecting the filter which produces the maximum response at each perimeter pixel. Every perimeter pixel contributes a vote to a histogram bin depending on its orientation.

Perimeter pixels are easily detected by tracing around the boundary of an object given an initial seed point on the perimeter. It is therefore intuitive and computationally efficient to use an orientation histogram with 4 bins, each corresponding to the direction of an adjacent pixel ($0^\circ, 45^\circ, 90^\circ, 135^\circ$), as shown in Figure 3.9. When tracing the perimeter from one boundary pixel to the next, the direction of movement determines which histogram bin receives the pixel's vote.

The vote weight is the square root of the density map, $\sqrt{S(i,j)}$, as perimeter pixels are a one dimensional feature. Vertical edges in the absence of horizontal features are more likely to indicate individuals in a scene, whereas a combination of many perimeter pixels at all orientations may indicate larger crowds.

The value stored in each histogram bin h constitutes a feature, and the four shape features are denoted $V_n(h)$, for $h \in [0, 3]$. The equivalent holistic features are:

$$V(h) = \sum_n V_n(h) \quad (3.25)$$

This is simply the sum of the local perimeter orientation features, taken at a holistic level.

3.2.3.4 Edges

Edge features are calculated from the relative change in pixel intensities across an image. A horizontal kernel, L_x , and vertical kernel, L_y , are used to calculate the gradients using convolution with an image, I . Common kernels include central differences, $L_x = [-1, 0, 1]$ and $L_y = [-1, 0, 1]^T$, or the Sobel kernels:

$$L_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad (3.26)$$

$$L_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad (3.27)$$

The gradients are calculated by convolving the image I with the kernels:

$$G_x = L_x * I \quad (3.28)$$

$$G_y = L_y * I \quad (3.29)$$

where $*$ denotes convolution, and the horizontal and vertical components of the gradient are stored in G_x and G_y respectively. The gradient $G(i, j)$ may therefore be considered a vector. The magnitude of the gradient at pixel (i, j) is denoted $|G(i, j)|$, and is calculated as follows:

$$|G(i, j)| = \sqrt{G_x(i, j)^2 + G_y(i, j)^2} \quad (3.30)$$

The gradient is *oriented* at an angle of $\angle G(i, j)$, which is calculated as follows:

$$\angle G(i, j) = \arctan \left(\frac{G_y(i, j)}{G_x(i, j)} \right) \quad (3.31)$$

The atan2 function is used to return angles in the range $(-\pi, \pi]$, which constitutes the full range of 2π radians. However, the only difference between a gradient oriented at θ and $\theta + \pi$ is whether the pixel intensity has changed from light to dark or vice versa. In practice we are uninterested in this; it only matters that the value has changed. (Whether a person is lighter or darker than the background is irrelevant, but in any event, they tend to produce vertical edges in the image.) Consequently, we consider orientations of θ and $\theta + \pi$ to be equivalent, as shown in Figure 3.9. Therefore negative angles are mapped to the range $(0, \pi]$ by adding π , resulting in a final range of $\angle G(i, j) \in [0, \pi]$.

Binary edge detection usually involves applying a threshold to the gradient mag-

nitude, for example:

$$E(i, j) = \begin{cases} 1 & |G(i, j)| > T_{\text{edge}} \\ 0 & \text{otherwise} \end{cases} \quad (3.32)$$

This is often followed by thinning or linking as in the Canny edge detection algorithm [32]. Canny edge detection is used in the proposed method due to its use of non-maximum suppression and hysteresis thresholding which results in a cleaner output. The resulting binary image E labels edges with the value 1, and all other pixels 0.

Edges have been commonly used in crowd counting systems. For example, Kong [103] introduced the use of an edge angle histogram on a holistic scale, while Dalal [57] introduced the histogram of oriented gradients (HOG) for person detection. Davies [58], Chan [37] and many others have used the total number of edge pixels on a holistic level, regardless of orientation.

In the proposed algorithm, an edge orientation histogram is constructed for each foreground segment in an image using the following procedure. For the n th blob segment, a histogram of edge orientations E_n is constructed by allocating each edge pixel to a histogram channel, based on the pixel's orientation $\angle G(i, j)$. The orientation bins are evenly divided over the range $[0, \pi]$, and a total of 6 bins are used. Each edge pixel within the blob contributes a weighted vote to a histogram bin. This contribution (or vote) is equal to the square root of the density map, $\sqrt{S(i, j)}$, to normalise for perspective. If the value of the h th histogram bin is

denoted $H_n(h)$, and the orientation angle for that bin is lower-bounded by θ_h :

$$H_n(h) = \sum_{(i,j) \in B_n} \begin{cases} \sqrt{S(i,j)} & \text{if } \theta_h \leq \angle G(i,j) < \theta_{h+1} \text{ and } E(i,j) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.33)$$

The edge orientation histogram is used to help distinguish between humans and other structures in the scene [103]. Strong vertical edges tend to be most indicative of human crowding, while horizontal edges are indicative of it to a lesser extent. Edges also help to identify occlusions when multiple pedestrians partially block one another from view. Although the blob's *size* features are reduced by occlusions, the edge features become stronger due to the overlapping body parts, differing skin tones and conflicting clothing.

At the holistic level, the edge orientation histogram is calculated as follows:

$$H(h) = \sum_n H_n(h) \quad (3.34)$$

3.2.3.5 Keypoints

Keypoints refer to specific pixels of interest, such as corners, which are detected in an image. Keypoints are useful for detecting salient points of interest in a scene, and these are often indicative of human crowding. For example, Conte [55] used speeded-up robust features (SURF), as introduced by Bay [18, 19], to detect keypoints within an image. These points were masked by optical flow so that stationary points were ignored. The number of moving keypoints was used to predict crowding. Similarly, Albiol [6] utilised Harris corners [81] to estimate crowd size on a holistic level.

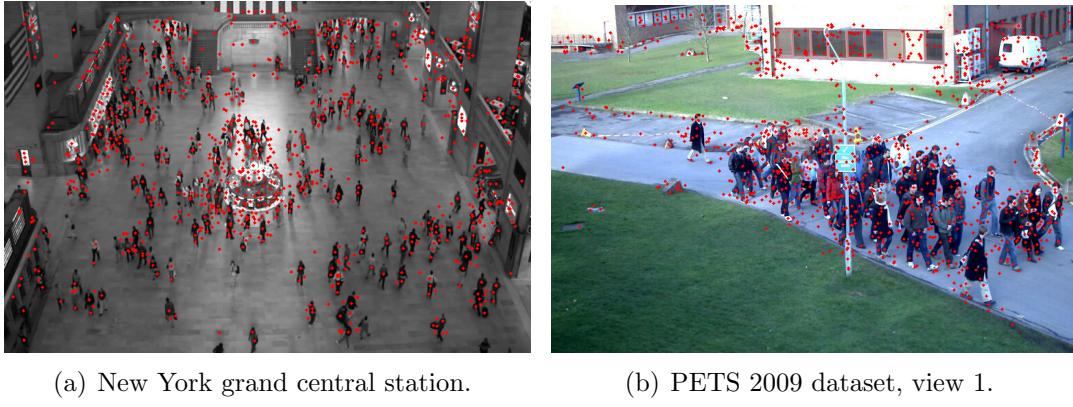


Figure 3.10: Keypoint feature extraction using Speeded-Up Robust Features (SURF) [18]. Keypoints are indicated with red dots.

Two types of feature detectors are considered for the proposed algorithm. Firstly, corners are detected using the ‘FAST’ algorithm recently proposed by Rosten [160], and the set of keypoints detected within the foreground blob segment n is denoted κ_n^{FAST} . Secondly, SURF keypoints [18] are extracted and this set of keypoints is denoted κ_n^{SURF} .

Figure 3.10 shows the features detected using SURF on the New York grand central station and PETS 2009 datasets. In the proposed algorithm, these keypoints are masked by the foreground detection result (the summation only takes place across each foreground segment), so that keypoints belonging to background objects and surrounding structures are not included in the feature vector.

The two keypoint features are calculated as follows:

$$K_n^{FAST} = \sum_{(i,j) \in \kappa_n^{FAST}} \sqrt{S(i,j)} \quad (3.35)$$

$$K_n^{SURF} = \sum_{(i,j) \in \kappa_n^{SURF}} \sqrt{S(i,j)} \quad (3.36)$$

Note that the notation κ_n^{FAST} is used to refer to a *set* of FAST keypoints, while K_n^{FAST} represents the scalar keypoint feature that is calculated from this set. (Similarly, κ_n^{SURF} and K_n^{SURF} represent the *set* of SURF keypoints and the *scalar* keypoint feature, respectively.)

In Equations 3.35 and 3.36, keypoints are weighted by $\sqrt{S(i, j)}$ rather than $S(i, j)$. This is motivated by the observation that keypoints behave more like sparse 1D features (edges and perimeter pixels) than 2D features (area). To validate the use of $\sqrt{S(i, j)}$, both weighting strategies were compared using the first 2000 frames of the UCSD crowd counting dataset [36]. Figure 3.11 depicts the relationship between the group size and each keypoint feature (using both weighting strategies). The Pearson correlation coefficient ρ is calculated to measure the strength of the linear correlation between group size and each feature.

For the FAST keypoint feature, the correlation coefficient was $\rho = 0.9889$ with the $\sqrt{S(i, j)}$ weighting strategy, and $\rho = 0.9813$ with $S(i, j)$. There is negligible difference between the two strategies, with a slightly stronger linear correlation observed when $\sqrt{S(i, j)}$ is used. This is also true for the SURF keypoint features, with $\rho = 0.9661$ and $\rho = 0.9563$ for the two strategies ($\sqrt{S(i, j)}$ and $S(i, j)$, respectively). Therefore $\sqrt{S(i, j)}$ is adopted as the keypoint weighting strategy in Equations 3.35 and 3.36.

3.2.3.6 Proposed Features

In order to determine the best types of features for crowd counting, various feature vectors are evaluated. The features are categorised under four different labels (size, shape, edges and keypoints), as shown in Table 3.2, and different combinations of these labels are tested in Section 3.3.3. Note that textures are not used due to evaluation presented in Section 3.2.3.1. The *full* feature vector for the n th

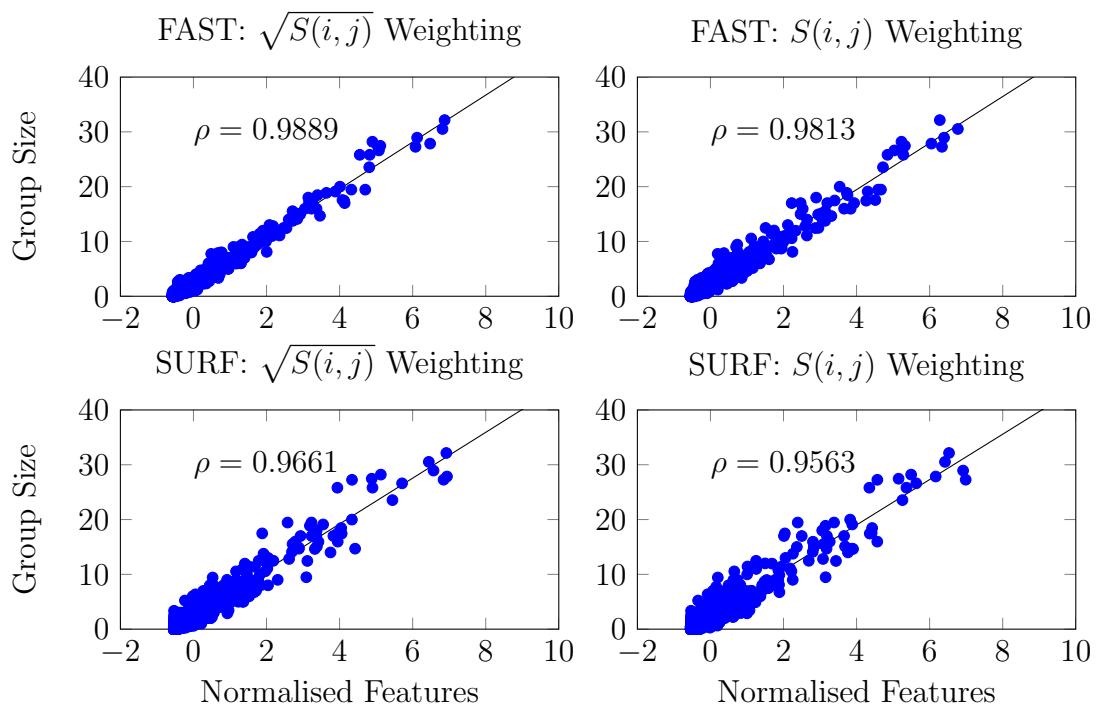


Figure 3.11: Relationship between keypoint features and group size using two different weighting strategies: $\sqrt{S(i,j)}$ and $S(i,j)$.

Category	Local Features	Holistic Features	Description
Size	A_n, L_n	A, L	Area and perimeter length
Shape	$V_n(0) \dots V_n(3)$	$V(0) \dots V(3)$	Perimeter orientation histogram
Edges	$H_n(0) \dots H_n(5)$	$H(0) \dots H(5)$	Edge orientation histogram
Keypoints	K_n^{SURF}, K_n^{FAST}	K^{SURF}, K^{FAST}	SURF and FAST keypoint features

Table 3.2: Feature categories used for crowd counting in this thesis. The subscript n indicates the index of the blob under consideration, although an equivalent holistic feature is represented by omitting the subscript, as shown in Equation 3.24 (p. 126), for example.

blob is denoted:

$$\mathbf{x}_n = [A_n, L_n, V_n(0), \dots, V_n(3), H_n(0), \dots, H_n(5), K_n^{SURF}, K_n^{FAST}] \quad (3.37)$$

The value of each feature is evident in Figure 3.12, which shows the relationship between each normalised feature and the corresponding group size within the UCSD dataset.

Before the system can be tested, however, it must first be trained, and this procedure is described in the subsequent section.

3.2.4 System Training

The proposed algorithm is designed to count the number of people in each foreground segment. Feature extraction and regression is therefore performed at the local level. The system is trained by annotating the number of people occupying each blob in a training dataset. A straightforward implementation of this procedure is as follows:

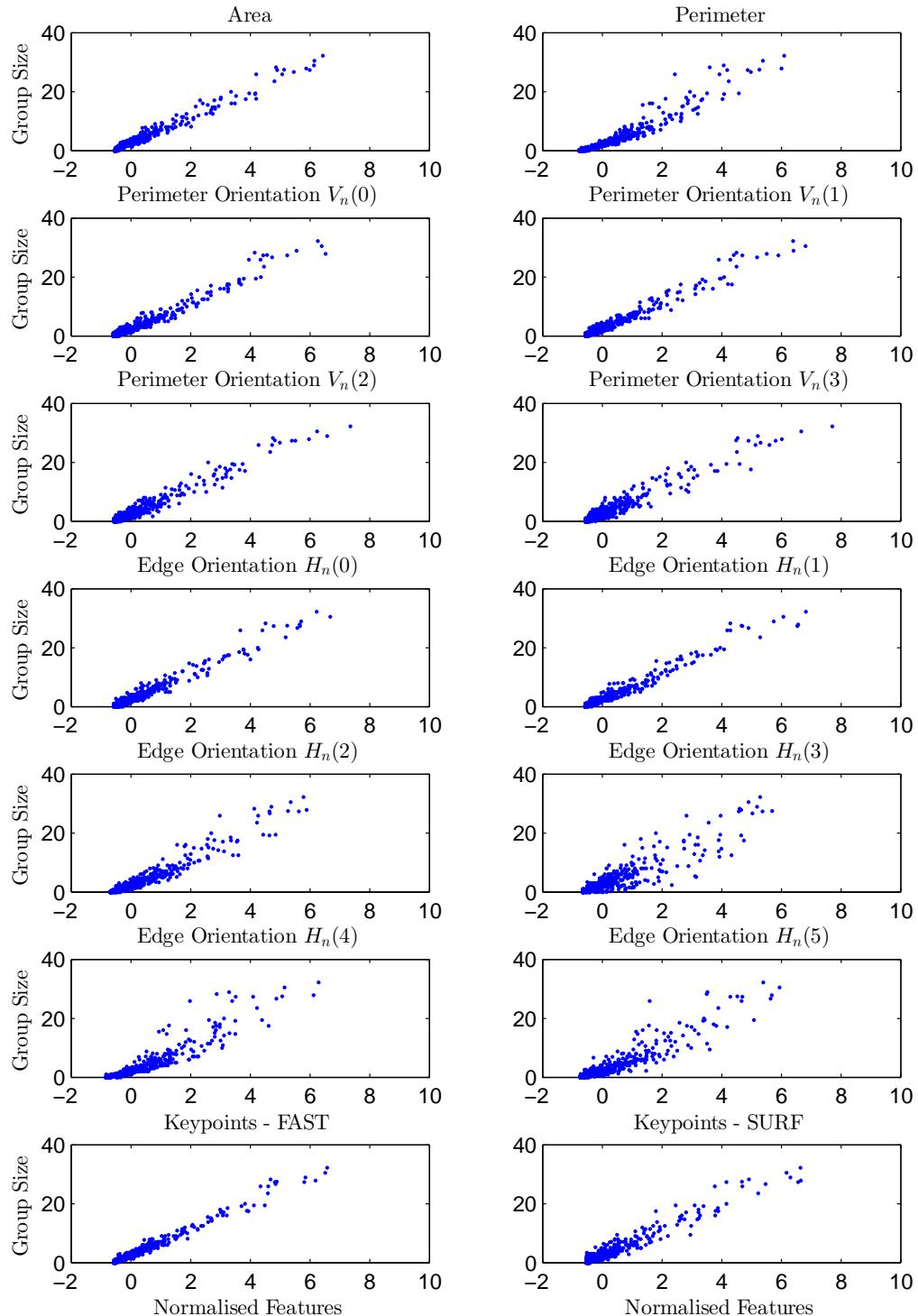


Figure 3.12: Relationship between normalised features and group size on the UCSD dataset [36].

1. **Foreground detection.** The system processes the training dataset using an adaptive background model, and performs foreground detection and group segmentation.
2. **Annotation.** The user annotates each blob in the image using a graphical user interface (GUI) or text input. The ground truth for the n th blob is denoted f_n .
3. **Feature extraction.** The system extracts features from each user-annotated blob. The feature vector for the n th blob is denoted \mathbf{x}_n .
4. **Regression.** The system uses the training dataset $\{\mathbf{x}_n, f_n\}$ to train the regression model.

Due to imperfect foreground segmentation, some blobs are prone to error such as splitting, fading and noise (Figure 3.13). This raises the question of whether fractional counts should be assigned to these imperfect segments, which would be a tedious task for a human user to perform. In such a scenario it would be desirable for a computer to automate this process.

Another problem with the aforementioned annotation strategy is that the ground truth is inherently tied to the foreground detection results. If the background model used by the system is modified at a future date (for example, if a different algorithm is used, or if the parameters are changed), then the nature of the algorithm's output will be different, and therefore the previous annotations will be invalid as they were tied to the results of a different model.

It is desirable for the ground truth to be annotated *independently* of the processing stage. Ideally this would be done in a more conventional manner: by simply identifying the image coordinates of each person in the scene. This process is referred to as ‘dotting’ by Lempitsky [110] because it only requires the user to



(a) Person is fragmented into two blobs (left).

(b) Person is fragmented into two parts (top, centre), one of which is merged with a nearby blob.



(c) Correct extraction of individuals, with additional noise (i.e. small bar near centre).

Figure 3.13: Typical errors in foreground extraction.

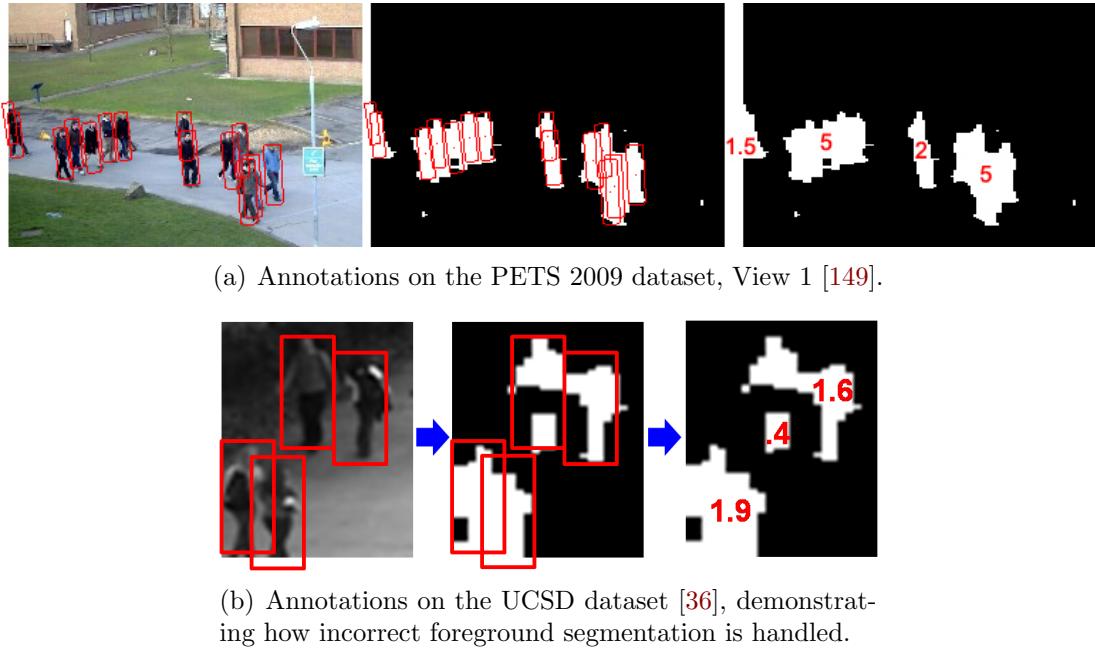


Figure 3.14: The ground truth annotation process. Manual annotations (left) are overlayed on the foreground segmentation results (centre), and the region overlaps are used to automatically determine ground truth counts for each blob (right). Tiny blobs resulting from noise are assigned zero.

click once on the centre of each object in the scene, thereby providing a ‘dot’ annotation. The surrounding region of a person can then approximated by the outline of a *pedestrian template*, such as a rectangle or a cylinder model (Figure 3.14).

The local blob-level annotations are then performed automatically by the system, by assigning the annotated pedestrians to their corresponding foreground segments. This is done by considering the overlap between foreground blobs and the pedestrian templates. For example, in the case of large groups, multiple pedestrian templates will overlap the same blob (Figure 3.14(a)). On the other hand, when blob fragmentation occurs, a single pedestrian template may span multiple blobs (Figure 3.14(b)).

Notation	Description
M	Mask of scene (region of interest).
B	Foreground pixels within the ROI mask. Consists of blobs $\{B_n\}$. (Equation 3.2)
B_n	Blob n within B , where $B = \bigcup_n B_n$. (Equation 3.3)
R_p	Pedestrian template for person p .
$R_p \cap M$	The portion of the pedestrian template R_p that lies within the ROI, M .
$R_p \cap B_n$	The foreground pixels inside R_p belonging to blob B_n , of which there are $ R_p \cap B_n $.
$R_p \cap B$	The foreground pixels inside R_p , of which there are $ R_p \cap B = \sum_n R_p \cap B_n $.

Table 3.3: Various regions in an image. These regions are defined as *sets* of pixels.

In order to handle these various scenarios, the following automated annotation procedure is followed. First we define a number of regions using set notation (Table 3.3). The set of pixels belonging to the region of interest mask is denoted M , while set of foreground pixels detected within the region of interest is denoted B (Equation 3.2). This is partitioned into the set of blobs $\{B_n\}$ (Equation 3.3). Each annotated person p is approximated by the pedestrian template R_p , which is the set of pixels belonging to the rectangular model or a cylinder model which has been projected onto the image plane. Rectangular pedestrian templates were introduced in Section 3.2.2, and the cylinder models are described Section 4.2.2.

The outline of these templates are shown in Figure 3.14.

A template may not necessarily be fully inside the ROI; for example, when pedestrians are entering or exiting a scene and are only partially visible (Figure 3.15, template R_2). In this case a partial blob should receive a fractional annotation.

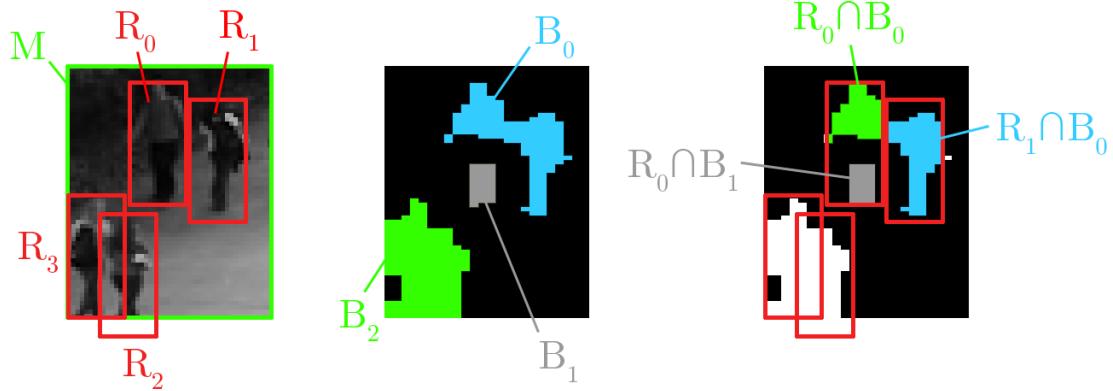


Figure 3.15: Example usage of the notation described in Table 3.3.

Let Q_p define the *quantity* of person p within the scene's ROI:

$$Q_p = \frac{|R_p \cap M|}{|R_p|} \quad (3.38)$$

where $0 \leq Q_p \leq 1$. For example, in Figure 3.15, $Q_0 = 1$ and $Q_2 \approx 0.9$ (corresponding to regions R_0 and R_1 respectively).

The *contribution* of person p to blob n is denoted $C_{p,n}$, and is calculated as follows:

$$C_{p,n} = \frac{|R_p \cap B_n|}{|R_p \cap B|} \times Q_p \quad (3.39)$$

Here the numerator represents the overlap between the pedestrian template and the n th blob, while the denominator represents the total overlap between R_p and *any* foreground pixels. An example of this is shown in Figure 3.15, where:

$$C_{0,0} = \frac{|R_0 \cap B_0|}{|R_0 \cap B|} \times Q_0 = \frac{|R_0 \cap B_0|}{|R_0 \cap B_0| + |R_0 \cap B_1|} \times 1 \approx 0.6 \quad (3.40)$$

$$C_{0,1} \approx 0.4 \quad (3.41)$$

$$C_{1,0} = 1 \quad (3.42)$$

$$C_{2,2} = \frac{|R_2 \cap B_2|}{|R_2 \cap B|} \times Q_1 \approx 1 \times 0.9 = 0.9 \quad (3.43)$$

$$C_{3,2} = 1 \quad (3.44)$$

Finally, the total number of people represented by blob n is the summation of all contributions from all pedestrians:

$$f_n = \sum_p C_{p,n} \quad (3.45)$$

In Figure 3.15, for example:

$$f_0 = \sum_p C_{p,0} = C_{0,0} + C_{1,0} + C_{2,0} + C_{3,0} = 0.6 + 1 + 0 + 0 = 1.6 \quad (3.46)$$

$$f_1 = \sum_p C_{p,1} = 0.4 \quad (3.47)$$

$$f_2 = \sum_p C_{p,2} = 1.9 \quad (3.48)$$

Thus $\{f_n\}$ are the target counts for the blobs in the scene, which have been computed automatically from the ‘dot’ annotations of pedestrian coordinates. This procedure simplifies the annotation process (as the user merely need click once on each person using a GUI); and separates the annotation stage from the segmentation stage. A graphical depiction of this ground truth annotation process from start to finish is displayed in Figure 3.14.

An advantage of this methodology is that small blobs generated by noise are assigned an annotation of zero, while fragmented blobs are assigned fractional counts in proportion to their size. This allows some tolerance for errors in the foreground segmentation algorithm. In other words, we do not assume that an adaptive background model can give perfect results, as this is unrealistic.

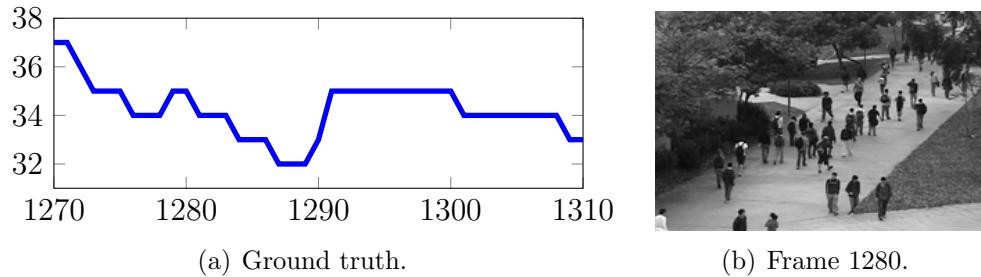


Figure 3.16: Ground truth for the UCSD dataset (frames 1270-1310).

So far we have described a methodology for local ground truth. The *holistic* ground truth can be measured in two ways. We consider ‘hard’ and ‘soft’ values. Hard ground truth is the number of pedestrians whose central dot annotations lie within the region of interest. This measurement introduces an ambiguity when classifying a pedestrian as either ‘in’ or ‘out’ of a region. The exact point in time at which a person is deemed to have entered or exited a frame is never clearly defined. It may take several seconds between a pedestrian reaching the border of the region of interest, and being fully inside or outside of it. Figure 3.16(a) shows the number of people inside the region of interest for the UCSD dataset, over 40 frames (4.0 seconds). As suggested by the number of increments and decrements in this graph, there are at least 12 instances of pedestrians either entering or exiting the scene in this time.

An example frame from this sequence is shown in Figure 3.16(b). The pedestrian at the bottom left in this sequence takes more than 30 frames to fully enter the scene. With groups entering and exiting the scene at this frequency, yet taking several frames to do so, it would be difficult even for a human to estimate the exact crowd size, and impossible for them to remain consistent in their definition of what constitutes being ‘in’ or ‘out’ of the scene. For this reason an alternative definition to ‘hard’ ground truth is proposed, which may be described as ‘soft’ ground truth. In contrast to hard ground truth, soft ground truth assigns fractional values to

those pedestrians lying on the perimeter of the region of interest:

$$Q = \sum_p Q_p \quad (3.49)$$

where Q_p denotes the quantity of person p within the ROI. This allows the holistic ground truth to be temporarily fractional as pedestrians enter or exit the scene's boundary. Both measures are considered when presenting results in Section 3.3. If hard ground truth is provided with a dataset, this is used to allow a fair and direct comparison with the results reported by other authors. However, when hard ground truth is unavailable, manual annotations are used to generate soft ground truth.

3.2.5 Regression

Section 3.2.3 outlines the feature extraction process used to generate a vector of features, \mathbf{x}_n , to describe the crowding within a group, n ; while Section 3.2.4 outlines an annotation methodology for obtaining the localised ground truth, f_n . In order to train the proposed system, a regression function must be learned using the set of training examples, $\{\mathbf{x}_n, f_n\}$, to count the number of people present in each group.

Existing approaches use linear regression [58, 103, 164], neural networks [103, 128, 163] and Gaussian process regression [36]. Although the linear model has demonstrated acceptable performance on single datasets, it is not clear that the relationship between the image features and crowd size is indeed linear across all operating conditions and viewpoints.

We adopt Gaussian process regression (GPR) because it does not place any prior

assumptions on the functional relationship between the features and the crowd size. Instead, GPR may be thought of as defining a distribution over functions, where inference takes place in the space of functions [155, 156]. GPR is reviewed in Section 2.7.3.

The Gaussian Process is defined by the covariance function, $k(\mathbf{x}_m, \mathbf{x}_n)$, which expresses the covariance of outputs as a function of inputs. The covariance function used in this system is designed to capture both short-range and long-range trends in the data. For example, the squared exponential (Equation 2.104) captures the intuitive notion that similar inputs should produce similar outputs:

$$k_{\text{SE}}(\mathbf{x}_m, \mathbf{x}_n) = \sigma_{\text{SE}}^2 \exp\left(-\frac{1}{2\ell^2} |\mathbf{x}_m - \mathbf{x}_n|^2\right) \quad (3.50)$$

In order to extrapolate the longer trends beyond the training range, the dot product covariance function [156] is also used:

$$k_{\text{DP}}(\mathbf{x}_m, \mathbf{x}_n) = \sigma_{\text{DP}}^2 (1 + \mathbf{x}_m^T \mathbf{x}_n) \quad (3.51)$$

Combining these terms results in a regression model that preserves the non-linearities within the training range while extrapolating outside of the training range in a predominantly linear fashion. Finally, independent Gaussian noise is modeled using the term:

$$k_{\text{GN}}(\mathbf{x}_m, \mathbf{x}_n) = \sigma_{\text{GN}}^2 \delta(m, n) \quad (3.52)$$

where δ denotes Kronecker's delta function, and contributes only to the diagonals of \mathbf{K} and \mathbf{K}^{**} (Section 2.7.3). The final covariance function is therefore:

$$k(\mathbf{x}_m, \mathbf{x}_n) = k_{\text{SE}}(\mathbf{x}_m, \mathbf{x}_n) + k_{\text{DP}}(\mathbf{x}_m, \mathbf{x}_n) + k_{\text{GN}}(\mathbf{x}_m, \mathbf{x}_n) \quad (3.53)$$

$$= \sigma_{\text{SE}}^2 \exp \left(-\frac{1}{2\ell^2} |\mathbf{x}_m - \mathbf{x}_n|^2 \right) + \sigma_{\text{DP}}^2 (1 + \mathbf{x}_m^T \mathbf{x}_n) + \sigma_{\text{GN}}^2 \delta(m, n) \quad (3.54)$$

The GPR is ‘trained’ by choosing the hyperparameters, $\{\sigma_{\text{SE}}, \ell, \sigma_{\text{DP}}, \sigma_{\text{GN}}\}$, so as to maximise the likelihood of the observed training data (Section 2.7.3). Optimisation is performed using the conjugate gradient method. Once optimised, prediction is then performed using Equations 2.110-2.114 (p. 97).

For each group of people, the crowd size estimate is a predictive distribution, $\mathcal{N}(\mu_n, \sigma_n^2)$. To obtain a holistic estimate, these distributions must be combined to get the total number of people in the scene. By calculating the sum of N^* Gaussian random variables, an overall prediction and variance is obtained for the scene:

$$\mu = \sum_{n=1}^{N^*} \mu_n \quad (3.55)$$

$$\sigma^2 = \sum_{n=1}^{N^*} \sigma_n^2 \quad (3.56)$$

Thus the holistic crowd size estimate is μ , with variance σ^2 and 95% confidence interval $(\mu - 1.96\sigma, \mu + 1.96\sigma)$.

3.2.6 Tracking Module

Crowd counting algorithms have typically analysed each frame independently of one another, estimating the crowd size based on the features extracted from that

frame alone. Although a temporal smoothing may be applied to the holistic count to reduce outliers [58, 163], we propose a local method which employs blob-level tracking to improve each *group*'s estimate.

When two or more groups merge to form a larger group, for example, occlusions often occur that obscure the crowd size estimate. By tracking and counting these groups before they merge, their prior estimates can be used to anticipate the size of the newly formed group. Because occlusions are usually temporary (consider two pedestrians passing by one another on a walkway), this prior information can be used to prevent the estimate from being diminished.

Blobs are tracked as they move through a scene by detecting direct correspondences, splits and merges. This is formulated as an optimisation problem by Masoud [131], however in this section we describe an efficient set of heuristics based on blob overlap criteria. As we are not concerned with ensuring consistent labeling of objects throughout the sequence, as is required in object tracking, a heuristic based approach that can model the merges and splits of blobs is adequate.

Denoting the m th blob in frame t as $B_{t,m}$, we define the overlap of two blobs in consecutive frames as the number of pixels belonging to both:

$$O_t(m, n) = |B_{t,m} \cap B_{t+1,n}| \quad (3.57)$$

Using this notation we track groups throughout a sequence by determining direct matches, merges and splits as follows.

- 1. Direct Match:** The first step in comparing consecutive frames is to detect direct matches between overlapping blobs. Any blob pair, $B_{t,m}$ and $B_{t+1,n}$,

which satisfies the following conditions is deemed a match:

$$O_t(m, n) > 0 \quad (3.58)$$

$$O_t(i, n) = 0 \quad \forall i \neq m \quad (3.59)$$

$$O_t(m, j) = 0 \quad \forall j \neq n \quad (3.60)$$

These criteria simply require both blobs to overlap one another exclusively.

2. **Merging:** After direct matches have been determined, the matched blobs are removed from consideration. The system then detects P:1 merges and 1:Q splits by combining the remaining blobs as follows. A set of P blobs, $\{B_{t,M_1}, B_{t,M_2}, \dots, B_{t,M_P}\}$, are deemed to have merged to form the blob, $B_{t+1,n}$, when the following conditions are met:

$$O_t(M_p, n) > 0 \quad \forall p \in [1, P] \quad (3.61)$$

$$O_t(i, n) = 0 \quad \forall i \notin \{M_0, M_1, \dots, M_P\} \quad (3.62)$$

$$O_t(M_p, j) = 0 \quad \forall p \in [1, P], \quad \forall j \neq n \quad (3.63)$$

3. **Splitting:** Similarly, a split occurs when blob $B_{t,m}$ is divided into the set of blobs: $\{B_{t+1,S_1}, B_{t+1,S_2}, \dots, B_{t+1,S_Q}\}$. A split is determined when the following conditions are met:

$$O_t(m, S_j) > 0 \quad \forall j \in [1, Q] \quad (3.64)$$

$$O_t(i, S_j) = 0 \quad \forall j \in [1, Q], \quad \forall i \neq m \quad (3.65)$$

$$O_t(m, j) = 0 \quad \forall j \notin \{S_0, S_1, \dots, S_Q\} \quad (3.66)$$

The crowd counting estimates obtained for each blob can then be improved by taking advantage of the detected tracks. The splitting and merging of blobs may

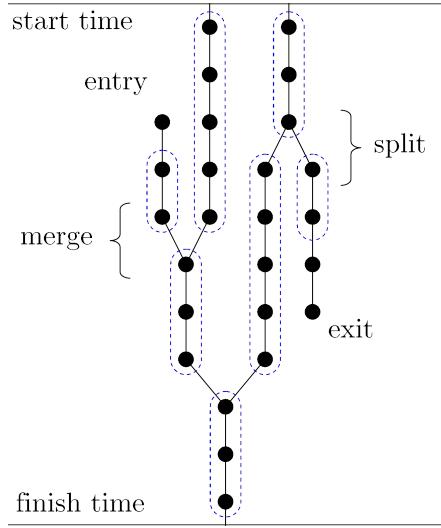


Figure 3.17: Visualisation of blob tracking results. Groups of constant size are circled.

be visualised using a graph structure as shown in Figure 3.17. As blobs enter and exit the scene, the number of persons that they represent may change while in contact with the perimeter of the scene. Once fully inside the region of interest, however, we assume that directly-matched blobs represent a constant number of people, while merged blobs represent the sum of their constituents' group sizes.

The estimate for the m th blob in frame t is denoted $\mu_{t,m}$ with variance $\sigma_{t,m}^2$. Estimates are obtained using Equations 2.110-2.113. A group which has been tracked across N frames, from time $t = t_1$ to $t = t_N$, and containing the blobs with indices $\{m_t\}_{t=t_1}^{t_N}$, has an associated set of group size predictions: $\{\mu_{t,m_t}, \sigma_{t,m_t}^2\}_{t=t_1}^{t_N}$. It is reasonable to expect that the number of people contained in this group is constant, if it is fully contained within the region of interest. We therefore seek to obtain an improved estimate for this group size, $\mu'_{t_N, m_{t_N}}$, by incorporating the tracking history.

Previous experiments [164] used the mean or median value of the group's historical list of estimates, while Kilambi [97] rounded each to an integer and then took the mode. These approaches assume each estimate to be equally valid, and therefore

assign an equal weighting to each. However, in practice some frames may be less reliable due to changing environmental conditions or noise, which contribute to uncertainty in the predicted group size.

The variance provides a measurement of the system's uncertainty in the group size prediction, therefore each estimate within a track is weighted by the inverse of its variance. The improved estimate for the most recent blob in a track is thus the weighted average:

$$\mu'_{t_N, m_{t_N}} = \frac{\sum_{t=t_1}^{t_N} \mu_{t, m_t} / \sigma_{t, m_t}^2}{\sum_{i=t_1}^{t_N} 1 / \sigma_{t, m_t}^2} \quad (3.67)$$

When two or more groups merge to form a new group, each contains a historical list of estimates and variances. A new list is formed by summing their corresponding elements and truncating the new list's length to the shortest of those being merged. The merged group adopts this list and then appends to it any subsequent estimates while it continues to be tracked. Consequently, a tracked person who is temporarily occluded from view by another group may still be represented in the crowd size estimate due to the weight of its prior history.

Note that when a blob $\{t, n\}$ does not correspond to a match or merge from the previous frame, its estimate is not modified in any way, i.e. $\mu'_{t, n} = \mu_{t, n}$ for these blobs.

Given the improved estimates in a frame at time t , denoted by $\{\mu'_{t, n}\}$, we obtain an improved holistic estimate:

$$\mu'_t = \sum_n \mu'_{t, n} \quad (3.68)$$

The tracking procedure described in this section effectively filters the group size

estimates over time as the blobs are tracked. As such, it may be expected to produce a modest improvement over the underlying ‘raw’ estimates obtained using the procedure in Section 3.2.5. Experimental results for the tracking module are presented in Section 3.3.5.

3.3 Experimental Results

This section presents experimental results of the proposed crowd counting algorithm. Section 3.3.1 describes the operational requirements of a crowd counting system in practice. Section 3.3.2 introduces the benchmark datasets used in these experiments. Section 3.3.3 uses cross validation to compare the various features and regression models in the proposed algorithm. A comparison of local and holistic features is also presented. Section 3.3.4 compares the proposed approach to other algorithms found in the literature. Section 3.3.5 evaluates the tracking module of the proposed algorithm. Section 3.3.6 evaluates long-term performance of the algorithm.

3.3.1 Operational Requirements

The accuracy of a system is a measure of how closely the estimate follows the ground truth. The predictive performance of a crowd counting system can be evaluated using the following criteria. For testing purposes we consider the *mean absolute error* (MAE), the *mean square error* (MSE) and *mean relative error* (MRE). Letting μ'_t and Q_t denote the system’s estimate and the ground truth respectively (Equations 3.49 and 3.68), these metrics are calculated across a set of testing frames, T , as follows:

Mean absolute error (MAE):

$$\epsilon_{mae} = \frac{1}{|T|} \sum_{t \in T} |\mu'_t - Q_t| \quad (3.69)$$

Mean square error (MSE):

$$\epsilon_{mse} = \frac{1}{|T|} \sum_{t \in T} (\mu'_t - Q_t)^2 \quad (3.70)$$

Mean relative error (MRE):

$$\epsilon_{mre} = \frac{1}{|T|} \sum_{t \in T \wedge Q_t \neq 0} \left| \frac{\mu'_t - Q_t}{Q_t} \right| \times 100\% \quad (3.71)$$

Note that the calculation of MRE ignores frames whose ground truth is zero because the relative error is undefined for these cases. There are few instances of this in our datasets.

According to an internal study cited by Regazzoni [157], “End users accept a mean error of 20% with respect to the real number of people present in a controlled area.” This means that a system should achieve $\epsilon_{mre} < 20\%$ to meet the minimum accuracy requirements of system operators.

In these Equations, μ'_t can be replaced with μ_t to omit the tracking and refinement procedure of Section 3.2.6. The ground truth Q_t can be defined as *hard* or *soft*, as discussed in Section 3.2.4.

3.3.2 Benchmark Datasets

Five benchmark datasets were used to evaluate the performance of the proposed algorithm.

1. The **UCSD** dataset introduced by Chan [36].
2. The **PETS 2009** dataset introduced at the Eleventh IEEE International Workshop on Performance Evaluation of Tracking and Surveillance [149].
3. The **Mall** dataset introduced by Chen [42].
4. The **Fudan** dataset introduced by Tan [181].
5. The **Grand Central** dataset introduced by Zhou [206].

So far in the literature, testing has focused heavily on single datasets such as UCSD and PETS 2009. The use of limited datasets can result in overfitting due to the lack of varying crowding conditions. For example, both the UCSD and PETS 2009 databases contain only *moving* pedestrians, allowing methods based on optical flow [52] or dynamic textures [35] to obtain good results that may not translate to other scenes in which pedestrians come to a stop, which is particularly likely in overcrowded environments. It is especially important for a crowd counting algorithm to work under these conditions, because overcrowding is an anomalous condition that may require intervention from security personnel. Furthermore, the sequences in the PETS 2009 dataset are very short (less than 35 seconds), while the UCSD dataset is annotated for only slightly longer (less than four minutes) making long term performance difficult to assess.

Eventually, the optimisation of algorithms and parameters can achieve excellent results on these datasets, but at the expense of generalisation. Ideally, system

parameters are selected to optimise performance on a validation set during the training procedure, in the hope that performance will also be optimised for an *unseen* test set. In practice, these test sets are not unseen, and over time they effectively become part of the validation set (i.e. training procedure). A similar observation is made by Geiger [76] regarding the popular Middlebury database [16] which is used for evaluating optical flow:

Perhaps not surprisingly, many algorithms that do well on established datasets such as Middlebury [16, 169] struggle on our benchmark. We conjecture that this might be due to their assumptions which are violated in our scenarios, as well as overfitting to a small set of training (test) images.

As Geiger notes, the differentiation between the “training (test)” sets has become ambiguous. Of course, this is not the intent of the researchers who use the datasets, but it is true nonetheless: beyond a certain point, a single short sequence can not be used to distinguish between true improvements in technology and overfitting. This is difficult to avoid because *some* data must be used to evaluate performance. Nevertheless, there are a number of steps which can be taken to reduce the problem:

1. Use as few manual parameters as possible in the design of a system.
2. Hold manual parameters constant across all experiments and datasets. This avoids unrealistic overfitting designed to optimise performance for each sequence.
3. Use as many datasets as possible to avoid overfitting to any individual scene or sequence.

4. Use a sequence whose duration is as long as possible, or at least use numerous sequences captured at different times, in order to test across sufficiently broad conditions.
5. Follow a cross validation procedure such as K -fold cross validation to assess the generality of an algorithm.

With regards to the experiments used in this thesis, the following points are noted:

1. The proposed system uses very few parameters. The perspective map is calculated once when calibrating the system. The procedure in Section 3.2.2 is followed once without any additional tweaking. The features defined in Section 3.2.3 are defined explicitly without any tunable parameters. Training and regression (Sections 3.2.4 and 3.2.5) have no parameters, aside from the GPR hyperparameters which are automatically selected by maximising the likelihood on the training data (Equation 2.115). This is an automated process. The tracking procedure in Section 3.2.6 does not use any parameters either¹.

The adaptive background model does use some thresholds, which are held constant for each scene. It is appropriate to vary these thresholds between different datasets because some scenes are prone to greater lighting fluctuations than others, and these parameters would be established once during the setup of a system. However, it is not appropriate to vary thresholds between different sequences/timestamps *within* a dataset, because this would not occur in practice while a system is running. Therefore these parameters are held constant across all experiments for a dataset.

¹Although the memory length of the tracker could be considered a parameter, it is set to a large value so that groups are rarely tracked for this duration of time.

2. Five datasets are used to assess performance. The UCSD and PETS 2009 datasets are used to compare the proposed algorithm with other published research, and in addition to this, some very recent datasets have also been included: the Fudan, Mall and Grand Central datasets. This is the first analysis of an algorithm over such a broad range of data.
3. The datasets, taken together, contain various conditions and crowd distributions. The UCSD dataset is annotated for the first 2000 frames. The PETS 2009 dataset contains short sequences, although we select five of them to incorporate some variation in crowd properties. The Mall dataset contains 2000 frames at 2fps, which covers more than 16 minutes of footage containing both moving and stationary pedestrians. The Fudan dataset contains five sequences, each of length 300 frames taken at different times, in an outdoor setting. Finally, the Grand Central dataset features very large crowds which vary quite significantly, containing between 125 and 245 people over 33 minutes.

The datasets are summarised in Table 3.4 and Table 3.5.

The PETS 2009 database was released prior to the Eleventh IEEE International Workshop on Performance Evaluation of Tracking and Surveillance [149] in order to test a multitude of visual surveillance tasks: object tracking, crowd counting and event recognition. Two sequences were designated for counting the number of people in the image, labelled 13-57 and 13-59, and a region of interest is specified for View 1. Additionally, sequences 12-34, 12-43 and 14-06 are used. The video is provided at a resolution of 768×576 pixels and ~ 7 fps in RGB colour format. An example frame from View 1 is shown in Figure 3.18(a). Annotations for these datasets were obtained from Milan [136].

The Fudan dataset was introduced by Tan [181] and contains five sequences each of length 300 frames. The greyscale images are provided at a resolution of 320×240 . Holistic ground truth for each frame is provided with the dataset, and additional local annotations were added manually to train the system. These manual annotations were performed on frames 10:20:290 from each sequence, as indicated in Table 3.4. An image from this dataset is shown in Figure 3.18(b).

The Grand Central dataset was introduced by Zhou [206] to model the collective behaviour of crowds. The footage is obtained in greyscale from New York's Grand Central station, and the resolution is 720×480 . Due to the extremely large size of this crowd, annotation of each frame is not feasible, therefore a sparse subset of frames has been selected over a long period of time (33 minutes) and annotated individually. The dataset is shown in Figure 3.18(c).

The UCSD pedestrian database was introduced by Chan [36] and contains 2000 annotated frames of pedestrian traffic moving in two directions along a walkway. The video has been distributed at a down-sampled resolution of 238×158 pixels and 10 fps, in greyscale. An example frame is shown in Figure 3.18(d).

The Mall pedestrian database was introduced by Chen [42]. This database contains 2000 annotated frames of pedestrian traffic moving and stopping inside a cluttered indoor shopping centre. The video is provided at a resolution of 640×480 pixels and 2 fps, in colour format. An example frame is shown in Figure 3.18(e).

3.3.3 Cross Validation Experiments

In this section the performance of the proposed algorithm is evaluated using the feature sets described in Section 3.2.3 and summarised in Table 3.2. A cross validation procedure is used to evaluate the model parameters, including the



(a) PETS 2009 [149]



(b) Fudan [181]



(c) Grand Central [206]



(d) UCSD [36]



(e) Mall [42]

Figure 3.18: Images from each of the five benchmark datasets used to evaluate the proposed crowd counting algorithm.

Dataset	Sequence	Test Set	Training Subset	Crowd Size
PETS 2009	12-34	0:794	20:40:780	2 to 8
	12-43	0:106	5:10:105	1 to 7
	13-57	0:220	5:10:215	5 to 34
	13-59	0:240	5:10:235	3 to 25
	14-06	0:200	5:10:195	0 to 42
Fudan	1	1:300	10:20:290	3 to 15
	2	1:300	10:20:290	2 to 15
	3	1:300	10:20:290	1 to 14
	4	1:300	10:20:290	2 to 11
	5	1:300	10:20:290	0 to 12
Grand Central	1	1000, 6000	1000, 6000	132 to 152
	2	11000, 16000	11000, 16000	151 to 160
	3	21000, 26000	21000, 26000	125 to 138
	4	31000, 36000	31000, 36000	141 to 176
	5	41000, 46000	41000, 46000	200 to 245
UCSD	1	1:400	10:20:390	12 to 27
	2	401:800	410:20:790	11 to 25
	3	801:1200	810:20:1190	11 to 40
	4	1201:1600	1210:20:1590	29 to 45
	5	1601:2000	1610:20:1990	17 to 31
Mall	1	1:400	20:40:380	13 to 50
	2	401:800	420:40:780	20 to 50
	3	801:1200	820:40:1180	20 to 53
	4	1201:1600	1220:40:1580	17 to 48
	5	1601:2000	1620:40:1980	20 to 48

Table 3.4: The benchmark datasets used to evaluate the proposed crowd counting algorithm. The total number of frames is listed, and a subset of these frames have been annotated at regular intervals with ground truth, indicated using MATLAB notation (p. xxxiii). (The frames of the UCSD and Mall datasets are 1-indexed, while the remaining datasets are 0-indexed. We retain the indexing used by the original authors to avoid confusion.) Note that training and testing is performed on different sequences using K -fold cross validation (Section 3.3.3)

	PETS 2009	Fudan	Grand Central	UCSD	Mall
Length (frames)	1565	1500	46009	2000	2000
Frame Rate (fps)	~7	10	23	10	<2
Resolution	768×576	320×240	720×480	236×158	640×480
Colour	RGB	Grey	Grey	Grey	RGB
Location	Outdoor	Outdoor	Indoor	Outdoor	Indoor
Shadows	Yes	Yes	No	No	Yes
Reflections	No	No	Yes	No	Yes
Loitering	No	Yes	Yes	No	Yes
Crowd Size	0 to 42	0 to 15	125 to 245	13 to 53	11 to 45

Table 3.5: Summary of the various conditions in the benchmark datasets.

feature selection and regression models.

K -fold cross validation is a procedure used to evaluate a model by rotating the training and testing datasets. The data is divided into K subsets, and the model is trained using $(K - 1)$ subsets. Testing is then performed on the remaining set. This procedure is repeated, as shown in Figure 3.19, until each subset has been used exactly once for testing. The average performance across all of the testing subsets is then taken as a measure of the system’s performance, and it is used as an indicator of a model’s ability to generalise across all conditions. K -fold cross validation has been shown to be most effective for accuracy estimation and model selection on real world datasets [101].

The Fudan dataset lends itself to 5-fold cross validation as it is already divided into five sequences of length 300 frames. The UCSD and Mall datasets are each 2000 frames, which are divided into 5×400 frame sequences. The PETS 2009 dataset contains numerous sequences designed for various challenges (tracking, crowd counting and event detection). Five of these are selected: crowd counting sequences (13-57 and 13-59), sparse crowd sequences (12-34 and 12-43) and a very densely crowded sequence (14-06). These sequences capture a good variation in crowd properties at different times. Finally, the Grand Central dataset contains

	<i>Set 1</i>	<i>Set 2</i>	<i>Set 3</i>	<i>Set 4</i>	<i>Set 5</i>
Experiment 1	Testing	Training	Training	Training	Training
Experiment 2	Training	Testing	Training	Training	Training
Experiment 3	Training	Training	Testing	Training	Training
Experiment 4	Training	Training	Training	Testing	Training
Experiment 5	Training	Training	Training	Training	Testing

Figure 3.19: K -fold cross validation procedure (with $K = 5$). The data is divided into K sets, and one set is withheld for testing during each experiment. The training and testing sets are rotated to test the model’s capacity for generalisation.

extremely large crowds of up to 245 people. In order to capture different crowd properties over time, the frames are annotated at extremely sparse intervals and then divided into five contiguous subsets.

In each experiment, one sequence is withheld for testing, while the remaining four sequences are used to train the system. From these four training sequences, a subset of frames is selected to train the system. It is not necessary to use *all* of the frames within the training sequences for the following reasons:

1. Samples are not independent due to temporal correlation between consecutive frames. In fact, two consecutive frames will be nearly identical, both on a holistic level and in terms of the foreground segments contained within the image. In order to avoid redundant training data the samples should be selected at sparse intervals (from within the training set).
2. Excessive redundant data results in slower computation of the covariance matrix \mathbf{K} (Equation 2.103). The training procedure is particularly slowed because Equation 2.115 must be calculated during each iteration of maximum likelihood optimisation. This involves the calculation of $|\mathbf{K}|$ and $\mathbf{K}^{-1}\mathbf{f}$, which requires computation of the inverse (or pseudoinverse) of \mathbf{K} . These terms do not need to be recalculated during testing, however their size

- still has a negative impact on computation time (Equations 2.111-2.112).
3. Each frame actually yields multiple training samples. This is due to the annotation strategy described in Section 3.2.4, in which each foreground segment is automatically labelled with a local crowd count. Therefore it is not necessary to use hundreds [58, 103] or thousands [38] of training frames, as has been used by some methods in the literature. In the proposed system, for example, a single frame may yield between 0 and 40 samples of training data (depending on crowd distribution and noise levels), and for the Grand Central dataset [206] more than 150 blobs are detected per frame. This suggests that far fewer frames of annotated data are required for the proposed system than holistic methods.
 4. Consecutive frames appear in different subsets because the datasets have been partitioned into contiguous subsets. For example, the UCSD dataset has been divided into subsets containing frames 1-400, 401-800, and so on. It is not appropriate for frame 400 to appear in the training dataset if frame 401 will be used for testing, as this may result in overfitting to a specific set of conditions rather than proper generalisation. Therefore a sparse subset of the training frames are used, and those occurring at the ‘edge’ of each sequence are avoided. For example, if frames 1:400 are set aside for testing, then the system will be trained on frames 410:20:1990 (rather than 401:2000).

In the literature, ‘dense’ frame annotations have been commonly used: Chan [36] used frames 600-1399 of the UCSD dataset for training, while the remaining frames (1-599, 1400-2000) were used for testing. Similarly, Chen [42] used frames 1-800 of the Mall dataset for training and frames 801-2000 for testing.

Sparse frame annotations were employed by Tan [181], however the training

frames were mixed in amongst the testing frames. Tan used K -means clustering to divide the Fudan dataset into $K = 400$ clusters, and the desired number of training frames were selected at random from different clusters to achieve maximum variability in the training data. The remaining frames were used for testing. Since consecutive frames were used for training and testing, excellent accuracy was observed because many of the images in the training and testing sets were nearly identical. It is evident that this approach is particularly susceptible to overfitting.

In addition to sparse frame annotations, a sparse subset of blobs may be selected from within the training dataset, in order to remove redundant data and hence reduce the size of the covariance matrix \mathbf{K} . Redundancy is introduced into the training data when the foreground detection result includes small instances of noise, such as that shown in Figure 3.20. Although the larger blobs correspond to human groups of various sizes, small blobs usually do not, and therefore have a target value of 0. While it is important for the regression model to learn that these small blobs correspond to a null target, an excessive number of such instances will inflate the size of \mathbf{K} needlessly. Table 3.6 outlines the parameters used for blob subsampling on the benchmark datasets: samples whose target falls within the specified range are identified, and a random subset of these are chosen for inclusion in the regression model; the number of instances used in each case is chosen so as to keep the size of \mathbf{K} smaller than 1000×1000 .

In summary, K -fold cross validation is a more appropriate evaluation procedure compared to using a designated training and test set because all samples are used for testing at some point during the rotation. Furthermore, by avoiding frames at the ‘edges’ of consecutive sequences, the training and test sets are not allowed to bleed into one another due to temporal correlations, and therefore overfitting is avoided. Finally, sparse annotations are used to avoid excessive redundant data.

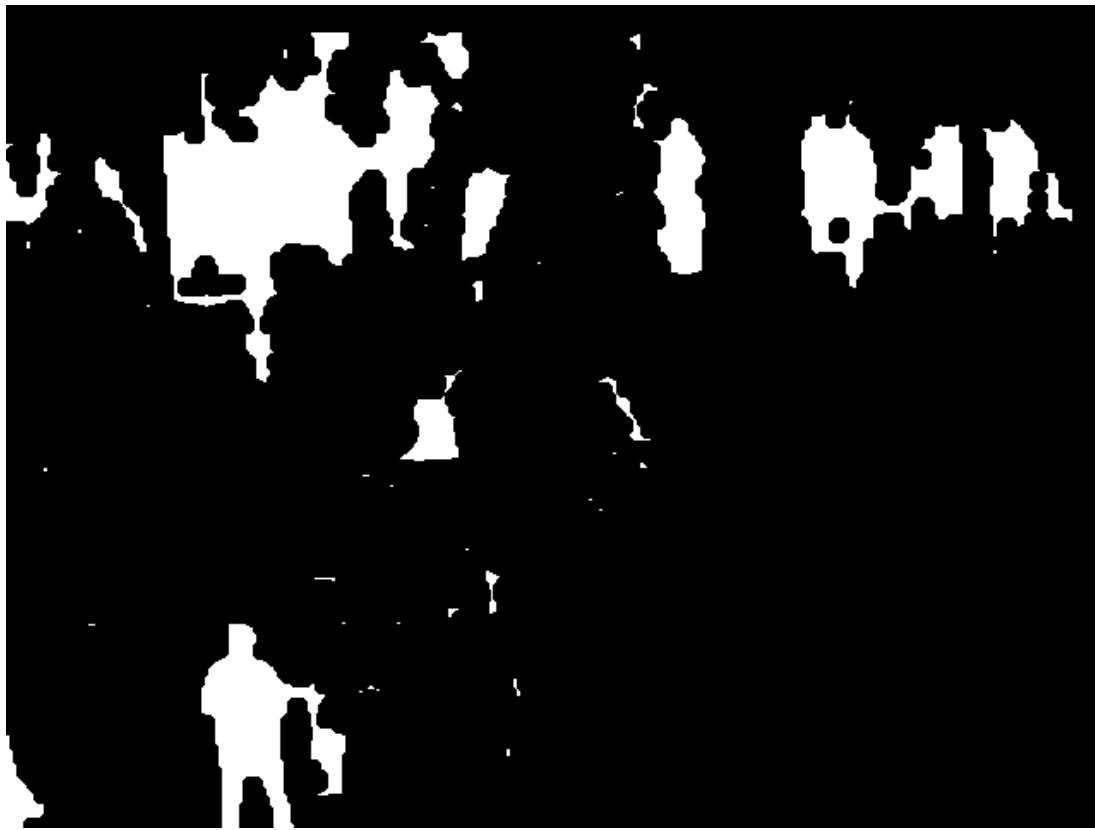


Figure 3.20: Foreground detection result from frame 60 of the Mall dataset [42]. Note that the foreground includes small instances of noise in addition to the larger blobs which correspond to humans.

Dataset	Target Range	Samples
PETS 2009	[0.0, 0.1]	100
Fudan	[0.0, 0.1]	100
Grand Central	[0.0, 0.1]	100
	(0.1, 1.0)	100
	[1.0, 2.0]	100
UCSD	[0.0, 0.1]	100
Mall	[0.0, 0.1]	400

Table 3.6: Parameters used for blob subsampling on the benchmark datasets. Any samples whose target falls within the specified range are subsampled at random, and the number of samples selected is specified by the ‘Samples’ column.

The sparse training subsets used for each sequence are shown in Table 3.4.

The remainder of this section is structured as follows. Section 3.3.3.1 evaluates the performance of various image features; Section 3.3.3.2 evaluates several regression models; and Section 3.3.3.3 compares the performance of local features to holistic features.

3.3.3.1 Feature Evaluation

Features are categorised into *size*, *shape*, *edge* and *keypoint* types (Table 3.2). Fifteen different combinations of these features are assessed, as shown in Table 3.7. Using the 5-fold cross validation procedure as described in Section 3.3.3, error rates are averaged across all frames for each dataset, and reported in terms of mean absolute error (MAE), mean relative error (MRE) and mean square error (MSE).

The algorithm proposed in this thesis is described as ‘Local Features’ (in contrast to ‘Holistic Features’, whereby equivalent features are analysed at the holistic level). Both local and holistic features will be evaluated in this section, however a more detailed comparison of the two approaches is presented in section 3.3.3.3.

Table 3.7 summarises the results for the UCSD dataset. Average error rates are reported under their respective columns, as well as a *ranking* from 1 to 15 indicating the relative performance of each feature set. For example, the lowest error rate is observed when *size*, *shape* and *edge* features are used, whereas the highest error rate is observed from *shape* features alone. These feature sets are ranked 1 and 15 respectively.

In each column, the top three results (ranked 1-3) are highlighted in bold. The best performing feature sets for the UCSD dataset are:

1. Size, Shape, Edges
2. Shape, Edges, Keypoints
3. Size, Shape, Edges, Keypoints (*all features*)

In general, it can be seen that performance improves on this dataset as more features are included. Poor performance is particularly seen when only one feature type is used (size, shape or keypoints alone). The exception to this is edge features, which perform quite well when taken alone, perhaps because ‘edges’ refers to the 6-bin orientation histogram, which constitutes a total of six features. In fact, local edge features alone outperform *any* of the holistic feature combinations (shown on the right hand side of the table).

The mean relative error of the proposed system is less than 6% for the top three results. This is well below the 20% requirement suggested by Regazzoni [157].

Table 3.8 summarises the results for the PETS 2009 dataset. As before, individual features (particularly *size* and *shape* features taken alone) exhibit relatively poor performance, with mean relative errors falling above the 20% threshold of acceptability. However, more features tend to perform better in general, and a mean relative error of 13-14% is observed for the best feature sets. As indicated in bold, the best performing features were: “Size, Keypoints”, “Shape, Edges”, “Size, Shape, Edges”, “Shape, Edges, Keypoints”, and all features.

Interestingly, the *size and keypoints* feature vector achieved a MSE of 7.536 (ranked 3rd), while its MRE failed at 21.45% (ranked 13 out of 15). This is due to errors occurring when the crowd size is small (for example, estimating the crowd size to be 2 when the true value is 1; a 100% error). This suggests that the MRE requirement may not be reasonable for small crowds, or that all three metrics should be taken into account collectively when analysing a system.

The Fudan dataset is summarised in Table 3.9. The best performance is observed when *all features* are used, and the worst performance is seen when individual feature types are used (particularly size and shape). This is consistent with the results for the UCSD and PETS 2009 datasets. Edges and keypoints perform significantly better on the Fudan dataset, even though one might expect blob size and shape to be the most direct indicators of crowd size.

The Mall dataset is summarised in Table 3.10. Due to the cluttered nature of this scene, as well as the many shadows and reflective surfaces, it is a particularly difficult dataset, and this is reflected in the MAE and MSE values reported in the table. However, because the crowds are relatively large (13-53), the *relative* error (MRE) is not as significant as in the other datasets (less than 10%). As before, size and shape features alone perform quite poorly. The best feature sets are:

1. Keypoints
2. Edges, Keypoints
3. Size, Edges, Keypoints

Suboptimal performance is seen when *all features* are used. This is due to the inclusion of the *shape* feature, which performs relatively poorly on this dataset; note that the shape feature appears in 5 out of the 6 worst-performing feature sets. Due to the reflective surfaces and complicated structure of this scene, foreground segmentation is particularly poor, resulting in substantial noise and unusually-shaped blobs. It is not surprising, therefore, than blob-shape features perform relatively poorly under these conditions. (Note, however, that performance is still quite good compared to the equivalent holistic system shown on the right-hand side of Table 3.10; 12 out of 15 local feature sets outperform the *best* holistic feature set.)

Finally, Table 3.11 presents results on the Grand Central dataset. Due to the small number of annotated frames in this dataset, it was not possible to obtain a trained model for the equivalent holistic system, but the results for local features are shown. These results confirm that single features perform poorly, with more stable performance obtained by larger feature sets. The best feature sets were:

1. Size, Keypoints
2. Shape, Keypoints
3. Size, Edges, Keypoints

Although the absolute error and squared error are quite high for this dataset, the crowd size ranges from 125 to 245 people, so this is understandable; the mean *relative* error is less than 5%, which is the most accurate MRE observed in these experiments, and well within the threshold of acceptability. The “Size, Keypoints” feature achieved a mean relative error of just 3.49%.

Features	Local Features						Holistic Features					
	MAE		MSE		MRE		MAE		MSE		MRE	
Size	1.970	(13)	6.218	(13)	8.14%	(13)	6.640	(11)	152.954	(11)	31.97%	(11)
Shape	2.991	(15)	14.804	(15)	11.85%	(15)	3.306	(10)	17.870	(10)	13.71%	(10)
Edges	1.486	(7)	3.505	(5)	6.60%	(8)	14.679	(15)	918.143	(15)	41.67%	(15)
Keypoints	2.006	(14)	6.570	(14)	8.24%	(14)	1.970	(9)	6.010	(9)	8.57%	(9)
Size, Shape	1.950	(12)	6.102	(12)	7.93%	(12)	7.865	(14)	211.021	(14)	34.43%	(13)
Size, Edges	1.470	(6)	3.522	(6)	6.40%	(5)	1.597	(2)	3.975	(2)	7.18%	(3)
Size, Keypoints	1.763	(9)	5.069	(9)	7.31%	(9)	1.731	(7)	4.720	(7)	7.75%	(8)
Shape, Edges	1.437	(4)	3.409	(4)	6.13%	(4)	1.647	(5)	4.242	(5)	7.28%	(4)
Shape, Keypoints	1.798	(10)	5.319	(10)	7.37%	(10)	7.403	(12)	164.653	(12)	36.77%	(14)
Edges, Keypoints	1.456	(5)	3.580	(7)	6.41%	(6)	7.513	(13)	195.668	(13)	32.72%	(12)
Size, Shape, Edges	1.377	(1)	3.145	(1)	5.83%	(1)	1.580	(1)	3.911	(1)	6.95%	(1)
Size, Shape, Keypoints	1.827	(11)	5.453	(11)	7.45%	(11)	1.757	(8)	4.891	(8)	7.66%	(7)
Size, Edges, Keypoints	1.499	(8)	3.754	(8)	6.46%	(7)	1.637	(4)	4.180	(4)	7.29%	(5)
Shape, Edges, Keypoints	1.412	(3)	3.369	(3)	5.98%	(3)	1.711	(6)	4.523	(6)	7.55%	(6)
Size, Shape, Edges, Keypoints	1.387	(2)	3.225	(2)	5.85%	(2)	1.620	(3)	4.127	(3)	7.13%	(2)

Table 3.7: Comparison of features on the UCSD dataset.

Features	Local Features						Holistic Features					
	MAE		MSE		MRE		MAE		MSE		MRE	
Size	2.134	(14)	8.312	(9)	22.18%	(14)	2.061	(4)	7.899	(2)	21.21%	(9)
Shape	3.336	(15)	33.803	(15)	26.02%	(15)	3.339	(15)	33.642	(15)	27.11%	(15)
Edges	1.754	(5)	8.102	(6)	15.81%	(5)	2.599	(10)	14.174	(9)	22.85%	(13)
Keypoints	1.876	(9)	9.184	(13)	17.33%	(9)	2.105	(5)	9.572	(5)	20.54%	(8)
Size, Shape	1.793	(6)	7.723	(5)	17.21%	(8)	1.841	(2)	8.645	(3)	18.76%	(6)
Size, Edges	1.898	(10)	8.640	(10)	18.37%	(10)	2.690	(14)	17.245	(13)	21.50%	(10)
Size, Keypoints	2.012	(13)	7.536	(3)	21.45%	(13)	2.214	(8)	10.031	(6)	21.92%	(11)
Shape, Edges	1.607	(2)	8.171	(7)	13.47%	(2)	2.677	(13)	27.843	(14)	18.95%	(7)
Shape, Keypoints	1.806	(7)	8.792	(12)	16.97%	(7)	1.875	(3)	8.776	(4)	17.72%	(3)
Edges, Keypoints	1.910	(11)	9.841	(14)	16.16%	(6)	2.645	(11)	14.560	(10)	23.84%	(14)
Size, Shape, Edges	1.618	(3)	7.172	(1)	15.03%	(4)	2.149	(7)	12.824	(7)	17.59%	(2)
Size, Shape, Keypoints	1.835	(8)	7.602	(4)	18.85%	(12)	1.833	(1)	7.629	(1)	18.40%	(5)
Size, Edges, Keypoints	1.916	(12)	8.758	(11)	18.81%	(11)	2.655	(12)	16.458	(12)	22.33%	(12)
Shape, Edges, Keypoints	1.584	(1)	8.217	(8)	13.16%	(1)	2.122	(6)	12.912	(8)	17.24%	(1)
Size, Shape, Edges, Keypoints	1.645	(4)	7.437	(2)	14.95%	(3)	2.283	(9)	14.699	(11)	18.15%	(4)

Table 3.8: Comparison of features on the PETS 2009 dataset.

Features	Local Features						Holistic Features					
	MAE		MSE		MRE		MAE		MSE		MRE	
Size	1.094	(14)	2.079	(14)	18.05%	(14)	1.125	(14)	2.124	(14)	18.86%	(14)
Shape	1.223	(15)	2.670	(15)	20.34%	(15)	1.449	(15)	3.664	(15)	24.88%	(15)
Edges	0.941	(8)	1.535	(9)	15.39%	(5)	0.928	(1)	1.402	(1)	15.98%	(3)
Keypoints	1.001	(12)	1.711	(12)	16.58%	(11)	0.986	(7)	1.602	(7)	16.44%	(6)
Size, Shape	0.980	(11)	1.655	(11)	16.81%	(12)	1.083	(13)	2.062	(13)	18.05%	(13)
Size, Edges	0.950	(10)	1.557	(10)	15.41%	(7)	0.958	(5)	1.513	(4)	16.02%	(4)
Size, Keypoints	1.014	(13)	1.748	(13)	17.01%	(13)	1.005	(10)	1.738	(12)	16.81%	(9)
Shape, Edges	0.939	(7)	1.486	(5)	15.82%	(9)	1.013	(11)	1.721	(10)	17.00%	(11)
Shape, Keypoints	0.911	(2)	1.413	(2)	15.14%	(1)	0.933	(2)	1.496	(3)	15.82%	(1)
Edges, Keypoints	0.931	(5)	1.486	(6)	15.39%	(6)	0.954	(4)	1.475	(2)	16.30%	(5)
Size, Shape, Edges	0.943	(9)	1.496	(7)	15.89%	(10)	1.017	(12)	1.737	(11)	17.09%	(12)
Size, Shape, Keypoints	0.912	(4)	1.427	(4)	15.29%	(4)	0.944	(3)	1.525	(5)	15.93%	(2)
Size, Edges, Keypoints	0.936	(6)	1.508	(8)	15.44%	(8)	0.976	(6)	1.560	(6)	16.49%	(7)
Shape, Edges, Keypoints	0.911	(3)	1.426	(3)	15.21%	(3)	1.002	(9)	1.693	(9)	16.83%	(10)
Size, Shape, Edges, Keypoints	0.899	(1)	1.373	(1)	15.17%	(2)	0.998	(8)	1.674	(8)	16.80%	(8)

Table 3.9: Comparison of features on the Fudan dataset.

Features	Local Features						Holistic Features					
	MAE		MSE		MRE		MAE		MSE		MRE	
Size	2.959	(13)	13.631	(13)	9.85%	(13)	3.047	(10)	14.410	(10)	10.09%	(10)
Shape	3.912	(15)	24.856	(15)	12.64%	(15)	4.317	(12)	31.496	(12)	13.31%	(12)
Edges	2.519	(5)	10.404	(5)	8.26%	(5)	15.853	(14)	516.369	(14)	50.25%	(15)
Keypoints	2.448	(2)	9.593	(1)	8.12%	(3)	2.803	(1)	12.796	(1)	8.80%	(1)
Size, Shape	3.028	(14)	14.621	(14)	9.97%	(14)	9.636	(13)	266.395	(13)	30.34%	(13)
Size, Edges	2.610	(9)	11.004	(9)	8.43%	(8)	15.922	(15)	518.101	(15)	49.68%	(14)
Size, Keypoints	2.498	(4)	9.940	(3)	8.20%	(4)	2.872	(3)	12.816	(2)	9.33%	(5)
Shape, Edges	2.659	(11)	11.510	(11)	8.60%	(11)	2.887	(4)	13.316	(5)	9.18%	(3)
Shape, Keypoints	2.581	(7)	10.995	(8)	8.48%	(9)	2.811	(2)	12.929	(4)	8.85%	(2)
Edges, Keypoints	2.436	(1)	9.629	(2)	8.08%	(1)	2.940	(8)	13.510	(8)	9.39%	(8)
Size, Shape, Edges	2.693	(12)	11.861	(12)	8.74%	(12)	3.113	(11)	14.842	(11)	10.21%	(11)
Size, Shape, Keypoints	2.614	(10)	11.187	(10)	8.53%	(10)	2.925	(7)	13.452	(7)	9.34%	(6)
Size, Edges, Keypoints	2.479	(3)	10.109	(4)	8.08%	(2)	2.970	(9)	13.400	(6)	9.71%	(9)
Shape, Edges, Keypoints	2.555	(6)	10.639	(6)	8.27%	(6)	2.917	(6)	13.561	(9)	9.30%	(4)
Size, Shape, Edges, Keypoints	2.584	(8)	10.965	(7)	8.34%	(7)	2.893	(5)	12.919	(3)	9.38%	(7)

Table 3.10: Comparison of features on the **Mall** dataset.

Features	Local Features					
	MAE		MSE		MRE	
Size	9.046	(9)	130.667	(9)	5.91%	(10)
Shape	12.129	(14)	261.593	(14)	8.09%	(14)
Edges	14.428	(15)	318.831	(15)	9.27%	(15)
Keypoints	9.693	(11)	158.345	(12)	5.92%	(11)
Size, Shape	9.042	(8)	133.305	(10)	5.40%	(9)
Size, Edges	8.705	(7)	118.288	(8)	5.21%	(7)
Size, Keypoints	5.454	(1)	68.025	(1)	3.49%	(1)
Shape, Edges	10.490	(13)	139.443	(11)	6.53%	(13)
Shape, Keypoints	6.756	(2)	85.028	(4)	4.48%	(3)
Edges, Keypoints	10.473	(12)	205.206	(13)	6.39%	(12)
Size, Shape, Edges	9.103	(10)	115.980	(7)	5.38%	(8)
Size, Shape, Keypoints	7.916	(6)	90.316	(6)	4.94%	(6)
Size, Edges, Keypoints	7.213	(3)	80.558	(2)	4.40%	(2)
Shape, Edges, Keypoints	7.880	(5)	84.444	(3)	4.76%	(5)
Size, Shape, Edges, Keypoints	7.833	(4)	85.056	(5)	4.75%	(4)

Table 3.11: Comparison of features on the **Grand Central dataset** using local features. Holistic features are omitted as training failed due to insufficient data.

In order to pool results across five datasets, mean error rates are not appropriate because some datasets have greater fluctuations in their error rates than others. For example, the Grand Central dataset has massive variability in terms of MSE, and therefore the feature set which performs best on the Grand Central dataset will dominate the average MSE across all datasets.

Instead, features are ranked from 1 to 15 on each dataset, as shown in Tables 3.7-3.11, and the *average rank* across all datasets is reported in Table 3.12 for each feature set. For example, “Shape” alone obtains an average ranking of 14.8 out of 15, indicating a consistently poor performance for this feature across all five datasets. By contrast, when *all features* are used, an average rank of 3.80 is observed for MAE, 3.40 for MSE and 3.60 for MRE.

Features	Average Rank						
	Local Features			Holistic Features			MRE
	MAE	MSE		MAE	MSE		
Size	12.60	11.60	12.80	9.75	9.25	11.00	
Shape	14.80	14.80	14.80	13.00	13.00	13.00	
Edges	8.00	8.00	7.60	10.00	9.75	11.50	
Keypoints	9.60	10.40	9.60	5.50	5.50	6.00	
Size, Shape	10.20	10.40	11.00	10.50	10.75	11.25	
Size, Edges	8.40	8.60	7.40	9.00	8.50	7.75	
Size, Keypoints	8.00	5.80	8.00	7.00	6.75	8.25	
Shape, Edges	7.40	7.60	7.80	8.25	8.50	6.25	
Shape, Keypoints	5.60	7.20	6.00	4.75	5.75	5.00	
Edges, Keypoints	6.80	8.40	6.20	9.00	8.25	9.75	
Size, Shape, Edges	7.00	5.60	7.00	7.75	7.50	6.50	
Size, Shape, Keypoints	7.80	7.00	8.60	4.75	5.25	5.00	
Size, Edges, Keypoints	6.40	6.60	6.00	7.75	7.00	8.25	
Shape, Edges, Keypoints	3.60	4.60	3.60	6.75	8.00	5.25	
Size, Shape, Edges, Keypoints	3.80	3.40	3.60	6.25	6.25	5.25	

Table 3.12: Average rank across **all datasets**, when features are ranked from 1 to 15 on each dataset. Values shown are not actual error rates, but rather an average ranking. (Note that the average rank for *holistic* features does not include the Grand Central dataset.)

Average ranking provides a much clearer picture than any individual dataset taken alone. It becomes particularly clear that when more features are used, the average ranking across all datasets is improved. There is a significantly higher ranking observed for these feature sets:

- Shape, Edges, Keypoints
- Size, Shape, Edges, Keypoints (*all features*)

The ranking of these feature sets are noticeably better than all of the other fea-

ture sets, and there is little difference between them. The full feature vector (size, shape, edges, keypoints) suffers a reduction in performance if any feature is omitted, *except* for size. As the omission of size does not detract from performance, this suggests that it may not be as crucial in achieving optimal performance as previously expected.

This is a surprising result because the size feature is a direct measure of foreground pixels, and foreground pixel counting has formed the basis of most traditional algorithms in the literature. It suggests that foreground detection provides a means for *segmenting* an image (for the purpose of localisation), but the actual *size* of these segments is not a critical feature for crowd counting. In occluded and complicated crowd scenes, the presence of edges, keypoints and shape cues within those segments appear to be the most important features.

Nevertheless, segment size does provide an intuitive measurement of the physical space occupied by a group, even if it is not a critical feature in these experiments. Size does not *detract* from optimal performance, and therefore it is included in the full feature vector for the subsequent analyses.

Figure 3.21 plots the estimate of the proposed algorithm (using the full feature vector) against the ground truth for the UCSD dataset. The equivalent holistic system is also plotted. Each of the subsets from the 5-fold cross validation procedure are shown (these subsets were summarised in Table 3.4). Similarly, Figure 3.22 plots the PETS 2009 dataset, Figure 3.23 plots the Fudan dataset, Figure 3.24 plots the Mall dataset and Figure 3.25 plots the Grand Central dataset. These figures indicate that the proposed algorithm provides accurate crowd counting results across a wide range of conditions.

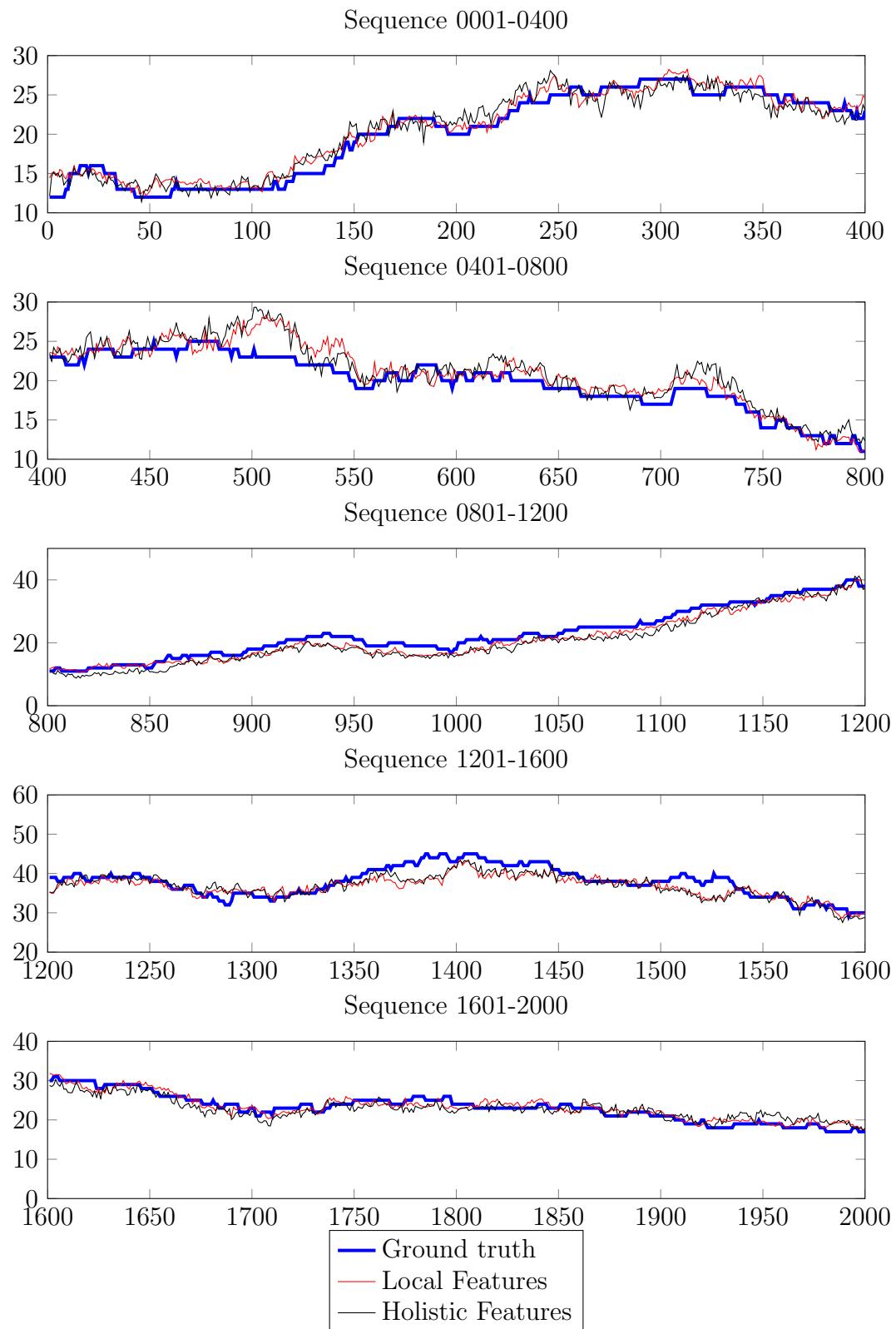
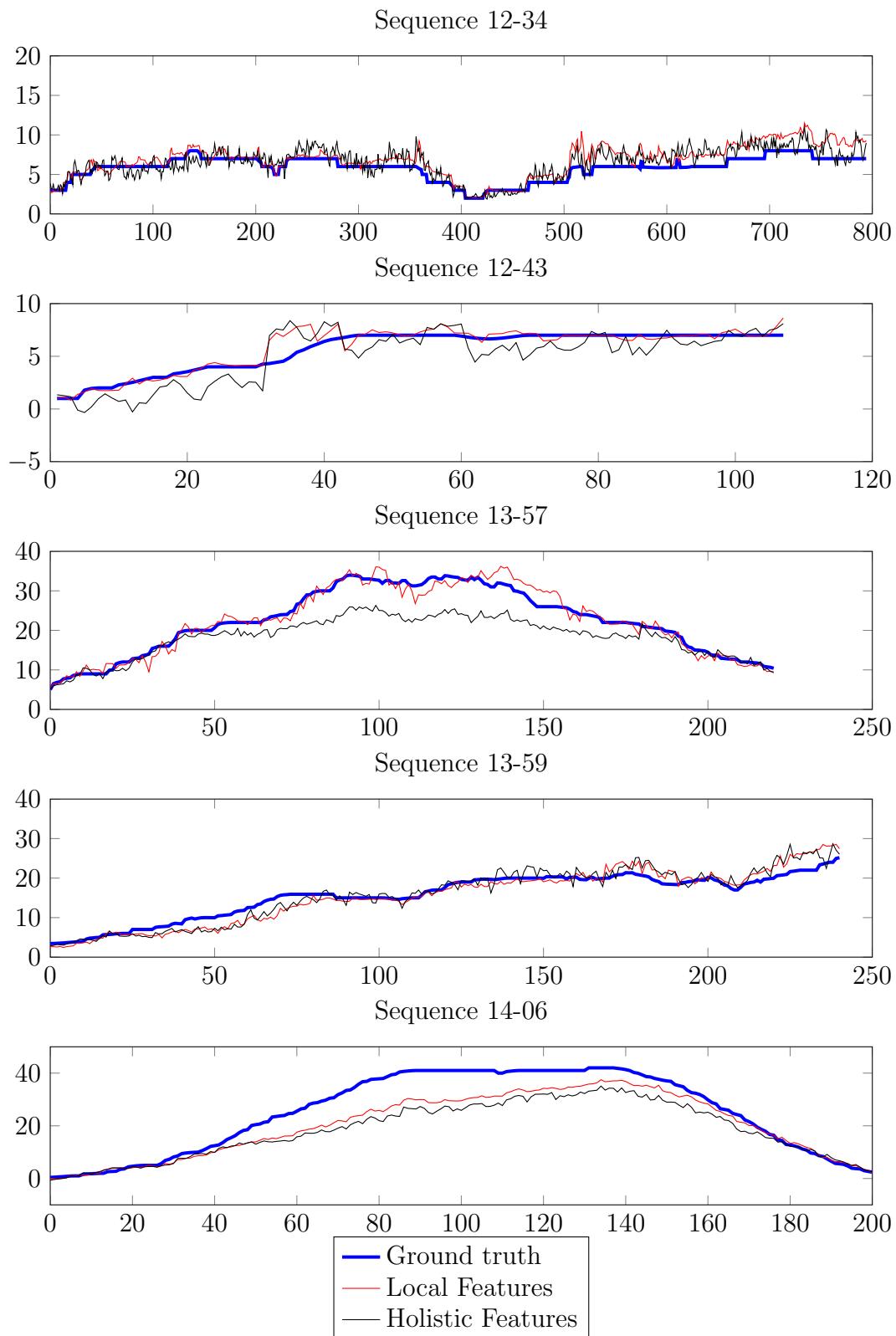


Figure 3.21: Cross validation results on the **UCSD** dataset.

Figure 3.22: Cross validation results on the **PETS 2009 dataset**.

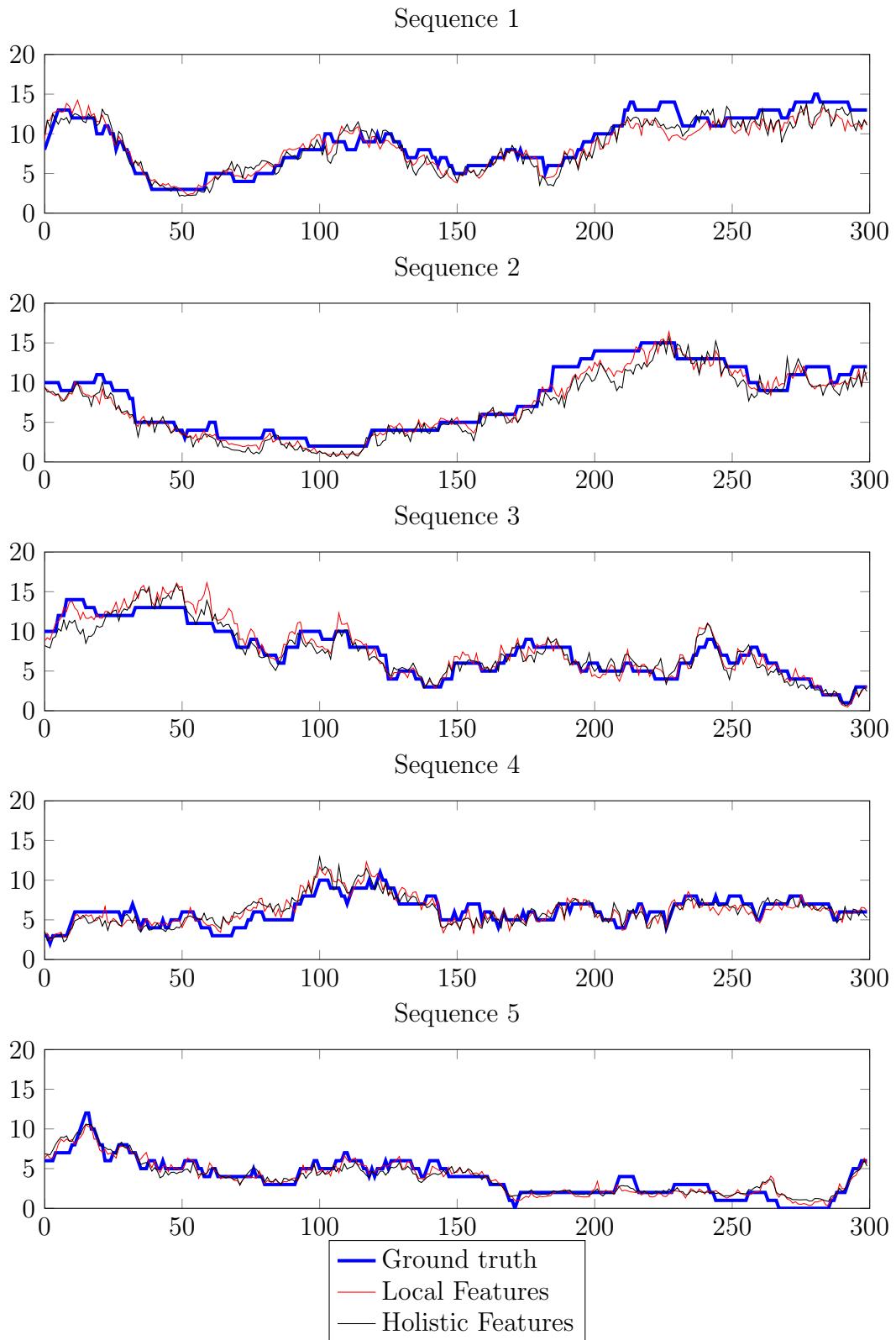
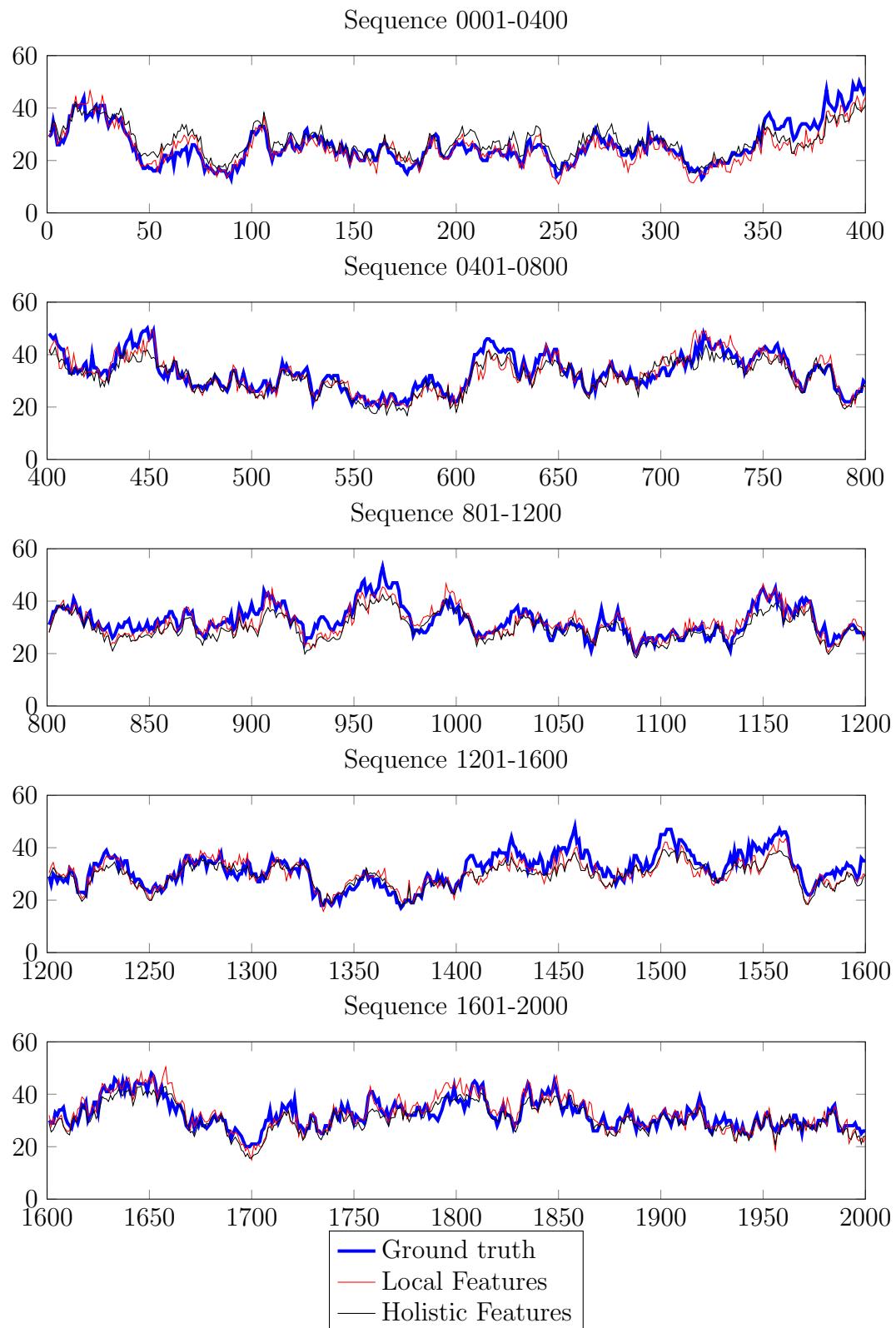


Figure 3.23: Cross validation results on the **Fudan** dataset.

Figure 3.24: Cross validation results on the **Mall dataset**.

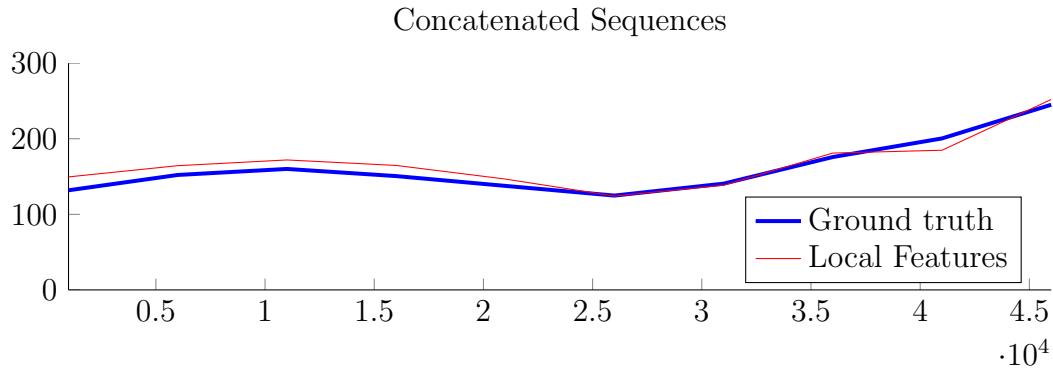


Figure 3.25: Cross validation results on the **Grand Central** dataset. Sequences are concatenated into a single plot due to their short length.

3.3.3.2 Regression Models

In this section, K -fold cross validation is used to evaluate a number of regression models used to perform the crowd estimation. For comparison we use Gaussian process regression (GPR), ordinary least squares linear regression (Linear), K -Nearest Neighbours (KNN) with $K = 1, 2, 4, 8, 16, 32$, and a neural network (NN) with a Sigmoid activation function and one hidden input layer (containing 4, 8, 16 or 32 neurons). In total there are 12 regression models with various parameters. Note that in some cases training fails with the neural network model and large error values are reported in these instances.

Table 3.13 summarises the results for the UCSD dataset. Error rates are ranked from 1 to 12 in each column, with 1 corresponding to the most accurate regression model and 12 the least accurate. The GPR and linear models provide most accurate performance on the UCSD dataset (for both the local and holistic features).

Table 3.14 presents the results for the PETS 2009 datasets. Once again the GPR model provides superior performance, followed by linear regression. This is also seen for the Fudan dataset (Table 3.15) and Mall dataset (Table 3.16). While

the neural network and K -nearest neighbours approaches provide good results in some instances, their performance is sporadic and less reliable than the GPR model.

The Grand Central dataset is summarised in Table 3.17 for local features only. Best results are seen from KNN with $K = 2$, followed by GPR.

Results across all datasets are pooled in Table 3.18 and the average rank is reported. As expected, Gaussian process regression outperforms the other models with an average ranking of 1.4 out of 12 for MAE, and 1.2 for both MSE and MRE. Similarly for holistic features, GPR achieves an average rank of 1.75 for all error metrics.

These results provide very strong support for the use of Gaussian process regression in both local and holistic crowd counting systems.

Regression Model	Local Features					Holistic Features						
	MAE		MSE		MRE	MAE		MSE		MRE		
GPR	1.387	(1)	3.225	(1)	5.85%	(1)	1.620	(2)	4.127	(2)	7.13%	(2)
Linear	1.525	(2)	3.921	(2)	6.38%	(2)	1.582	(1)	3.999	(1)	6.98%	(1)
KNN (K=1)	2.769	(5)	17.415	(3)	10.16%	(6)	2.287	(7)	7.999	(6)	9.68%	(7)
KNN (K=2)	2.710	(4)	17.852	(4)	9.73%	(4)	2.096	(4)	6.707	(4)	8.89%	(4)
KNN (K=4)	2.672	(3)	18.667	(5)	9.42%	(3)	2.229	(5)	8.623	(7)	8.93%	(5)
KNN (K=8)	2.856	(6)	22.569	(6)	9.84%	(5)	2.762	(9)	15.752	(9)	10.12%	(8)
KNN (K=16)	3.096	(7)	27.260	(7)	10.43%	(7)	3.460	(11)	26.085	(11)	12.81%	(10)
KNN (K=32)	3.734	(8)	38.680	(9)	12.27%	(8)	4.326	(12)	39.046	(12)	16.87%	(12)
NN (4)	5.969	(10)	61.329	(10)	25.99%	(10)	2.295	(8)	8.896	(8)	9.46%	(6)
NN (8)	4.280	(9)	29.963	(8)	17.78%	(9)	1.676	(3)	4.517	(3)	7.34%	(3)
NN (16)	10.677	(11)	222.225	(11)	54.46%	(11)	3.110	(10)	15.968	(10)	14.54%	(11)
NN (32)	12.812	(12)	594.049	(12)	66.76%	(12)	2.240	(6)	7.814	(5)	10.37%	(9)

Table 3.13: Comparison of regression models on the **UCSD dataset**.

Regression Model	Local Features						Holistic Features					
	MAE		MSE		MRE		MAE		MSE		MRE	
GPR	1.645	(1)	7.437	(2)	14.95%	(1)	2.283	(1)	14.699	(2)	18.15%	(1)
Linear	1.703	(2)	6.835	(1)	16.82%	(2)	2.289	(2)	14.714	(3)	19.48%	(2)
KNN (K=1)	2.862	(3)	34.719	(4)	18.47%	(5)	2.724	(8)	20.612	(10)	22.28%	(4)
KNN (K=2)	2.913	(4)	39.647	(5)	18.33%	(4)	2.625	(6)	19.043	(7)	22.10%	(3)
KNN (K=4)	2.922	(5)	42.815	(6)	18.01%	(3)	2.730	(9)	20.126	(8)	23.85%	(5)
KNN (K=8)	3.150	(6)	50.862	(7)	18.83%	(6)	2.836	(10)	20.440	(9)	26.39%	(7)
KNN (K=16)	3.488	(8)	59.794	(9)	20.31%	(7)	3.202	(11)	22.495	(11)	35.15%	(11)
KNN (K=32)	3.790	(9)	67.094	(10)	21.83%	(8)	4.095	(12)	32.038	(12)	52.40%	(12)
NN (4)	10.911	(12)	171.639	(12)	126.42%	(11)	2.722	(7)	17.552	(6)	28.83%	(9)
NN (8)	4.345	(10)	54.256	(8)	32.22%	(9)	2.614	(5)	15.857	(4)	33.07%	(10)
NN (16)	3.481	(7)	25.803	(3)	34.60%	(10)	2.433	(4)	17.076	(5)	24.70%	(6)
NN (32)	10.581	(11)	145.978	(11)	146.64%	(12)	2.390	(3)	11.410	(1)	26.76%	(8)

Table 3.14: Comparison of regression models on the **PETS 2009** dataset.

Regression Model	Local Features				Holistic Features			
	MAE	MSE	MRE		MAE	MSE	MRE	
GPR	0.899 (1)	1.373 (1)	15.17% (1)		0.998 (1)	1.674 (1)	16.80% (1)	
Linear	0.925 (2)	1.433 (2)	15.79% (2)		1.010 (2)	1.743 (4)	17.16% (2)	
KNN (K=1)	1.144 (7)	2.214 (6)	18.96% (7)		1.155 (9)	2.337 (9)	19.69% (9)	
KNN (K=2)	1.046 (5)	1.890 (5)	17.10% (6)		1.044 (6)	1.860 (8)	17.95% (4)	
KNN (K=4)	0.992 (3)	1.755 (3)	15.99% (4)		1.018 (3)	1.735 (3)	17.33% (3)	
KNN (K=8)	1.002 (4)	1.832 (4)	15.90% (3)		1.043 (5)	1.798 (5)	18.16% (5)	
KNN (K=16)	1.117 (6)	2.456 (7)	16.60% (5)		1.258 (10)	2.666 (10)	23.59% (10)	
KNN (K=32)	1.410 (8)	4.176 (9)	19.22% (8)		1.693 (12)	5.177 (12)	32.55% (12)	
NN (4)	4.733 (12)	49.853 (12)	76.75% (12)		1.072 (8)	1.810 (7)	19.10% (6)	
NN (8)	1.627 (10)	4.598 (10)	27.70% (9)		1.065 (7)	1.807 (6)	19.46% (7)	
NN (16)	2.208 (11)	8.627 (11)	43.21% (11)		1.391 (11)	2.902 (11)	28.19% (11)	
NN (32)	1.497 (9)	3.661 (8)	30.24% (10)		1.042 (4)	1.734 (2)	19.46% (8)	

Table 3.15: Comparison of regression models on the **Fudan dataset**.

Regression Model	Local Features				Holistic Features			
	MAE	MSE	MRE		MAE	MSE	MRE	
GPR	2.584 (2)	10.965 (1)	8.34% (1)		2.893 (3)	12.919 (2)	9.38% (3)	
Linear	2.579 (1)	11.062 (2)	8.52% (2)		3.565 (9)	19.278 (9)	12.03% (9)	
KNN (K=1)	3.449 (7)	19.496 (7)	11.22% (7)		3.265 (8)	17.189 (7)	10.30% (7)	
KNN (K=2)	3.048 (5)	15.165 (5)	9.94% (5)		3.003 (6)	14.916 (6)	9.51% (5)	
KNN (K=4)	2.886 (3)	13.506 (3)	9.28% (4)		2.968 (5)	14.223 (5)	9.47% (4)	
KNN (K=8)	2.917 (4)	14.056 (4)	9.20% (3)		3.241 (7)	17.473 (8)	10.40% (8)	
KNN (K=16)	3.249 (6)	18.322 (6)	9.96% (6)		3.757 (10)	23.343 (10)	12.31% (10)	
KNN (K=32)	4.190 (8)	30.479 (8)	12.51% (8)		4.832 (12)	39.388 (12)	15.95% (12)	
NN (4)	11.874 (9)	280.169 (9)	38.88% (9)		2.950 (4)	13.449 (4)	9.62% (6)	
NN (8)	35.470 (12)	4497.871 (12)	113.47% (11)		2.859 (2)	12.734 (1)	9.28% (2)	
NN (16)	35.032 (11)	1693.499 (11)	114.92% (12)		2.840 (1)	13.162 (3)	8.99% (1)	
NN (32)	21.339 (10)	637.392 (10)	71.75% (10)		4.008 (11)	26.848 (11)	14.28% (11)	

Table 3.16: Comparison of regression models on the Mall dataset.

Regression Model	Local Features					
	MAE		MSE		MRE	
GPR	7.833	(2)	85.056	(1)	4.75%	(2)
Linear	28.647	(8)	845.320	(8)	18.10%	(8)
KNN (K=1)	9.909	(4)	127.275	(3)	5.97%	(5)
KNN (K=2)	7.356	(1)	107.895	(2)	3.89%	(1)
KNN (K=4)	9.766	(3)	198.653	(4)	5.08%	(3)
KNN (K=8)	10.670	(5)	214.643	(5)	5.53%	(4)
KNN (K=16)	11.972	(6)	302.815	(6)	6.05%	(6)
KNN (K=32)	21.865	(7)	786.671	(7)	11.92%	(7)
NN (4)	237.877	(12)	92758.839	(12)	149.90%	(12)
NN (8)	111.790	(9)	37767.287	(10)	82.61%	(9)
NN (16)	173.443	(10)	37287.121	(9)	105.08%	(10)
NN (32)	177.374	(11)	56524.009	(11)	107.98%	(11)

Table 3.17: Comparison of regression on the **Grand Central dataset** using local features. Holistic features are omitted as training failed due to insufficient data.

3.3.3.3 Holistic Features

So far only local features have been discussed. This section evaluates holistic features and compares them to the use of local features which are proposed in this thesis.

Table 3.18 lists the average ranking for each regression model, averaged across all datasets. As discussed in Section 3.3.3.2, the results provide strong support for the use of Gaussian process regression for both local and holistic features. Therefore we continue to use GPR in the following analysis.

Table 3.12 (p. 174) lists the average ranking for each feature set, averaged across all datasets. The GPR model was used for these experiments. As indicated in bold, the best feature sets on a holistic level were:

Regression Model	Local Features			Holistic Features		
	MAE	MSE	MRE	MAE	MSE	MRE
GPR	1.40	1.20	1.20	1.75	1.75	1.75
Linear	3.00	3.00	3.20	3.50	4.25	3.50
KNN (K=1)	5.20	4.60	6.00	8.00	8.00	6.75
KNN (K=2)	3.80	4.20	4.00	5.50	6.25	4.00
KNN (K=4)	3.40	4.20	3.40	5.50	5.75	4.25
KNN (K=8)	5.00	5.20	4.20	7.75	7.75	7.00
KNN (K=16)	6.60	7.00	6.20	10.50	10.50	10.25
KNN (K=32)	8.00	8.60	7.80	12.00	12.00	12.00
NN (4)	11.00	11.00	10.80	6.75	6.25	6.75
NN (8)	10.00	9.60	9.40	4.25	3.50	5.50
NN (16)	10.00	9.00	10.80	6.50	7.25	7.25
NN (32)	10.60	10.40	11.00	6.00	4.75	9.00

Table 3.18: Average rank across **all datasets**, when regression models are ranked from 1 to 12 on each dataset. Values shown are not actual error rates, but rather an average ranking. (Note that the average rank for *holistic* features does not include the Grand Central dataset.)

- Shape, Keypoints
- Size, Shape, Keypoints

The average ranking for these feature sets was 4.75 out of 15 for MAE, 5.75 and 5.25 each for MSE, and 5.0 for MRE. Unlike local features, the use of all features does not provide improved accuracy. There is no clear trend between the number of features used and overall performance. For example, the use of *keypoints* alone achieves a similar rank to using *all features*.

It is also worth noting that *edges* do not perform as well on a holistic level, despite their wide usage within the literature.

In order to compare local and holistic features, we select the best-performing

feature sets across both systems, namely:

- Shape, Keypoints
- Size, Shape, Keypoints
- Shape, Edges, Keypoints
- Size, Shape, Edges, Keypoints

The first two feature sets are optimal for holistic systems, while the last two are optimal for local systems. For ease of comparison, the results for these features are taken from Tables 3.7-3.11 and presented concisely in Table 3.19 across all datasets. Each row lists results for a given feature set, and the best result (local vs holistic) is highlighted in bold.

For the Fudan and Mall datasets, local features outperform holistic features in every experiment, including those feature sets which were found to be optimal on a holistic level. For the UCSD and PETS 2009 datasets, results favour local features in 6 of the 8 experiments. There are no experiments in which holistic features outperform local features across all three error metrics.

In each dataset, the best performance is underlined (out of local and holistic systems). In every case the lowest error is observed with local features.

Dataset	Features	Local Features			Holistic Features		
		MAE	MSE	MRE	MAE	MSE	MRE
UCSD	Shape, Keypoints	1.798	5.319	7.37%	7.403	164.653	36.77%
	Size, Shape, Keypoints	1.827	5.453	7.45%	1.757	4.891	7.66%
	Shape, Edges, Keypoints	1.412	3.369	5.98%	1.711	4.523	7.55%
	Size, Shape, Edges, Keypoints	<u>1.387</u>	<u>3.225</u>	<u>5.85%</u>	1.620	4.127	7.13%
PETS 2009	Shape, Keypoints	1.806	8.792	16.97%	1.875	8.776	17.72%
	Size, Shape, Keypoints	1.835	7.602	18.85%	1.833	7.629	18.40%
	Shape, Edges, Keypoints	<u>1.584</u>	8.217	<u>13.16%</u>	2.122	12.912	17.24%
	Size, Shape, Edges, Keypoints	1.645	7.437	14.95%	2.283	14.699	18.15%
Fudan	Shape, Keypoints	0.911	1.413	<u>15.14%</u>	0.933	1.496	15.82%
	Size, Shape, Keypoints	0.912	1.427	15.29%	0.944	1.525	15.93%
	Shape, Edges, Keypoints	0.911	1.426	15.21%	1.002	1.693	16.83%
	Size, Shape, Edges, Keypoints	<u>0.899</u>	<u>1.373</u>	<u>15.17%</u>	0.998	1.674	16.80%
Mall	Shape, Keypoints	2.581	10.995	8.48%	2.811	12.929	8.85%
	Size, Shape, Keypoints	2.614	11.187	8.53%	2.925	13.452	9.34%
	Shape, Edges, Keypoints	<u>2.555</u>	<u>10.639</u>	<u>8.27%</u>	2.917	13.561	9.30%
	Size, Shape, Edges, Keypoints	2.584	10.965	8.34%	2.893	12.919	9.38%

Table 3.19: Comparison of local and holistic features on all datasets using GPR. Each row represents a different feature set, and the best result (local or holistic) is indicated in bold, in terms of MAE, MSE and MRE.

3.3.4 Comparison To Other Algorithms

In this section the performance of the proposed system using local features is assessed in comparison to other algorithms in the literature. The vast majority of existing systems are combinations of holistic features and various regression models, and these algorithms are well represented by the ‘Holistic Features’ system described in Section 3.3.3. Additional systems are evaluated in this section, and we compare results with the following systems:

1. **Kong** - blob size histograms and edge angle histograms (holistic) [103, 104].
2. **Chan** - segmentation using dynamic textures (holistic features) [36, 38].
3. **Lempitsky** - pixel-level features (local) [110].
4. **Conte** - moving SURF points, clustered (local) [55].
5. **Albiol** - moving corner points (holistic) [6, 55].
6. **Chen** - block-level features (local) [39].

Kong [103, 104] has proposed the use of blob size histograms, as described in Equations 2.40-2.42 (pp. 53-55). The use of histogram bins is somewhat different from the ‘local’ and ‘holistic’ systems tested in Section 3.3.3, because the blob segments are detected locally and the histogram features are accumulated holistically. As Kong’s algorithm is one of the most widely cited articles in the crowd counting literature, it is tested directly against the proposed algorithm using the cross validation procedure of Section 3.3.3.

Kong’s algorithm was implemented as faithfully as possible to [103, 104], however some assumptions were necessary. Although Kong used a bin width of 500 for the blob size histogram, this value is not suitable for all datasets due to differences

in image resolution and camera distance with respect to the scene. Instead, the bin width is set to roughly $\frac{2}{3}$ of the size of a person in the scene, so that smaller blobs (noise) are assigned to the first histogram bin and larger groups occupy the other bins. This provides good separation between different blob sizes, as is the intent of the algorithm. The bin width is calculated by positioning a pedestrian template at the centre of an image and taking the sum of weighted pixels belonging to the template, the result of which is multiplied by $\frac{2}{3}$. Six bins are used for the blob size histogram, and eight blobs are used for the edge angle histogram, as proposed by the author in [103, 104].

Kong proposes the use of neural networks and linear regression. For completeness we also evaluate GPR and KNN regression on Kong's feature set. The 5-fold cross validation procedure of Section 3.3.3 is followed and regression models are ranked from 1 to 12 for each dataset. The average rankings across all datasets are reported in Table 3.20. The best performance was seen with the linear model, which ranked 3.0 out of 12 for MAE, 2.5 for MSE, and 4.25 for MRE. The neural networks also ranked very highly when 8 or 16 neurons were used in the hidden layer. These results confirm that linear regression and neural networks are the most appropriate regression models to be used in conjunction with Kong's feature set, as proposed by the author.

Table 3.21 presents the error rates of Kong's system, using linear and NN regression, for each dataset. For comparison the results of the proposed system are shown in Table 3.19 under the heading 'Local Features'. Although Kong's approach performs well for most datasets, the error rates are significantly higher than the proposed system. The mean relative error is less than 20% for the UCSD, Fudan and Mall datasets, and is therefore deemed to be acceptable by Regazzoni's standard [157], however it exceeds this threshold for the PETS 2009 dataset.

Regression Model	MAE	MSE	MRE
GPR	5.00	5.25	5.25
Linear	3.00	2.50	4.25
KNN (K=1)	9.75	10.00	8.00
KNN (K=2)	7.25	7.50	5.75
KNN (K=4)	5.50	6.25	4.25
KNN (K=8)	7.25	8.25	7.75
KNN (K=16)	9.50	10.25	10.00
KNN (K=32)	11.75	11.75	11.75
NN (4)	3.75	2.75	4.75
NN (8)	3.75	2.50	4.50
NN (16)	4.00	4.75	4.00
NN (32)	7.50	6.25	7.75

Table 3.20: Average rank of **Kong’s system** across UCSD, PETS 2009, Fudan and Mall datasets, when regression models are ranked from 1 to 12 on each dataset. Values shown are not actual error rates, but rather an average ranking.

Dataset	Regression Model	MAE	MSE	MRE
UCSD	Linear	1.534	3.837	6.82%
	NN (8)	1.805	5.331	7.73%
	NN (16)	2.549	10.683	10.55%
PETS 2009	Linear	2.379	11.840	23.95%
	NN (8)	2.618	10.040	32.26%
	NN (16)	2.350	13.397	20.92%
Fudan	Linear	1.019	1.721	17.44%
	NN (8)	1.049	1.783	18.16%
	NN (16)	1.052	1.792	19.04%
Mall	Linear	3.417	17.673	11.49%
	NN (8)	3.168	15.673	10.08%
	NN (16)	3.186	15.932	10.25%

Table 3.21: Results of **Kong’s system** across UCSD, PETS 2009, Fudan and Mall datasets. The best-performing regression models are shown (Linear, NN).

Chan [36] proposed the use of holistic features in conjunction with a mixture model of dynamic textures [37]. Segmentation was performed using a mixture of dynamic textures, and Gaussian process regression was used to predict the crowd size moving in each direction (away or towards the camera). A large set of 29 holistic features were used including foreground pixels, edges and textures. The number of pedestrians moving in each direction was then summed to get the overall count. This system is referred to as ‘Chan: away + towards’. Additionally, the segmentation result for each moving class can be combined to obtain a full foreground mask (rather than one for each direction), which is then used to train the system to obtain the overall crowd count directly. This system is referred to as ‘Chan: all’.

For a direct comparison with Chan’s results, the training and testing protocol described in [36] was followed. Adhering to this protocol, frames 600:1399 of the UCSD dataset were used for training the system, while the remaining 1200 frames (1:599 and 1400:2000) were set aside for testing. For reasons discussed in Section 3.3.3, a subset of frames from this training range are selected. Two training sets are used, namely 605:5:1395 (‘Dense’) and 640:80:1360 (‘Sparse’), in accordance with Lempitsky [110].

In contrast to Chan’s holistic system, Lempitsky [110] proposed a highly localised algorithm in which every *pixel* was represented by a feature vector, a local density function was learned so that integrating the density over any region would yield the number of objects in the region.

Table 3.22 presents the results of this protocol on the UCSD dataset, for a number of crowd counting algorithms. Although not as rigorous as cross validation, Chan’s protocol has been used by a number of authors to report their results on this dataset, and therefore the proposed algorithm is evaluated with this procedure to enable a direct comparison. The best performance is observed for the

Algorithm	Details	Training Frames	MAE	MSE	MRE
Kong	Linear	600:1399	1.622	4.094	7.57%
	NN (8)	600:1399	1.412	3.177	6.09%
	NN (16)	600:1399	1.661	4.321	7.72%
Chan	away+towards	600:1399	1.953	5.753	-
	all	600:1399	1.945	6.057	-
Lempitsky		605:5:1400	1.70	-	-
		640:80:1360	2.02	-	-
Chen	RR	601:1400	2.25	7.82	11.01%
	GPR	601:1400	2.24	7.97	11.26%
	MLR	601:1400	2.60	10.1	12.49%
	MORR	601:1400	2.29	8.08	10.88%
Local Features		605:5:1395	1.466	3.339	6.64%
		640:80:1360	1.379	3.210	5.85%
Holistic Features		605:5:1395	1.806	4.879	8.48%
		640:80:1360	1.481	3.661	6.66%

Table 3.22: Results of various systems on the UCSD dataset, following the experimental protocol of Chan [36]. Frames 600:1399 were designated for training and the remaining frames were set aside for testing (1:599, 1400:2000).

proposed algorithm (using all local features and GPR) as well as Kong’s algorithm (using a neural network with 8 neurons in the hidden layer). These are indicated in bold.

Chan also evaluated his algorithm on the PETS 2009 dataset in [38], the results of which are summarised in Table 3.23 for sequences 13-57, 13-59 and 14-06. For comparison, the results of the proposed algorithm (local features), the equivalent holistic features and Kong’s algorithm are also reported in Table 3.23. The proposed algorithm performs best on sequence 13-57, Chan’s algorithm performs best on sequence 13-59 and Kong’s algorithm performs best on sequence 14-06. When averaged across all frames in these sequences, the proposed algorithm per-

forms best in terms of MAE and MRE, while Kong's algorithm performs best in terms of MSE. These results indicate that the proposed methodology performs competitively with existing methods when assessed on individual sequences of the PETS 2009 dataset.

Algorithm	13-57			13-59			14-06			All frames		
	MAE	MSE	MRE	MAE	MSE	MRE	MAE	MSE	MRE	MAE	MSE	MRE
Chan	2.308	8.362	-	1.647	4.087	-	4.328	44.159	-	2.680	17.661	-
Conte	1.920	-	8.70%	2.240	-	17.30%	4.660	-	20.50%	2.867	-	15.40%
Albiol	2.800	-	12.60%	3.860	-	24.90%	5.140	-	26.10%	3.895	-	21.16%
Kong, Linear	4.003	24.344	17.21%	1.746	5.223	12.68%	4.873	41.032	22.90%	3.446	22.453	17.29%
Kong, NN (8)	2.750	10.520	14.09%	2.088	7.665	14.49%	4.120	24.559	22.27%	2.925	13.738	16.72%
Kong, NN (16)	2.655	12.603	11.60%	3.055	12.806	21.74%	6.235	65.358	23.51%	3.886	28.671	18.89%
Local Features	1.327	3.081	6.26%	1.684	4.591	12.19%	4.963	41.449	20.89%	2.559	15.262	12.85%
Holistic Features	4.080	27.168	15.52%	1.678	4.599	11.76%	6.696	71.943	27.16%	4.000	32.539	17.68%

Table 3.23: Results of various systems on the PETS 2009 dataset (sequences 13-57, 13-59 and 14-06). The average performance across all frames of these sequences is also reported.

Algorithm	Details	Training Frames	MAE	MSE	MRE
Kong	Linear	1:800	3.088	13.844	9.58%
	NN (8)	1:800	2.994	13.344	9.46%
	NN (16)	1:800	4.869	30.183	15.11%
Chen	RR	1:800	3.59	19.0	11.09%
	GPR	1:800	3.72	20.1	11.59%
	MLR	1:800	3.90	23.9	11.96%
	MORR	1:800	3.15	15.7	9.86%
Local Features		5:10:795	2.552	10.417	7.90%
		40:80:760	2.645	11.078	8.33%
Holistic Features		5:10:795	3.011	13.933	9.33%
		40:80:760	2.683	11.360	8.24%

Table 3.24: Results of various systems on the Mall dataset, following the experimental protocol of Chen [42]. Frames 1:800 were designated for training and the remaining frames were set aside for testing (801:2000).

Chen [42] proposed block-level local features which were assessed on the Mall dataset; these results are shown in Table 3.24. Frames 1:800 are designated for training while frames 801:2000 are used for testing. This protocol was adopted and the results for the proposed system, equivalent holistic system and Kong’s algorithm are also reported in Table 3.24 for comparison. All of these algorithms outperform Chen’s system, and local features exhibit the lowest error rate across all metrics (MAE, MSE and MRE).

In conclusion, it is evident that the proposed algorithm outperforms existing local and holistic methods across a variety of datasets and testing protocols.

Dataset	Tracking			No tracking		
	MAE	MSE	MRE	MAE	MSE	MRE
UCSD	1.387	3.225	5.85%	1.459	3.528	6.23%
PETS 2009	1.645	7.437	14.95%	1.781	7.932	16.97%
Fudan	0.899	1.373	15.17%	0.916	1.416	15.51%
Mall	2.604	11.133	8.38%	2.584	10.965	8.34%
Grand Central	7.833	85.056	4.75%	8.120	91.064	4.93%

Table 3.25: Evaluation of the proposed algorithm with and without the tracking module. For each dataset the best result (Tracking vs. No tracking) is indicated in bold.

3.3.5 Tracking Algorithm

In this section the tracking algorithm described in Section 3.2.6 is assessed. The cross-validation procedure of Section 3.3.3 is followed using all local features and Gaussian process regression. The procedure is repeated with the tracking module turned on and off, and the results are summarised in Table 3.25.

For the UCSD, PETS 2009, Fudan and Grand Central datasets, the tracking module provides a consistent improvement in performance across all error metrics. For the Mall dataset, however, a decrease in performance is seen. This is due to the lower frame rate of this dataset, which is less than 2 fps (Table 3.5, p. 160). Low frame rates such as this are unsuitable for tracking.

It is concluded that tracking module described in Section 3.2.6 improves performance, provided that the frame rate is sufficiently high (at least 7-10 fps).

Frame range	Description	MAE	MSE	MRE
601 : 1400	Training range	1.14	2.39	4.81%
1401 : 2000	Testing range 1	1.35	2.97	4.76%
23000 : 24000	Testing range 2	0.80	1.38	8.03%
30000 : 32000	Testing range 3	0.48	0.46	13.94%

Table 3.26: Long term performance of the **proposed algorithm** on the UCSD dataset [36]. Performance is stable over the long term compared to textural features (Table 3.1, p. 119).

3.3.6 Long Term Performance

In this section, long term performance of the proposed algorithm is evaluated. A limitation of texture-based algorithms was identified in Section 3.2.3.1 when subtle environmental changes occur over time. The experimental protocol of Section 3.2.3.1 is repeated here using the proposed algorithm. Local features, GPR and group tracking are used.

Frames 601-1400 of the UCSD dataset were used for training, with testing being performed on frame ranges 1401-2000, 23000-24000 and 30000-32000. Table 3.26 presents the results of this analysis. Performance on the training range and testing range 1 are similar, with a MAE of 1.14 and 1.35, respectively. Testing ranges 2 and 3 occur later in the video sequence, and performance is not reduced, with a MAE of 0.80 and 0.48 observed, respectively. The reduction in MAE is due to smaller crowds in these ranges; note that the MRE increases slightly to 8.03% and 13.94% respectively, although this is still acceptable (less than 20%). By contrast, the texture-based system evaluated in Section 3.2.3.1 (Table 3.1, p. 119) has a MRE of 62.70% and 136.03% for these test ranges, which is unacceptably high.

It is concluded that the proposed algorithm does not suffer a significant reduction in performance over the long term compared to the texture-based method. This is

due to the use of the adaptive background model, which can incorporate gradual environmental changes over time (Section 3.2.1).

3.4 Visualisation

The results of the crowd counting algorithm must be presented to the user in an intuitive manner. This section describes the visual output of the proposed algorithm, as well as a method for calculating pixelwise ‘crowding density’ which is later applied to multi camera environments in Chapter 5.

All foreground segments in an image are highlighted to the end user by setting their *perimeter pixels* to red. The group estimate for each segment is rounded to the nearest integer, and non-zero counts are drawn at the *centroid* of the corresponding segment. The holistic estimate is shown at the top of the image, and this may be updated at a regular interval (e.g. every 50th frame) to avoid rapid changes in the estimate and therefore improve readability as part of a video stream. An example is shown in Figure 3.26.

One benefit of the proposed algorithm is the ability to provide *local* counts within an image compared to holistic systems, which only provide a total estimate. One possible application of this is a ‘heat map’, which indicates the relative level of crowding at each location in an image.

The crowd density of each pixel is calculated as follows. Firstly, the foreground detection result at pixel (i, j) is denoted $F(i, j) \in \{0, 1\}$ and the identity of the blob to which this pixel belongs is denoted $C(i, j)$ (Section 3.2.1). The set of pixels belonging to blob $C(i, j)$ is denoted $B_{C(i, j)}$ and the crowd estimate for this blob is denoted $\mu'_{C(i, j)}$ (Section 3.2.6). Finally, the density map for the viewpoint in question is denoted S (Section 3.2.2). Thus the ‘crowding density’ at pixel



Figure 3.26: Visualisation of the proposed algorithm. Local counts are displayed on each blob, and the holistic count is shown at the top of the image.

(i, j) is estimated to be:

$$d(i, j) = F(i, j) \times \frac{S(i, j)}{\sum_{(i', j') \in B_{C(i, j)}} S(i', j')} \times \mu'_{C(i, j)} \quad (3.72)$$

This equation effectively ‘spreads’ the crowd estimate $\mu'_{C(i, j)}$ for blob $C(i, j)$ across all of its pixels, each weighted by the value of the density map S at that pixel. The purpose of this approach is to provide an approximate level of crowding for each pixel within an image. This is utilised within multi camera environments as described in Chapter 5.

3.5 Summary

This chapter proposed a novel crowd counting algorithm based on local features, which are specific to groups and individuals in an image, to estimate the crowd size and its distribution across a scene. Unlike previous systems that have typically employed holistic features, the proposed approach is annotated, trained and tested at a local level. A tracking algorithm was described to further improve the system’s performance. An investigation was conducted to determine the best features and regression models to use for crowd counting. Results demonstrate that the proposed approach consistently outperforms holistic algorithms and existing algorithms in the literature.

The following contributions are noted:

- A new method of calculating the density map (to compensate for perspective) which does not require any manual annotations on the part of the human operator. Instead, the system receives a sequence of images and

uses pedestrian detection to learn the relative sizes of objects at different locations in the image. The density map S is computed automatically.

- A novel approach to crowd counting which replaces holistic features with local features. Instead of counting a crowd globally, the problem is broken down into groups detected using a foreground mask and features are extracted from each segment.
- A unique method of providing local annotations for each foreground segment, based only on simple ‘dot’ annotations which are easy to perform. A pedestrian template model is used to estimate the region occupied by the person represented by each annotation, and ground truth is apportioned to overlapping foreground segments accordingly. This is an automated procedure that separates the annotation stage from the segmentation and processing stage.
- Gaussian process regression was identified as an appropriate regression tool due to its flexibility and ability to provide confidence intervals, and this enables weighted smoothing to be performed as a subsequent step on tracked groups.
- A novel method of refining the estimates using group tracking. Previous approaches have generally ignored the fact that consecutive frames are likely to have similar pedestrian counts, while some have applied smoothing on a holistic level. By identifying groups and tracking them as they move through an image, smoothing can be applied on a local level. The confidence of the estimate provided by GPR is also used to weight each estimate.
- A comprehensive analysis of a crowd counting system across five datasets, capturing various crowding properties: indoor and outdoor, colour and greyscale, high and low resolution images. This is the most comprehensive and rigorous crowd counting analysis performed in the available literature.

The following conclusions were reached as a result of this analysis:

- The use of local features consistently outperformed holistic features.
- A greater quantity of local features generally improved performance compared to fewer features. The best performance was observed with the following feature vectors: ‘size, shape, edges, keypoints’ (all features) and ‘shape, edges, keypoints’. (The omission of size did not improve or detract from the overall performance.)
- The use of Gaussian process regression consistently outperformed linear regression, K -nearest neighbours and neural networks.
- The proposed tracking algorithm improved crowd counting accuracy, except on sequences with low frame rates, such as the Mall dataset (2 fps).

The proposed approach outperforms the baseline equivalent holistic system and the methods of Kong [103], Chan [36], Lempitsky [110], Conte [55], Albiol [6, 55] and Chen [42] on a variety of datasets. The proposed system was demonstrated to be highly accurate, scalable and practical, with very minimal training requirements.

Chapter 4

Scene Invariant Crowd Counting

4.1 Introduction

Existing approaches to crowd counting are scene specific, as they are designed to operate in the same environment that was used to train the system. In a facility containing numerous cameras, this requires each viewpoint to be trained independently, which can be an arduous and time consuming task (it is not practical to supply hundreds of frames of ground truth for every viewpoint).

Existing crowd counting methods often require a large number of frames to be annotated in order to operate successfully. In Chapter 3, an algorithm was described which greatly reduces the number of frames required to train a system by making use of local features, however this algorithm is still scene specific. In this chapter, a novel algorithm is proposed which utilises camera calibration to achieve *scene invariance* by scaling features appropriately between viewpoints. This enables the system to be deployed on different training and testing sets, including those captured at different locations.

In practice, this means that a system can be trained on a bank of ‘reference viewpoints’, and then deployed on any number of *new* viewpoints without any additional ground truth annotations being required. This would greatly reduce the time and cost of a crowd counting system’s setup procedure in real-world circumstances. A depiction of this configuration is shown in Figure 4.1.

The proposed algorithm is tested on seven datasets which utilise camera calibration: PETS 2009, Views 1 and 2 [149]; PETS 2006, Views 3 and 4 [148]; and QUT, Cameras 3, 5 and 8 (Section 4.3.1). These datasets feature crowds of size 1 to 43 people in various lighting conditions and differing camera angles. The system is demonstrated to be scene invariant and capable of supporting multiple cameras, with accurate crowd counting results.

The remainder of this chapter is structured as follows: Section 4.2 outlines the proposed scene invariant crowd counting algorithm, Section 4.3 presents the datasets and experimental results, and Section 4.4 presents conclusions of this research.

4.2 Scene Invariant Crowd Counting

Existing approaches to crowd counting are *scene-specific*, as they are designed to operate in the same environment that was used to train the system. In this section, the use of camera calibration is proposed to achieve scene invariance by scaling features appropriately between viewpoints. This enables the system to be deployed on different training and testing sets.

The section is structured as follows: Section 4.2.1 describes the camera calibration technique; Section 4.2.2 describes the 3D pedestrian template used to model humans; Section 4.2.3 discusses scene invariant feature normalisation; and Section

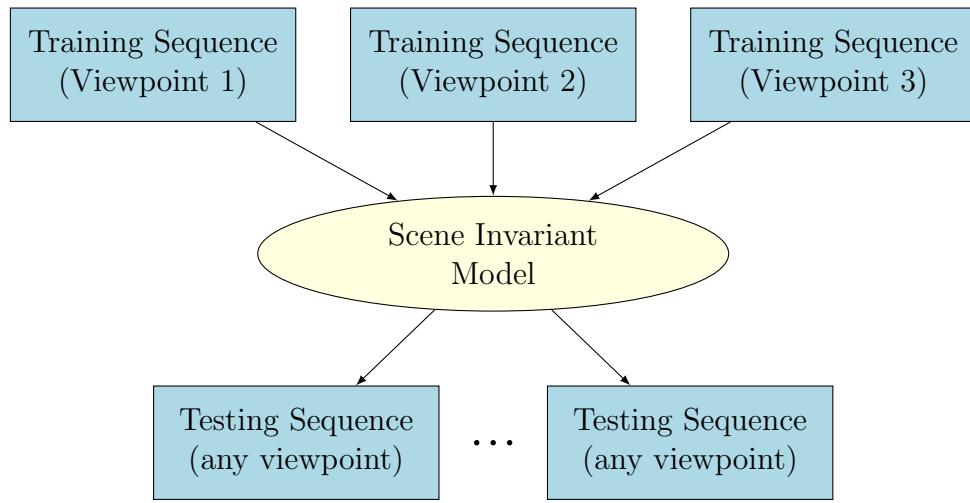


Figure 4.1: A scene invariant crowd counting system is trained on a bank of reference viewpoints, so that it may be deployed on any number of novel viewpoints without additional training being required.

4.2.4 summarises the operation of the algorithm using these components.

4.2.1 Camera Calibration

In order to scale features between viewpoints it is necessary to calibrate each camera so that it is possible to work in real world measurements rather than image pixels.

A number of camera calibration methods have been described in the literature [1, 189, 202], although the most popular of these is Tsai's model [188, 189], which is frequently used on visual surveillance databases such as PETS 2006 and PETS 2009 [148, 149]. Tsai's model incorporates camera position, rotation angle, focal length and radial lens distortion parameters to map between the real world coordinate system and the image plane.

As shown in Figure 4.2, the real world's 3D coordinate system is denoted (X_w, Y_w, Z_w) with origin O_w . The (X_w, Y_w) plane usually represents the ground

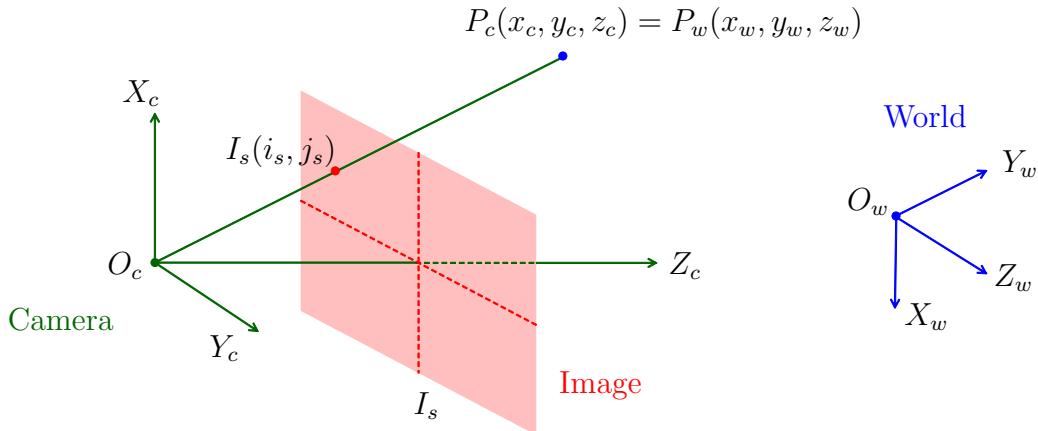


Figure 4.2: The camera calibration model includes the real world’s 3D coordinate system, the camera’s 3D coordinate system and the 2D sensor plane. Tsai’s model also includes radial lens distortion as a subsequent step [189].

plane, with the Z_w axis pointing in the vertical direction, indicating height. The camera’s 3D coordinate system is denoted (X_c, Y_c, Z_c) , with the Z_c axis indicating the direction of the camera.

The camera is located at position O_c which is the origin of the camera’s coordinate system (X_c, Y_c, Z_c) . An arbitrary point in the real-world coordinate system is denoted as $P_w(x_w, y_w, z_w)$, which is equivalent to the point $P_c(x_c, y_c, z_c)$ in the camera’s coordinate system. The relation between the coordinate systems is defined by the rigid body transformation:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T \quad (4.1)$$

where R denotes a 3×3 rotation matrix and T denotes a translation. The rotation

matrix is defined by:

$$R = \begin{bmatrix} \cos \psi \cos \theta & \sin \psi \cos \theta & -\sin \theta \\ -\sin \psi \cos \phi + \cos \psi \sin \theta \cos \phi & \cos \psi \cos \phi + \sin \psi \sin \theta \sin \phi & \cos \theta \sin \phi \\ \sin \psi \sin \phi + \cos \psi \sin \theta \cos \phi & -\cos \psi \sin \phi + \sin \psi \sin \theta \cos \phi & \cos \theta \cos \phi \end{bmatrix} \quad (4.2)$$

where θ , ϕ and ψ denote the yaw, pitch and tilt of the camera respectively.

The point (x_c, y_c, z_c) is then mapped to the 2D coordinate (i_s, j_s) on the *sensor plane* using a perspective projection with pinhole geometry, as shown in Figure 4.2:

$$i_s = f \frac{x_c}{z_c} \quad (4.3)$$

$$j_s = f \frac{y_c}{z_c} \quad (4.4)$$

where f denotes the effective focal length of the camera. (It should be noted that i_s and j_s refer to physical distances in these equations, rather than pixel indices.) Radial lens distortion is responsible for mapping the point (i_s, j_s) in the sensor plane to the *distorted* point (i'_s, j'_s) . Radial lens distortion is approximated by the following equations:

$$i'_s + i'_s \kappa r^2 = i_s \quad (4.5)$$

$$j'_s + j'_s \kappa r^2 = j_s \quad (4.6)$$

where $r = \sqrt{i_s^2 + j_s^2}$. These points are then mapped to the final image coordinate system (i, j) using a linear scaling and offset which effectively moves the origin to the corner of the image.

The extrinsic camera parameters are the camera rotation matrix R and translation T , while the intrinsic camera parameters are f , κ and the linear scaling coefficients used in the final alignment step. These parameters are estimated from a manually-specified set of point correspondences between image pixels and real-world locations on the ground plane [189].

The above procedure can be reversed by inverting these equations, allowing a pixel (i, j) to be mapped back to a real world coordinate (x_w, y_w, z_w) , provided that one of the real world coordinates is known. Typically the z_w coordinate is fixed, for example at $z_w = 0$, which represents a point on the ground plane.

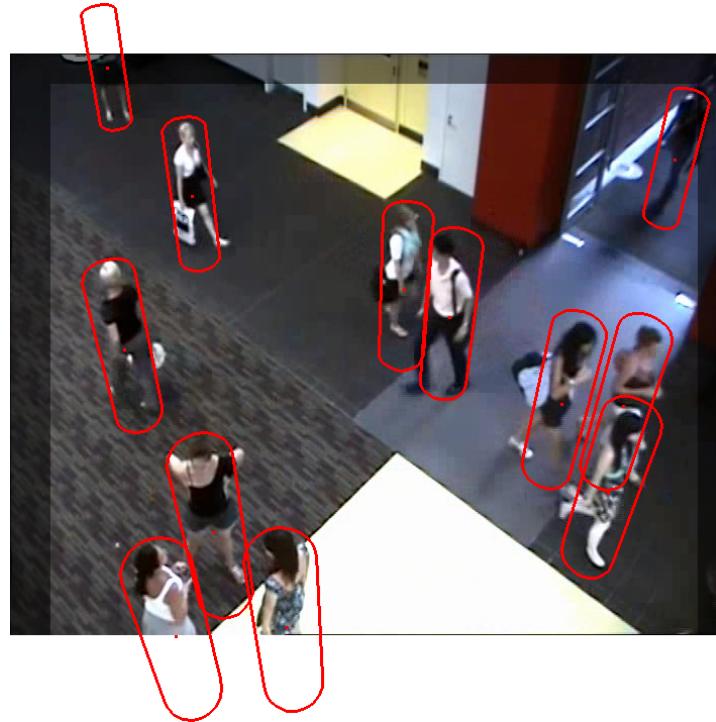
In addition to Tsai’s calibration model, a number of automated procedures exist for estimating camera calibration based on human or object tracking [26, 105, 119]. For example, Lv [119, 120] proposes a camera calibration model using vanishing points, and describes a self-calibration method based on moving humans where the head and feet positions can be located in multiple frames. Similarly, Krahnstoever [105] presents a Bayesian autocalibration algorithm which includes uncertainty estimates for each of the camera parameters. These approaches could readily be incorporated into the proposed framework. However, as Tsai calibration parameters are already available for public visual surveillance datasets, and the method is widely used and well understood, Tsai’s model has been used in this thesis.

4.2.2 3D Pedestrian Template

Section 3.2.2 described the 2D pedestrian template, a rectangle representing the approximate size of a human at any location in an image. In order to scale features between viewpoints accurately it is necessary to use a more accurate pedestrian template based on camera calibration.



(a) QUT, Camera 3.



(b) QUT, Camera 5.

Figure 4.3: Ground truth annotations on the QUT camera network. A camera calibration technique [189] is used to project a human-sized cylindrical object into the scene.

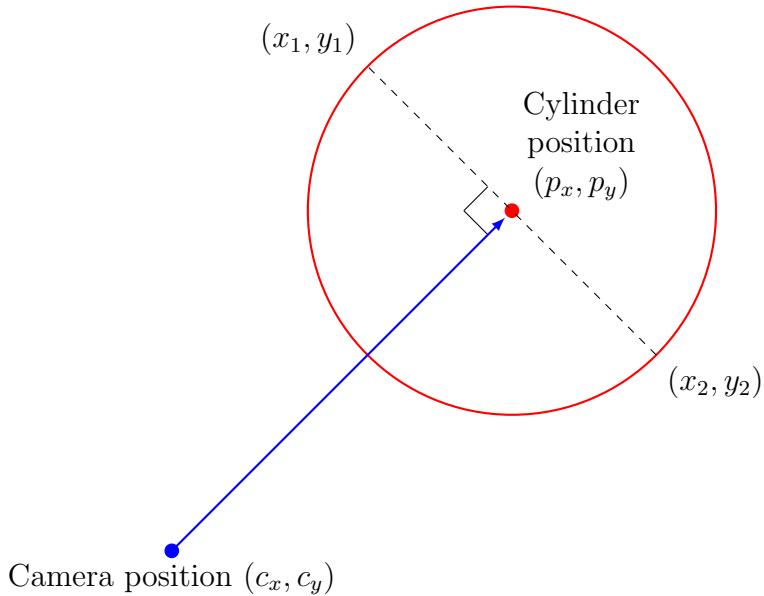


Figure 4.4: Overhead view of the camera position relative to the centre of the cylindrical pedestrian template. Real world coordinates are shown in the (X_w, Y_w) plane.

In the proposed system, a 3D cylinder model of a fixed size is used to approximate the size of a human. The cylinder has a radius of $r = 0.25$ and a height of $h = 1.7$ metres. As depicted in Figure 4.3, this cylinder may be projected into a scene centred around any pixel (i, j) .

The cylinder is projected into the scene as follows. Firstly the image coordinate (i, j) is selected, corresponding to the centre of a person (or a cylinder model representing them). This is converted to the real world position (p_x, p_y, p_z) with $p_z = \frac{h}{2} = 0.85$ at the centre of the model. The top and bottom circles are positioned at a height of $Z_w = h$ and $Z_w = 0$, respectively. Each circle is approximated using a 20-sided polygon, whose vertices are projected back into the image plane and joined together using a number of straight lines, imitating curvature.

The sides of the cylinder model are also drawn using straight lines in the image plane. The start and end points of these lines are determined using basic geometry: Figure 4.4 depicts an overhead view of the camera's position relative

to the cylinder model. When viewed from the camera's perspective, the 'sides' of the cylinder model are located at a distance of $r = 0.25\text{m}$ either side of the cylinder's position (p_x, p_y) , perpendicular to the Z_c axis in the (X_w, Y_w) plane. The sides of the cylinder are denoted (x_1, y_1) and (x_2, y_2) , so that the left side spans from $(x_1, y_1, 0)$ to (x_1, y_1, h) and similarly for the right side. These points are projected into the image plane and then connected, forming the sides of the cylinder which connect to the top and bottom circles. A number of examples are shown in Figure 4.3.

As in Section 3.2.2, the notation $R_{i,j}$ is used to represent the pedestrian template centred around pixel (i, j) , and $|R_{i,j}|$ represents the *area* of this template in the image plane.

4.2.3 Feature Normalisation

As described in Section 3.2.2, the effects of perspective are taken into account by use of a density map which supplies a weight to each pixel. The algorithm presented in this chapter is designed to operate over multiple viewpoints, therefore the features selected to represent a person or group should be invariant to camera position, distance and distortion properties. It is therefore important that features are normalised appropriately so that the trained system can count objects within the scene as well as objects in other scenes. This chapter proposes the use of camera calibration to achieve this normalisation.

A common approach to perspective normalisation is to calculate a density map which assigns to each pixel a weight to compensates for perspective [36, 38, 121, 144]. Typically, a reference pixel near the bottom of the image is assigned the value 1.0 and all other pixels are weighted with respect to this reference. For example, pixels higher in the image will be given a larger value because they

represent a greater area in the scene.

In this chapter, a cylinder model is proposed to approximate the size of a human, with radius $r = 0.25$ and height $h = 1.7$ metres. This pedestrian template may be projected into a scene centred around any pixel (i, j) , as described in Section 4.2.2. The area of this projected shape in the image plane is denoted:

$$A(i, j) = |R_{i,j}| \quad (4.7)$$

The density map is constructed using the inverse of this cylinder model's projected area, at every location within the image:

$$S(i, j) = \frac{1}{A(i, j)} \quad (4.8)$$

Therefore the value of the density map at any location is based on the size of a *fixed* real world object, centred at that location. This density map provides a weight to each pixel so that an object occupying a set of pixels, B , has a weighted area of $\sum_{(i,j) \in B} S(i, j)$. Consequently distant objects occupying fewer pixels are compensated by their larger weights in the density map.

The use of camera calibration in constructing the density map, rather than an arbitrary ‘reference’ pixel or shape, is advantageous because it is defined in terms of a fixed real-world object; this approach can scale readily between different camera angles and is inherently scene invariant.

It does not matter that the cylinder model does not match a human size or shape precisely, as its role is only to normalise features across viewpoints. A number of such features are described in Section 3.2.3, including the weighted area of each blob. The density map S is suitable for such two-dimensional features as

area. However, one-dimensional features such as perimeter and edge pixels are also considered, for which the square root of the density map is used instead.

4.2.4 Operation of the Algorithm

The proposed scene invariant algorithm is trained on multiple viewpoints, which serve as a bank of examples, so that future crowd counting can be performed on a new (unseen) dataset.

The image features and ground truth are obtained as described in Sections 3.2.3 and 3.2.4 respectively, using the cylindrical pedestrian template in place of the rectangular template and the density map described in Section 4.2.3.

The regression model and tracking procedure described in Sections 3.2.5 and 3.2.6 respectively remain unchanged. Once trained, the algorithm is tested on an unseen viewpoint without any additional training data being provided for that viewpoint.

Aside from size and shape based features, it is unclear how well other types of features (edges and keypoints) will scale between viewpoints. Consequently these features are re-assessed in Section 4.3 to determine the optimal feature sets for scene invariant crowd counting.

4.3 Experimental Results

This section presents experimental results of the proposed algorithm for scene invariant crowd counting. Section 4.3.1 outlines the datasets used to evaluate the algorithm, and introduces a new crowd counting dataset that has been created

as part of this research; and Section 4.3.2 presents the scene invariant crowd counting results of the proposed system.

4.3.1 Benchmark Datasets

Seven viewpoints were used to evaluate the performance of the proposed algorithm.

1. View 1 from the **PETS 2009** dataset introduced at the Eleventh IEEE International Workshop on Performance Evaluation of Tracking and Surveillance [149].
2. View 2 from the **PETS 2009** dataset.
3. View 3 from the **PETS 2006** dataset introduced at the Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance [148].
4. View 4 from the **PETS 2006** dataset.
5. Camera 3 from the **QUT** camera network.
6. Camera 5 from the **QUT** camera network.
7. Camera 8 from the **QUT** camera network.

The PETS 2009 database, as described in Section 3.3.2, was a benchmark dataset for the Eleventh IEEE International Workshop on Performance Evaluation of Tracking and Surveillance [149]. Sequences from cameras 1 and 2 are used for crowd counting. The video is provided at a resolution of 768×576 pixels and

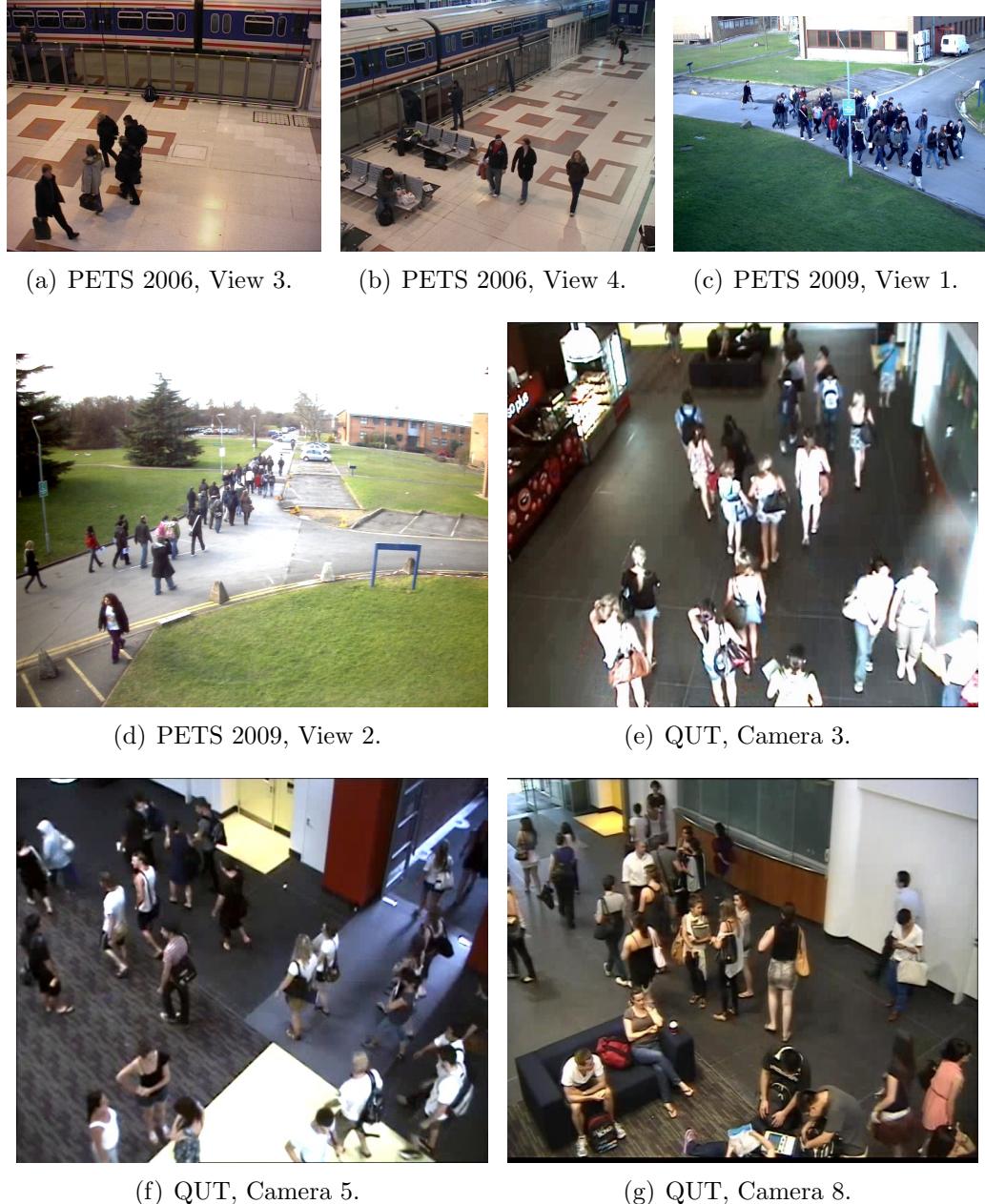


Figure 4.5: Images from each of the seven viewpoints used to assess the performance of the proposed algorithm.

~7 fps in RGB colour format. An example frame from each camera is shown in Figure 4.5.

The PETS 2006 database was released prior to the Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance [148] in order to test object tracking and abandoned luggage detection. As the sequences show pedestrians passing through the scene it is suitable for person counting. Two camera viewpoints are suitable for performing automated counting: View 3 and View 4. The video is provided at a resolution of 720×576 pixels and 25 fps in RGB colour format. An example frame from each camera is shown in Figure 4.5.

To supplement the existing public data sets, a new database has been developed containing footage obtained from the Queensland University of Technology's campus. This database is referred to as 'QUT' and contains data selected from a camera network installed on a single floor of a building.

This database contains three challenging viewpoints, which are referred to as Camera 3 (Figure 4.5(e)), Camera 5 (Figure 4.5(f)) and Camera 8 (Figure 4.5(g)). The sequences contain reflections, shadows and difficult lighting fluctuations, which makes crowd counting difficult.

Previous crowd counting datasets have been substantially shorter in length than those included in the QUT database. For example, PETS 2009 contains crowd counting sequences just over 200 frames in length, while the UCSD dataset (Section 3.3.2) contains 2000 consecutively annotated frames. Although these resources are extremely valuable for testing crowd counting algorithms, they do not adequately capture the long-term performance of a system over varying conditions. For example, if a system performs poorly on one particular frame, it is likely that the preceding and subsequent frames will suffer from the same vulnerability. On shorter sequences such as the PETS 2009 datasets, this may lead

to biased results that do not adequately describe a system’s true performance capabilities.

In order to combat this potential problem, the QUT datasets are annotated at more sparse intervals: typically 100-400 frames apart depending on crowd variation and sequence length. Testing is then performed by comparing the crowd size estimate to the ground truth at these sparse intervals, rather than at every frame. This closely resembles the intended real-world application of this technology, where an operator may periodically ‘query’ the system for a crowd count. Although the human operator does not require this information from *every* frame, the system should provide accurate results whenever it is requested.

Due to the difficulty of the environmental conditions in these scenes, the first 400-500 frames of each sequence is set aside for learning the background model. This is a requirement for proper operation of many multi-modal algorithms such as Stauffer-Grimson [178], Zivkovic [207, 209] and Denman [59, 60], which are used widely in the computer vision field. Existing databases generally do not provide time to learn the background, and although PETS 2009 provides some detached background sequences, they do not immediately precede the crowd counting sequences to be tested, limiting their usefulness.

The seven datasets are summarised in Tables 4.1-4.3.

4.3.2 Cross Validation Experiments

In this section scene invariance is tested using the seven calibrated datasets from Table 4.3. In each experiment one viewpoint was withheld for testing, and the remaining six viewpoints were used for training. A subset of annotated frames from each viewpoint was selected for training. Testing was then performed on the

	PETS-09, V1	PETS-09, V2	PETS-06, V3	PETS-06, V4
Length (frames)	1565	221	6422	6422
Frame Rate (fps)	~7	~7	25	25
Resolution	768×576	768×576	720×576	720×576
Colour	RGB	RGB	RGB	RGB
Location	Outdoor	Outdoor	Indoor	Indoor
Shadows	Yes	Yes	Yes	Yes
Reflections	No	No	No	No
Loitering	No	No	Yes	Yes
Crowd Size	0 to 42	0 to 43	0 to 5	0 to 7

Table 4.1: Summary of the various conditions in the PETS benchmark datasets [148, 149].

	QUT, Camera 3	QUT, Camera 5	QUT, Camera 8
Length (frames)	3100	10000	6100
Frame Rate (fps)	25	25	25
Resolution	704×576	352×288	704×576
Colour	RGB	RGB	RGB
Location	Indoor	Indoor	Indoor
Shadows	Yes	Yes	Yes
Reflections	No	No	Yes
Loitering	Yes	Yes	Yes
Crowd Size	4 to 21	3 to 23	2 to 20

Table 4.2: Summary of the various conditions in the QUT benchmark datasets.

Dataset	Sequence	Crowd Size
PETS 2009, View 1	12-34	2 to 8
	12-43	1 to 7
	13-57	5 to 34
	13-59	3 to 25
	14-06	0 to 42
PETS 2009, View 2	13-57	9 to 43
PETS 2006, View 3	S1	0 to 5
	S5	0 to 4
PETS 2006, View 4	S1	0 to 6
	S5	0 to 7
QUT, Camera 3	10-49	4 to 21
QUT, Camera 5	17-51	3 to 23
QUT, Camera 8	17-26	7 to 20
	19-43	2 to 10

Table 4.3: The benchmark datasets used to evaluate the proposed scene invariant crowd counting algorithm.

remaining viewpoint, using all of the annotated ground truth frames to calculate the error metrics described in Section 3.3.1 (p. 151).

Because the number of people in each scene was often fractional, we use the ‘soft’ ground truth defined in Equation 3.49 (p. 144). This makes sense when evaluating our algorithm because it makes use of local features and has been annotated with occasionally fractional counts (Section 3.2.4). A blob’s ground truth does not jump directly from 0 to 1 (or vice versa) when entering or exiting a scene, for example.

The remainder of this section is structured as follows. Section 4.3.2.1 evaluates the performance of various image features and Section 4.3.2.2 evaluates several regression models. As there were no scene invariant crowd counting algorithms found in the available literature it is not possible to compare with existing algorithms.

4.3.2.1 Feature Evaluation

In Section 3.3.3.1, various feature types were evaluated for crowd counting using local features within a single viewpoint. In this section the analysis is repeated across different viewpoints in order to determine which features are suitable for *scene invariant* crowd counting.

As summarised in Table 3.2 (p. 135), features are divided into *size*, *shape*, *edge* and *keypoint* types. Various combinations of features are assessed in order to determine the most accurate feature types. The following cross validation procedure is used:

1. Seven datasets are annotated with ground truth data.

2. One dataset is selected and set aside for testing. The proposed crowd counting system is trained using ground truth from the *other* six datasets. Testing is then performed on the selected test dataset. Error metrics are averaged across all frames: mean absolute error (MAE), mean square error (MSE) and mean relative error (MRE) are reported.
3. This procedure is repeated for all datasets, by rotating the training and testing sets as depicted in Figure 3.19 (p. 161). Error metrics are then combined across all datasets using average rank and average error, with equal weighting given to each dataset.

The results for View 1 of the PETS 2009 dataset are presented in Table 4.4. The best performance is observed when a greater number of features are used, which is consistent with the observations made previously in Chapter 3. It is noted that performance on this dataset is actually better here than in the previous chapter (see Table 3.8, p. 170), even though the system is being trained on different viewpoints. For example, when all features are used (size, shape, edges and keypoints), the mean absolute error has been reduced from 1.645 to 1.321 and the mean relative error is reduced from 14.95% to 10.32% (when comparing Table 3.8 and Table 4.4). This is most likely due to the greater quantity of training data and wider variety in the group samples which are available in this experiment.

The results for View 2 of the PETS 2009 dataset are also presented in Table 4.4. Good performance is generally seen with more features, except when ‘size’ is omitted from the feature vector, suggesting that object size is a key feature when performing scene invariant crowd counting. Once again, the use of all features ranks amongst the top three feature vectors, as indicated in bold. When all features are used the mean relative error is 9.55%.

Table 4.5 presents the results for Views 3 and 4 of the PETS 2006 dataset. Due

to the relatively small crowds observed in these sequences, the mean relative error is somewhat higher than usual, although the mean absolute error is very small. Some shadows are misclassified as foreground in View 3, resulting in ‘size’ being a less reliable feature for this dataset. In view 4, good performance is seen when more features (including size) are used.

Finally, the results for the QUT datasets are shown in Tables 4.6-4.8. Generally 2-4 feature types outperform 1 feature, confirming the observation that more features provide more accurate crowd counting performance. Size appears to be a critical feature on these datasets. Note, for example, that it has been included in all of the top-performing feature vectors in these QUT datasets. This is in contrast to our conclusions reached in Chapter 3 (Section 3.3.3.1), in which size was not a critical feature.

Features	PETS 2009, View 1						PETS 2009, View 2					
	MAE		MSE		MRE		MAE		MSE		MRE	
Size	2.283	(14)	13.963	(14)	13.91%	(8)	5.892	(7)	40.460	(7)	18.89%	(7)
Shape	3.350	(15)	35.118	(15)	19.01%	(15)	10.187	(12)	164.431	(13)	26.68%	(12)
Edges	1.786	(7)	12.044	(11)	11.43%	(4)	11.307	(15)	173.414	(15)	30.20%	(14)
Keypoints	2.092	(10)	12.926	(12)	15.52%	(14)	10.836	(13)	159.178	(12)	30.76%	(15)
Size, Shape	2.281	(13)	13.701	(13)	15.18%	(11)	6.333	(8)	45.590	(8)	20.37%	(9)
Size, Edges	1.426	(3)	5.456	(4)	10.18%	(2)	3.701	(5)	21.669	(5)	9.93%	(5)
Size, Keypoints	1.792	(8)	8.048	(6)	12.08%	(7)	1.765	(1)	4.795	(1)	6.76%	(1)
Shape, Edges	2.168	(11)	11.843	(10)	14.69%	(9)	9.035	(11)	119.654	(11)	23.57%	(10)
Shape, Keypoints	1.782	(6)	7.486	(5)	15.51%	(13)	7.405	(9)	88.570	(9)	18.91%	(8)
Edges, Keypoints	1.669	(5)	9.744	(8)	11.74%	(6)	11.041	(14)	164.800	(14)	29.56%	(13)
Size, Shape, Edges	1.307	(1)	4.301	(2)	9.83%	(1)	3.944	(6)	23.786	(6)	11.06%	(6)
Size, Shape, Keypoints	1.966	(9)	9.357	(7)	15.38%	(12)	2.095	(2)	6.289	(2)	7.72%	(2)
Size, Edges, Keypoints	1.450	(4)	5.207	(3)	11.60%	(5)	3.595	(4)	19.851	(4)	9.68%	(4)
Shape, Edges, Keypoints	2.176	(12)	11.783	(9)	14.96%	(10)	9.029	(10)	117.467	(10)	23.63%	(11)
Size, Shape, Edges, Keypoints	1.321	(2)	4.250	(1)	10.32%	(3)	3.365	(3)	17.514	(3)	9.55%	(3)

Table 4.4: Comparison of features on the **PETS 2009 datasets, View 1 and View 2**. In each case the system is trained on the six other viewpoints.

Features	PETS 2006, View 3						PETS 2006, View 4					
	MAE		MSE		MRE		MAE		MSE		MRE	
Size	0.580	(12)	0.886	(13)	33.44%	(12)	0.766	(9)	1.190	(8)	27.65%	(7)
Shape	0.604	(14)	0.926	(14)	34.53%	(13)	0.737	(7)	1.248	(9)	27.47%	(6)
Edges	0.299	(2)	0.205	(2)	21.70%	(3)	0.712	(6)	1.137	(7)	28.88%	(8)
Keypoints	0.595	(13)	0.847	(12)	39.60%	(14)	2.108	(15)	8.526	(15)	86.19%	(15)
Size, Shape	0.687	(15)	1.318	(15)	40.45%	(15)	0.608	(5)	0.780	(5)	23.03%	(5)
Size, Edges	0.493	(11)	0.615	(10)	32.41%	(11)	0.746	(8)	0.891	(6)	33.89%	(9)
Size, Keypoints	0.381	(5)	0.343	(4)	23.18%	(4)	1.208	(12)	2.217	(12)	50.87%	(12)
Shape, Edges	0.333	(3)	0.287	(3)	20.73%	(2)	0.476	(2)	0.412	(2)	22.37%	(4)
Shape, Keypoints	0.478	(10)	0.680	(11)	25.98%	(7)	1.429	(14)	3.854	(14)	58.63%	(14)
Edges, Keypoints	0.391	(6)	0.505	(7)	26.13%	(8)	1.214	(13)	2.621	(13)	53.64%	(13)
Size, Shape, Edges	0.463	(9)	0.602	(9)	28.51%	(10)	0.467	(1)	0.364	(1)	21.56%	(2)
Size, Shape, Keypoints	0.374	(4)	0.453	(5)	23.89%	(5)	1.136	(11)	2.171	(11)	47.51%	(10)
Size, Edges, Keypoints	0.418	(8)	0.537	(8)	26.79%	(9)	1.109	(10)	1.780	(10)	49.73%	(11)
Shape, Edges, Keypoints	0.263	(1)	0.180	(1)	16.35%	(1)	0.487	(4)	0.488	(4)	20.92%	(1)
Size, Shape, Edges, Keypoints	0.405	(7)	0.495	(6)	24.98%	(6)	0.487	(3)	0.441	(3)	22.20%	(3)

Table 4.5: Comparison of features on the **PETS 2006 datasets, View 3 and View 4**. In each case the system is trained on the six other viewpoints.

Features	MAE		MSE		MRE	
Size	2.176	(8)	7.065	(8)	17.60%	(8)
Shape	2.469	(9)	9.948	(9)	20.20%	(9)
Edges	4.300	(13)	23.409	(13)	38.48%	(14)
Keypoints	4.573	(15)	33.165	(15)	39.24%	(15)
Size, Shape	1.630	(6)	4.244	(6)	13.11%	(5)
Size, Edges	1.074	(3)	1.944	(3)	10.00%	(3)
Size, Keypoints	1.260	(4)	2.574	(4)	10.58%	(4)
Shape, Edges	4.318	(14)	25.562	(14)	36.22%	(12)
Shape, Keypoints	3.341	(10)	17.932	(10)	26.75%	(10)
Edges, Keypoints	3.942	(11)	19.744	(11)	36.41%	(13)
Size, Shape, Edges	1.811	(7)	4.694	(7)	16.35%	(7)
Size, Shape, Keypoints	1.067	(2)	1.831	(2)	9.42%	(1)
Size, Edges, Keypoints	1.026	(1)	1.712	(1)	9.82%	(2)
Shape, Edges, Keypoints	4.026	(12)	22.769	(12)	33.58%	(11)
Size, Shape, Edges, Keypoints	1.547	(5)	3.506	(5)	14.03%	(6)

Table 4.6: Comparison of features on the **QUT dataset, Camera 3**. The system is trained on the six other viewpoints.

In order to pool these results across all seven datasets, features have been ranked from 1 to 15 on each dataset (as indicated parentheses in Tables 4.4-4.8). The *average rank* for each feature across all datasets is reported in Table 4.9. Consistently poor performance is seen by ‘shape’ features and ‘keypoints’ taken alone, whereas a better ranking is observed when multiple features (including size) are used. Best performance is seen when all features are used. This supports the conclusions reached in Chapter 3, and confirms that these features can be used for scene invariant crowd counting.

Table 4.10 lists the average error rates across all datasets, weighted equally, when all features are used. The mean absolute error is 1.351 and the mean relative error is 15.92%. This falls within the 20% threshold suggested by Regazzoni [157]. The PETS 2006 datasets do exceed this threshold, although this is due primarily to

Features	MAE		MSE		MRE	
Size	1.058	(5)	2.022	(5)	14.27%	(5)
Shape	3.200	(15)	21.264	(15)	43.84%	(15)
Edges	1.583	(11)	5.475	(13)	19.71%	(10)
Keypoints	2.305	(14)	10.867	(14)	27.68%	(14)
Size, Shape	1.309	(9)	3.010	(8)	17.21%	(8)
Size, Edges	0.964	(3)	1.592	(3)	13.27%	(3)
Size, Keypoints	1.242	(7)	2.560	(6)	17.27%	(9)
Shape, Edges	1.295	(8)	3.187	(9)	16.97%	(7)
Shape, Keypoints	1.675	(12)	4.908	(12)	20.02%	(11)
Edges, Keypoints	1.697	(13)	4.803	(11)	22.18%	(13)
Size, Shape, Edges	0.906	(2)	1.584	(2)	12.32%	(2)
Size, Shape, Keypoints	1.511	(10)	3.964	(10)	20.14%	(12)
Size, Edges, Keypoints	0.966	(4)	1.610	(4)	13.81%	(4)
Shape, Edges, Keypoints	1.179	(6)	2.660	(7)	15.52%	(6)
Size, Shape, Edges, Keypoints	0.886	(1)	1.524	(1)	12.17%	(1)

Table 4.7: Comparison of features on the **QUT dataset, Camera 5**. The system is trained on the six other viewpoints.

Features	MAE	MSE	MRE
Size	1.228 (2)	3.105 (4)	13.45% (1)
Shape	3.417 (15)	20.715 (15)	43.17% (15)
Edges	2.008 (13)	7.184 (14)	25.53% (13)
Keypoints	1.784 (11)	7.050 (13)	22.44% (10)
Size, Shape	1.311 (4)	3.005 (2)	17.66% (5)
Size, Edges	1.633 (10)	4.487 (10)	19.68% (7)
Size, Keypoints	1.281 (3)	3.091 (3)	14.70% (2)
Shape, Edges	1.476 (7)	3.972 (8)	22.39% (9)
Shape, Keypoints	2.041 (14)	6.780 (12)	27.74% (14)
Edges, Keypoints	1.988 (12)	6.626 (11)	24.81% (12)
Size, Shape, Edges	1.542 (8)	3.925 (7)	19.88% (8)
Size, Shape, Keypoints	1.216 (1)	2.563 (1)	16.16% (3)
Size, Edges, Keypoints	1.433 (5)	3.504 (5)	17.52% (4)
Shape, Edges, Keypoints	1.597 (9)	4.485 (9)	23.93% (11)
Size, Shape, Edges, Keypoints	1.448 (6)	3.625 (6)	18.20% (6)

Table 4.8: Comparison of features on the **QUT dataset, Camera 8**. The system is trained on the six other viewpoints.

Features	Average Rank		
	MAE	MSE	MRE
Size	8.14	8.43	6.86
Shape	12.43	12.86	12.14
Edges	9.57	10.71	9.43
Keypoints	13.00	13.29	13.86
Size, Shape	8.57	8.14	8.29
Size, Edges	6.14	5.86	5.71
Size, Keypoints	5.71	5.14	5.57
Shape, Edges	8.00	8.14	7.57
Shape, Keypoints	10.71	10.43	11.00
Edges, Keypoints	10.57	10.71	11.14
Size, Shape, Edges	4.86	4.86	5.14
Size, Shape, Keypoints	5.57	5.43	6.43
Size, Edges, Keypoints	5.14	5.00	5.57
Shape, Edges, Keypoints	7.71	7.43	7.29
Size, Shape, Edges, Keypoints	3.86	3.57	4.00

Table 4.9: Average rank across **all seven datasets**, when features are ranked from 1 to 15 on each dataset. Values shown are not actual error rates, but rather an average ranking.

Dataset	MAE	MSE	MRE
PETS 2009, View 1	1.321	4.250	10.32%
PETS 2009, View 2	3.365	17.514	9.55%
PETS 2006, View 3	0.405	0.495	24.98%
PETS 2006, View 4	0.487	0.441	22.20%
QUT, Camera 3	1.574	3.506	14.03%
QUT, Camera 5	0.886	1.524	12.17%
QUT, Camera 8	1.448	3.625	18.20%
<i>Average</i>	1.351	4.479	15.92%

Table 4.10: Error rates for all datasets when **all features** are used. The *average* error rate across all datasets is also shown. Each dataset is weighted equally.

their small crowds of only 1-7 people, rather than inaccurate counting results; the mean absolute error rate is less than 0.5 for these datasets.

Figures 4.6 and 4.7 plot the estimate of the proposed algorithm (using all features) against the ground truth for Views 1 and 2 of the PETS 2009 dataset; Figures 4.8 and 4.9 plot Views 3 and 4 of the PETS 2006 dataset; and Figures 4.10-4.12 plot the results for the QUT datasets.

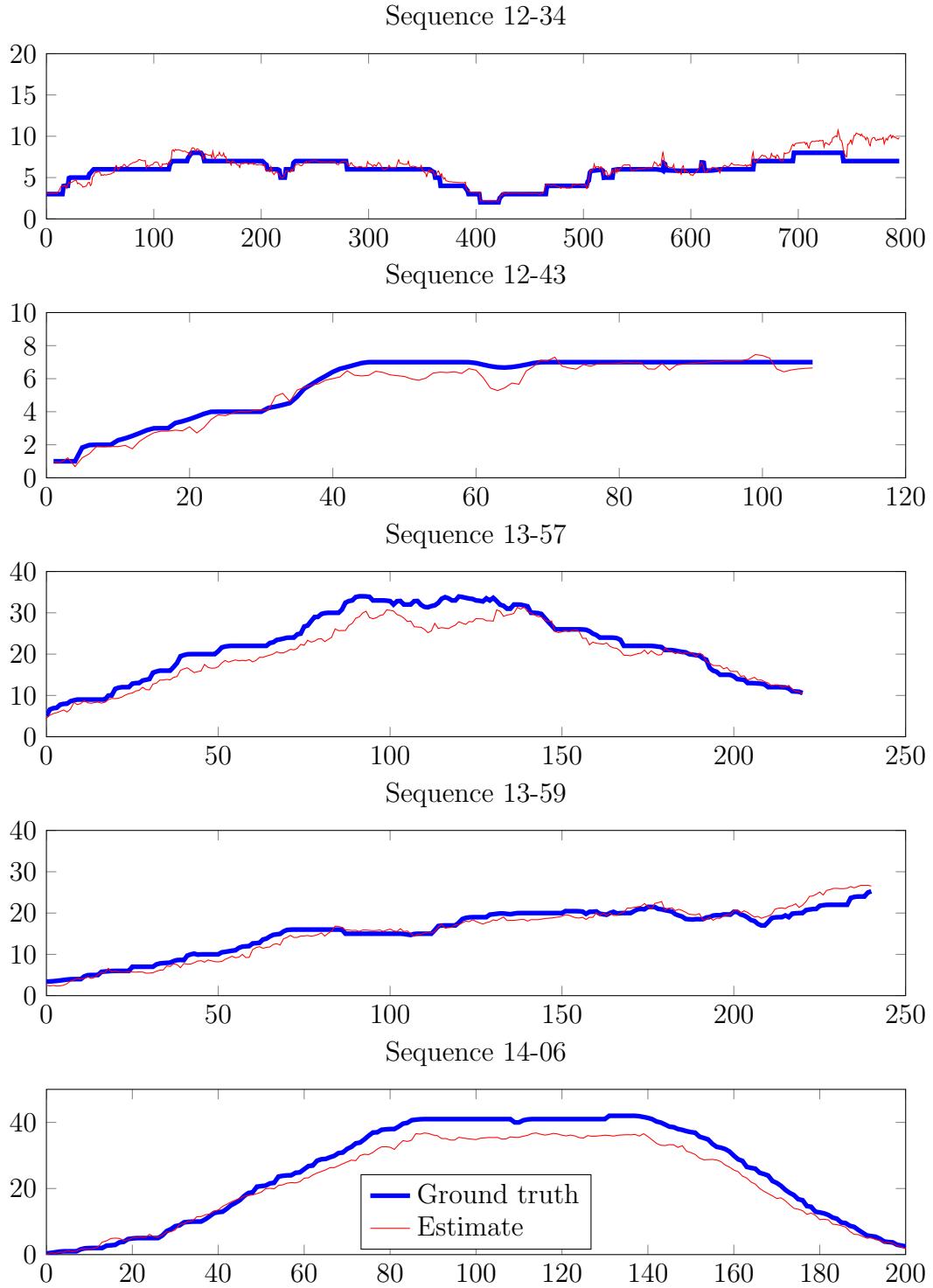


Figure 4.6: Scene invariant crowd counting results on **PETS 2009, View 1**.

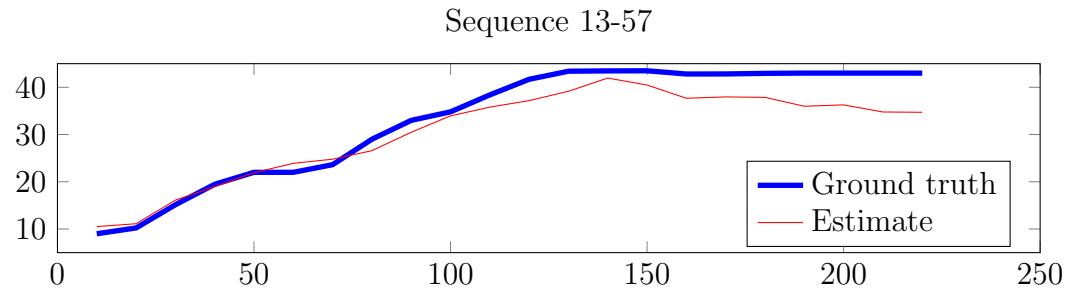


Figure 4.7: Scene invariant crowd counting results on **PETS 2009, View 2**.

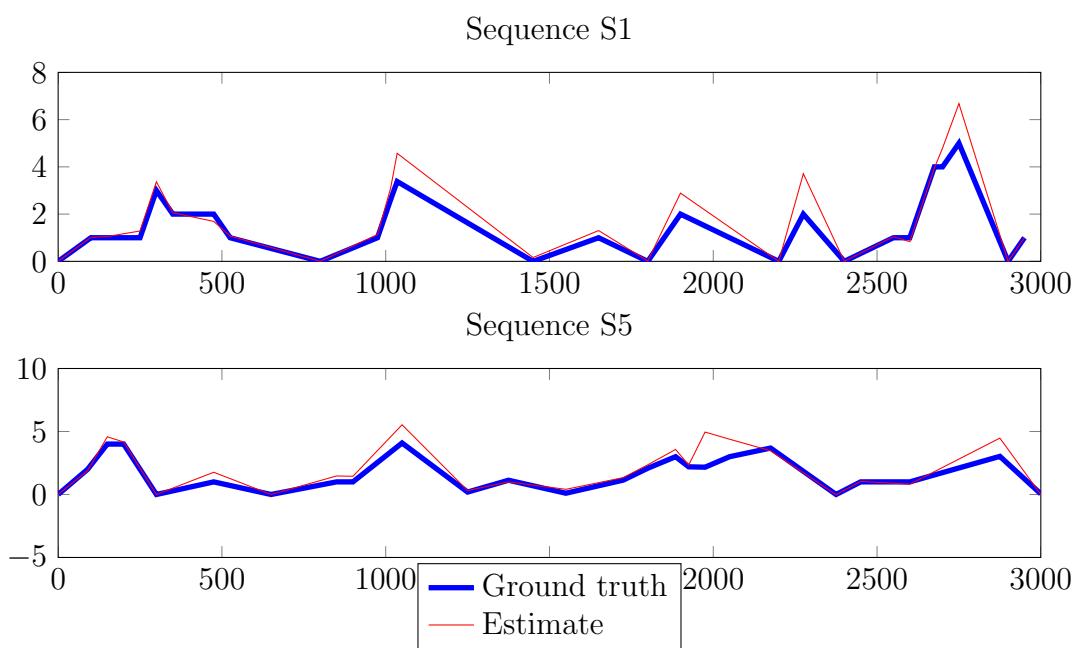


Figure 4.8: Scene invariant crowd counting results on **PETS 2006, View 3**.

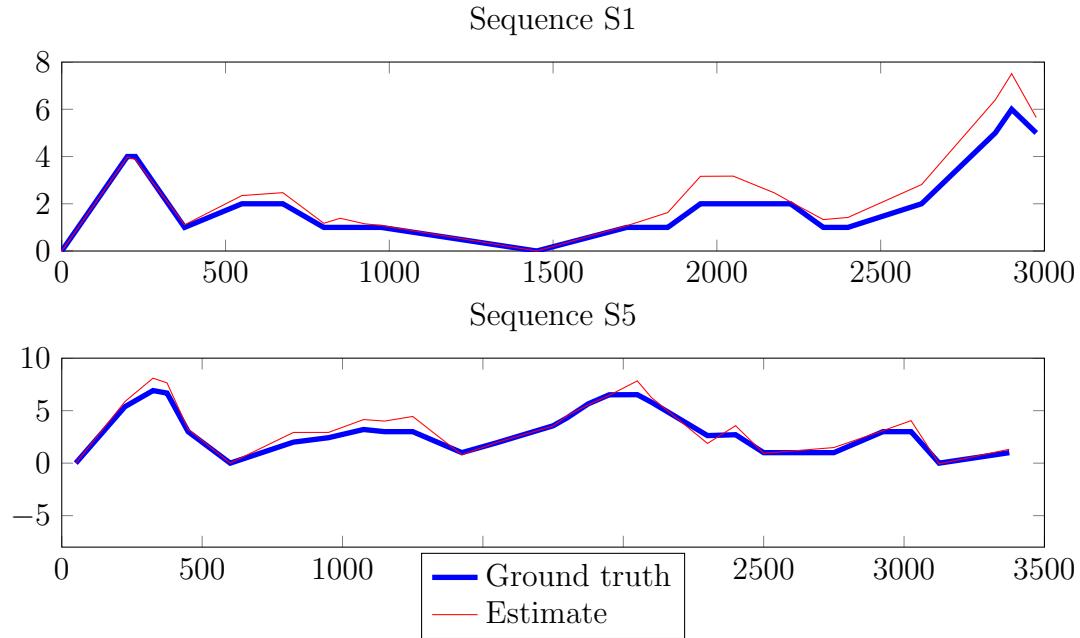


Figure 4.9: Scene invariant crowd counting results on **PETS 2006, View 4**.

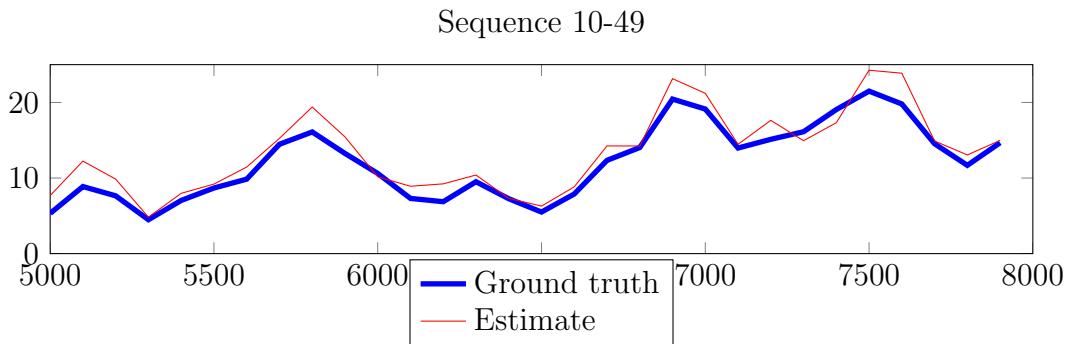


Figure 4.10: Scene invariant crowd counting results on **QUT, Camera 3**.

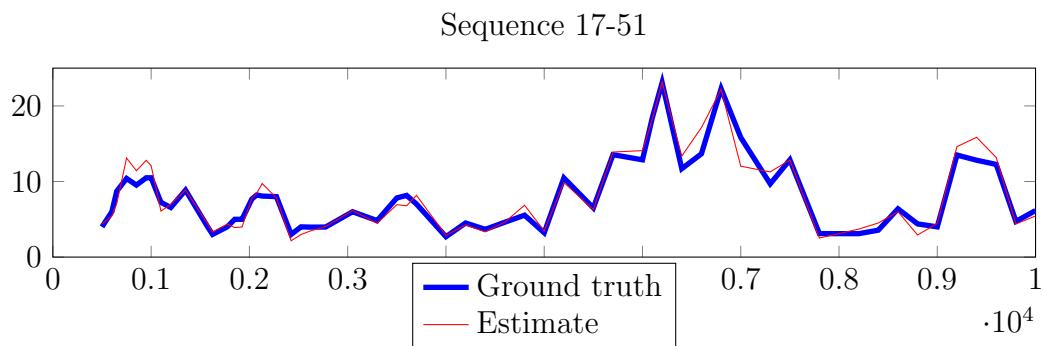


Figure 4.11: Scene invariant crowd counting results on **QUT, Camera 5**.

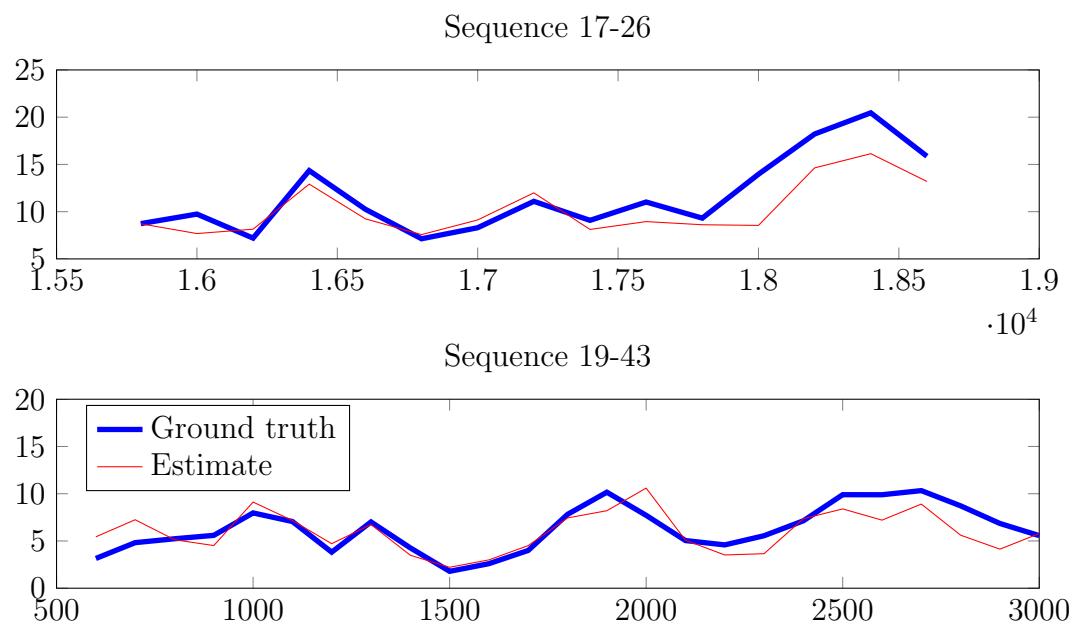


Figure 4.12: Scene invariant crowd counting results on **QUT, Camera 8**.

4.3.2.2 Regression Models

In this section, various regression models for scene invariant crowd counting are evaluated. These include: Gaussian process regression (GPR), linear regression (Linear), K -Nearest Neighbours (KNN) and neural networks (NN). Twelve different configurations are considered as described in Section 3.3.3.2.

The cross validation procedure of Section 4.3.2.1 is repeated using each of these regression models. For each dataset, these models are ranked from 1 to 12 for each error metric, and the *average ranking* is calculated across all seven datasets. Appendix B.1 presents the error rates and corresponding rankings for the regression models across *each* of the seven datasets.

Table 4.11 presents the average ranking across all of the seven datasets: average rank is tabulated for each regression model. Gaussian process regression consistently ranks highest, with an average rank of 2.43 (out of 12) in terms of MAE. This is followed by linear regression (average rank 4.57 in terms of MAE) and K -nearest neighbours.

These results are consistent with those presented in Section 3.3.3.2 of the previous chapter. It confirms that Gaussian process regression continues to outperform other regression models for scene invariant crowd counting. This provides strong support for the use of GPR in crowd monitoring applications.

4.4 Summary

This chapter proposed a novel scene-invariant crowd counting algorithm built upon local features. Camera calibration is incorporated into the system to scale features between viewpoints. Scene invariance was demonstrated by training

Regression Model	MAE	MSE	MRE
GPR	2.43	2.71	2.43
Linear	4.57	4.43	4.71
KNN (K=1)	6.14	7.00	6.14
KNN (K=2)	5.43	5.43	5.00
KNN (K=4)	4.57	5.00	4.57
KNN (K=8)	5.29	4.86	5.14
KNN (K=16)	5.14	5.00	5.43
KNN (K=32)	4.71	4.86	4.57
NN (4)	10.29	9.86	10.29
NN (8)	9.71	9.71	9.86
NN (16)	9.14	8.71	9.43
NN (32)	10.57	10.43	10.43

Table 4.11: Average rank across **all datasets**, when regression models are ranked from 1 to 12 on each dataset. Values shown are not actual error rates, but rather an average ranking.

the system on multiple cameras and then testing it on a new, unseen viewpoint. Accurate crowd counting results were obtained for the seven calibrated sequences, including a new QUT dataset designed to help evaluate the performance of crowd counting systems in difficult real-world conditions.

The following contributions were made in this chapter:

- A new method of calculating the density map to normalise features *across* viewpoints (not just within). The system uses camera calibration and a real world reference object, namely a human sized 3D cylinder model, to determine this density map.
- Three new benchmark datasets for crowd counting (QUT), with ground truth annotations and camera calibration parameters. These sequences feature challenging reflections, shadows and lighting fluctuations.
- A comprehensive evaluation on seven calibrated datasets, demonstrating scene invariance of the proposed algorithm.

The following conclusions were reached as a result of this evaluation:

- The proposed algorithm can successfully scale features across viewpoints and achieve accurate scene invariant crowd counting results.
- The use of multiple image features (size, shape, edges, keypoints) outperforms the use of fewer features across a wide range of datasets.
- The use of Gaussian process regression consistently outperforms the use of linear regression, neural networks and K -nearest neighbours.

Once trained, the proposed system does not require any additional training when deployed for crowd counting on a new camera. This brings the computer vision

field one step closer toward a ‘plug-and-play’ system which is pre-trained on a large bank of data from a variety of cameras. This technology has many potential applications, including automatic gathering of business intelligence, crowd safety monitoring and abnormality detection.

Chapter 5

Multi Camera Crowd Counting

5.1 Introduction

Crowd counting algorithms are typically designed to operate within a single camera viewpoint. This means that a crowd counting system is designed to estimate the number of people within an *image*, rather than a facility or a designated space in the real world.

Camera networks span multiple viewpoints within a facility, including some regions of overlap for redundancy. Since some individuals will be detected across multiple cameras, it is necessary to compensate for this overlap to avoid overestimation of the total number of people. The algorithm described in Chapters 3 and 4 is extended in this chapter to count crowds across multiple cameras. This is done by making use of camera calibration parameters to compensate for regions of overlap.

The proposed algorithm is tested on two multi-camera datasets, comprising five

cameras totals: Views 1, 2 and 3 from the PETS 2009 database [149]; and Views 3 and 4 from the PETS 2006 database [148]. These datasets feature crowds of size 1 to 43 people in various lighting conditions and differing camera angles. The system is demonstrated to be capable of supporting multiple cameras, in addition to being scene invariant, while achieving accurate crowd counting results.

The remainder of this chapter is structured as follows: Section 5.2 extends the algorithm of Chapter 4 to operate across multiple cameras, Section 5.3 presents the datasets and experimental results and Section 5.4 presents conclusions of this research.

5.2 Multi Camera Crowd Counting

The algorithm proposed in Chapter 4 is scene invariant, and therefore lends itself naturally to crowd counting across multiple cameras in a single environment. In this scenario, the same area is monitored using two or more cameras, with some potential overlap between the views. It is this overlap which presents a challenge in reconciling the crowd counts across all viewpoints.

A naive approach to multi camera crowd counting would be to take the sum of the crowd counts from each viewpoint. However, some pedestrians will appear in two or more cameras and will therefore be counted multiple times. One approach to deal with this scenario is to attempt to match groups between viewpoints and perhaps to identify individuals within groups. A complication with this approach is that groups segmented in one viewpoint will not necessarily correspond to those groups segmented in another. Figure 5.1 illustrates this point; the ‘groups’ that will be segmented from one angle are significantly different from another.

We seek to avoid the difficult problem of detecting individuals or matching blobs



Figure 5.1: There are not always direct correspondences between groups from different camera angles.

of various configurations. Two simple modifications to the existing algorithm of Chapter 4 are proposed which take into account the *overlap* between the viewpoints:

- 1. Density Map Modification:** This approach modifies the density map, S (Equation 4.8, p. 214) in regions of overlap so as to effectively compensate for the ‘double-up’ that occurs when a person is visible in more than one camera. This is done by reducing the density assigned to pixels in overlap regions. Counting is then performed as a subsequent step using this modified density map.
- 2. Pixel Density Assignment:** This approach leaves the density map unaltered, so that the system described in Chapter 4 operates unchanged. Instead, crowd densities are modified after counting has been performed, on a pixelwise basis. This makes use of the crowding density described in Section 3.4, and modifications are made to this crowding density in the regions of overlap.

Both approaches rely on the construction of an *overlap map*, which provides a

measure of how much of an object centred around a pixel is visible within other viewpoints.

The section is structured as follows: Section 5.2.1 describes the overlap map; Section 5.2.2 discusses the first approach, density map modification; Section 5.2.3 discusses the second approach, pixel density assignment; Section 5.2.4 discusses other baseline approaches; and Section 5.2.5 explains the procedure for ground truth annotation in a multi camera environment.

5.2.1 The Overlap Map

In a multi camera network there will be several video streams corresponding to the various cameras in the facility. Let the cameras monitoring a space be enumerated by v . This notation will be used as a superscript on the existing notation of Chapters 3 and 4. For example, $I^v(i, j)$ refers to pixel (i, j) in the image plane of camera v .

First we consider the pixel coordinates (i, j) in the image plane of camera v , around which a hypothetical object is centred. The cylinder model described in Section 4.2.2 is used for this purpose, with radius $r = 0.25\text{m}$ and height $h = 1.7\text{m}$ in real world measurements.

Using the camera calibration described in Section 4.2.1, let $(i, j)^{v \rightarrow u}$ denote this object's center in the image plane of camera u . That is, $(i, j)^{v \rightarrow u}$ denotes the conversion of an object's center in image plane v to its center in image plane u . This is done by first converting (i, j) from the image plane of camera v , into real world coordinates (x, y, z) where $z = \frac{h}{2}$, and then converting this position back into the image plane of camera u . This conversion and its implementation are described in Section 4.2.1 and in Tsai [189].

Let us also define $R_{i,j}^v$ as the region occupied by a cylinder model centred at (i,j) in the image plane of camera v . The magnitude $|R_{i,j}^v|$ represents the *area* of the cylinder model when projected into image plane v . We also use M^v to denote the region of interest mask for viewpoint v . The intersection of these regions in the image plane is $R_{i,j}^v \cap M^v$, and this represents the portion of the model which lies within the region of interest. The *fraction* of the model within the ROI is therefore:

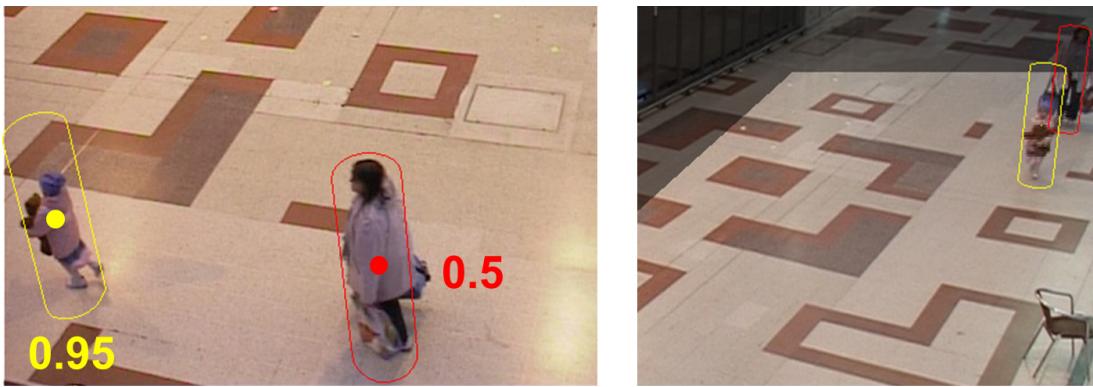
$$\frac{|R_{i,j}^v \cap M^v|}{|R_{i,j}^v|} \quad (5.1)$$

The cylinder model $R_{i,j}^v$ can be projected into another viewpoint, u , and this is denoted $R_{(i,j)^v \rightarrow u}^u$. The *overlap map* provides a measure for how much of this cylinder, $R_{i,j}^v$, is projected into other views. The overlap map for viewpoint v is therefore:

$$O_{i,j}^v = \sum_{\forall u \neq v} \frac{|R_{(i,j)^v \rightarrow u}^u \cap M^u|}{|R_{(i,j)^v \rightarrow u}^u|} \quad (5.2)$$

This is the value of the overlap map at pixel (i,j) in view v . The summation is taken across all *other* viewpoints, enumerated by u (where $u \neq v$), and the summand is the fraction of the cylinder model (centred around pixel (i,j) in view v) projected into the other viewpoints, u .

For example, if an object centred at (i,j) in viewpoint v is not projected into any other viewpoints, there is no overlap and hence $O_{i,j}^v = 0$. However, if 50% of the object is projected into another viewpoint, then $O_{i,j}^v = 0.5$. Figure 5.2(a) illustrates this. The person on the left is represented by a yellow cylinder model, 95% of which is projected into the region of interest of another viewpoint. Therefore



(a) Overlap example in the PETS 2006 dataset.



(b) Overlap example in the PETS 2009 dataset.

Figure 5.2: Examples of the overlap map at two points in the PETS 2006 and PETS 2009 datasets.

the value of the overlap map is 0.95 at this cylinder's center (i.e. the large yellow dot).

In some cases there may be more than two viewpoints. Equation 5.2 is taken as a summation across all other viewpoints, so that the total number of projections of the cylinder model into other viewpoints is included in the calculation. Figure 5.2(b) illustrates this. A red cylinder model representing a person in View 1 is

projected 100% into view 2 and 50% into view 3; therefore the value of the overlap map in View 1 is 1.5 at this person’s centroid.

The overlap map provides a measure of how much of an object centred at a particular pixel is visible in the other viewpoints. This enables us to compensate for over-estimation of the crowd in regions of overlap, as described in Sections 5.2.2 and 5.2.3.

5.2.2 Density Map Modification

Given a fully trained crowd counting system, as described in Chapters 3 and 4, we seek to perform counting across multiple viewpoints. The method proposed in this section is called density map modification because it modifies the density map, S (Equation 4.8, p. 214) in regions of overlap.

The density map is used to modify features to compensate for the effects of perspective, and to achieve scene invariance (Section 4.2.3). The value of the density map $S(i, j)$ is used to weight features extracted from the pixel (i, j) .

The following modification is proposed to reduce the value of the density map in regions of significant overlap. Let Δ^v to denote the *modified* version of the original density map, S^v , as follows:

$$\Delta^v(i, j) = \frac{S^v(i, j)}{1 + O_{i,j}^v} \quad (5.3)$$

The denominator contains two terms, $O_{i,j}^v$, which represents the projection of an object into other viewpoints, and ‘1’, which represents the object’s presence in the current viewpoint, v .

If an object centred at (i, j) in viewpoint v is 100% visible in another viewpoint u , then the value of the overlap map will be $O_{i,j}^v = 1$, so that the denominator of Equation 5.3 is exactly 2, resulting in a halving of the original density map's value. This compensates for the 'double-up' which would otherwise have occurred by counting the object twice. A smaller compensation will be applied when an object is only partially visible in another viewpoint, and no compensation is applied when the overlap is zero.

The density map modification described in Equation 5.3 is only applied *after* the system has been fully trained on individual viewpoints, but prior to the system being deployed across a multi camera environment. Once the density map has been modified, the crowd estimate for viewpoint v is denoted $\hat{\mu}^v$ so that the holistic estimate across all viewpoints can be summed directly:

$$\mu = \sum_v \hat{\mu}^v \quad (5.4)$$

This is the global count for the number of pedestrians in the scene. The value of $\hat{\mu}^v$ no longer provides a meaningful representation of the number of people in viewpoint v due to the modification of the density map.

5.2.3 Pixel Density Assignment

This method does not alter the density map, and instead allows the system to operate as intended, performing overlap compensation as a subsequent step. Crowding density, introduced in Section 3.4, is utilised to modify the crowd count in a pixelwise manner.

The 'crowding density' at a pixel is calculated as follows. Let μ_n^v denote the crowd

estimate for blob n in viewpoint v . The set of pixels belonging to this blob is denoted B_n^v ; and the density map for the viewpoint in question is denoted S^v . Finally, the ‘crowding density’ at pixel (i, j) is calculated by:

$$d^v(i, j) = F^v(i, j) \times \frac{S^v(i, j)}{\sum_{(i', j') \in B_{C^v(i, j)}^v} S^v(i', j')} \times \mu_{C^v(i, j)}^v \quad (5.5)$$

where F^v denotes the foreground detection result (with 0 and 1 representing the background and foreground pixels, respectively); and $C^v(i, j)$ denotes the identity of the blob to which pixel (i, j) belongs. In this way, the crowd count for blob $C^v(i, j)$ is split between its constituent pixels, weighted by the density map S^v to assign higher crowd density to more distant points in the scene to account for perspective.

Given this pixelwise crowd density, the overlap map is subsequently used to modify the crowd density in order to compensate for overlap between cameras and to avoid counting people multiple times. Let us use $\delta^v(i, j)$ to denote the *modified* version of the original crowding density, $d^v(i, j)$, as follows:

$$\delta^v(i, j) = \frac{d^v(i, j)}{1 + O_{i,j}^v} \quad (5.6)$$

The denominator in Equation 5.6 serves the same purpose as in Equation 5.3. The holistic count across all viewpoints is therefore the summation of all of the modified crowding densities across all viewpoints:

$$\mu = \sum_v \sum_{(i,j)} \delta^v(i, j) \quad (5.7)$$

This is the global count for the number of pedestrians in the scene.

5.2.4 Other Approaches

This section describes some additional approaches for crowd counting across multiple viewpoints, against which the proposed methods in Section 5.2.2 and 5.2.3 may be compared.

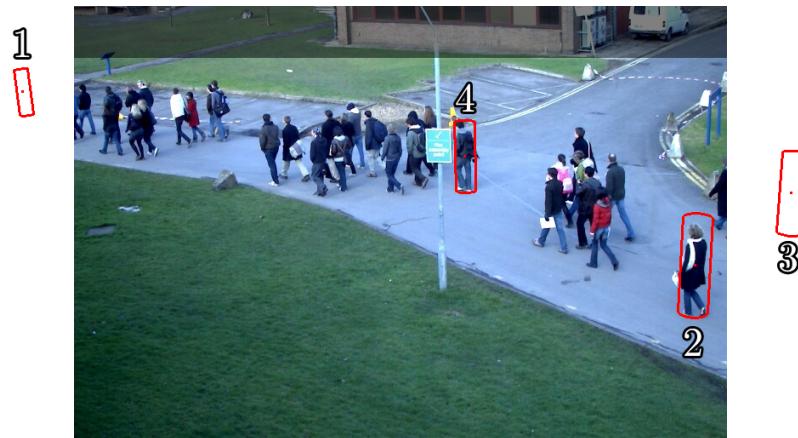
The first approach is referred to as the ‘naive’ method, in which the crowd counting algorithm of Chapter 4 operates unchanged, and the total crowd count is defined as the sum of the crowd counts μ^v from each individual camera:

$$\mu = \sum_v \mu^v \quad (5.8)$$

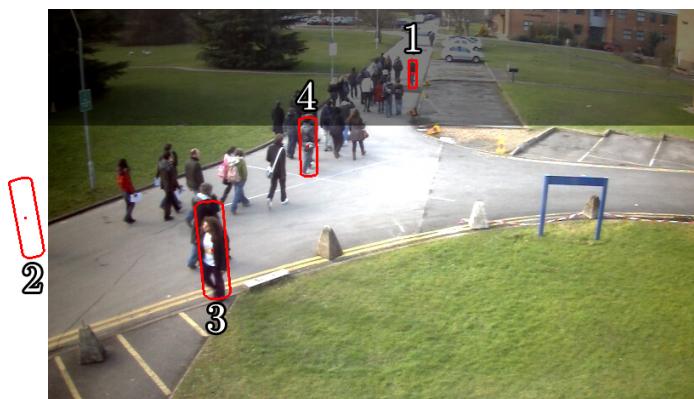
It is expected that this approach will overestimate the true crowd size due to camera overlap. An example of camera overlap in a three-camera setup is shown in Figure 5.3. The labelled annotations of four pedestrians indicate the regions which are unique to each viewpoint, as well as an overlap region.

The second approach is referred to as the ‘cutoff’ method, in which the ROI for each viewpoint is intentionally cut off to avoid any overlap between the viewpoints. This is based on a hypothetical straight line along the ground plane which is used to separate the views, as shown in Figure 5.4. A three-camera setup is also depicted in Figure 5.5.

Although the cutoff method is straightforward for two cameras, it may not be as easy to segment an arbitrary number of viewpoints from multiple angles using this technique.



(a) PETS 2009, View 1



(b) PETS 2009, View 2



(c) PETS 2009, View 3

Figure 5.3: The PETS 2009 database with ROIs highlighted and a sample of labelled pedestrian annotations. Pedestrians 1-3 indicate unique coverage areas, while pedestrian 4 indicates an overlap region.



(a) PETS 2009, View 1



(b) PETS 2009, View 2

Figure 5.4: ROI cutoff for a two-camera setup.



(a) PETS 2009, View 1



(b) PETS 2009, View 3

Figure 5.5: ROI cutoff for a three-camera setup. Views 1 and 3 shown; View 2 remains unchanged from Figure 5.4(b).

5.2.5 Ground Truth Annotation

In a single camera environment, the definition of ground truth is relatively straightforward: the number of pedestrians within the region of interest is taken as the ground truth, with fractional counts being assigned to pedestrians who are partially within the ROI (Section 3.2.4, p. 135). In the case of multiple cameras, a pedestrian may be partially visible in more than one camera, complicating the definition of ground truth.

This section defines holistic ground truth in a multi camera environment to be the *maximum* fraction of person to be visible within any camera in the environment.

Each person, enumerated by p , is annotated with a real world coordinate, (x, y, z) with $z = \frac{h}{2}$, located at approximately the center of their body. A cylindrical pedestrian template is projected around this point into each of the camera viewpoints, occupying a region in image plane v which we denote R_p^v . (The notation R_p^v in this section is used to represent the template of person p projected into image plane v , whereas the use of $R_{i,j}^v$ in the previous section was used to represent a template centred around pixel (i, j) in image plane v .)

The boundary of the pedestrian template, R_p^v , roughly covers the person in each viewpoint (Figures 5.2 and 5.3). We calculate the ‘quantity’ of person p within each region of interest mask M^v :

$$Q_p^v = \frac{|M^v \cap R_p^v|}{|R_p^v|} \quad (5.9)$$

This is similar to Equation 3.38 (p. 141), except that the cylinder model is projected into each viewpoint v . We can then determine the maximum quantity

of person p within the *scene* (across all viewpoints):

$$Q_p = \max_v Q_p^v \quad (5.10)$$

Thus the holistic ground truth for the scene is taken to be:

$$Q_s = \sum_p Q_p \quad (5.11)$$

as in Equation 3.49 (p. 144). This definition of ground truth is used for evaluating the performance of the proposed algorithm in Section 5.3.

5.3 Experimental Results

This section presents experimental results of the proposed crowd counting algorithm on multi camera networks. Section 5.3.1 evaluates the algorithm on two-camera environments, using both the PETS 2006 and PETS 2009 datasets. Section 5.3.2 combines scene invariance and multi camera crowd counting, and evaluates these algorithms on a three-camera environment using the PETS 2009 database.

5.3.1 Crowd Counting Across Two Cameras

In this section the multi camera crowd counting algorithm is evaluated on two-camera environments using the PETS 2006 and PETS 2009 datasets. The system is trained using the procedure described in Section 3.2.4 (p. 135), and then testing is performed using the algorithms presented in Section 5.2. The performance of

these algorithms is evaluated against the definition of ground truth described in Section 5.2.5.

The density map modification of Section 5.2.2 is referred to as the ‘Map’ method, and the pixel density assignment of Section 5.2.3 is referred to as the ‘Pixel’ method.

5.3.1.1 PETS 2006 Results

The PETS 2006 database provides camera calibration for each of its viewpoints, and Views 3 and 4 were selected because they provide the best coverage of the scene with some overlap. Two sequences from these cameras, S1 and S5, were chosen because they contain the largest crowds. Frames were annotated at regular intervals (every 100 frames) with the location of pedestrians in the scene.

A two-fold cross validation procedure was used to assess performance on this database. The system was first trained on sequence S5 and tested on sequence S1, and then vice versa. Average results across all frames in both sequences are then reported. Results are displayed in Table 5.1 for all four algorithms.

The proposed algorithms operate with a mean absolute error of 0.446 and 0.388 for the map and pixel methods, respectively. The mean relative error does not exceed 20% for either algorithm. By comparison, the naive and cutoff methods operate with a mean absolute error of 0.678 and 0.722 respectively.

The results are also plotted in Figure 5.6. As expected, the naive method overestimates the crowd size when pedestrians pass through the overlap region. By contrast the cutoff methods tends to underestimate the crowd during this time. Both the map and pixel methods give suitable results.

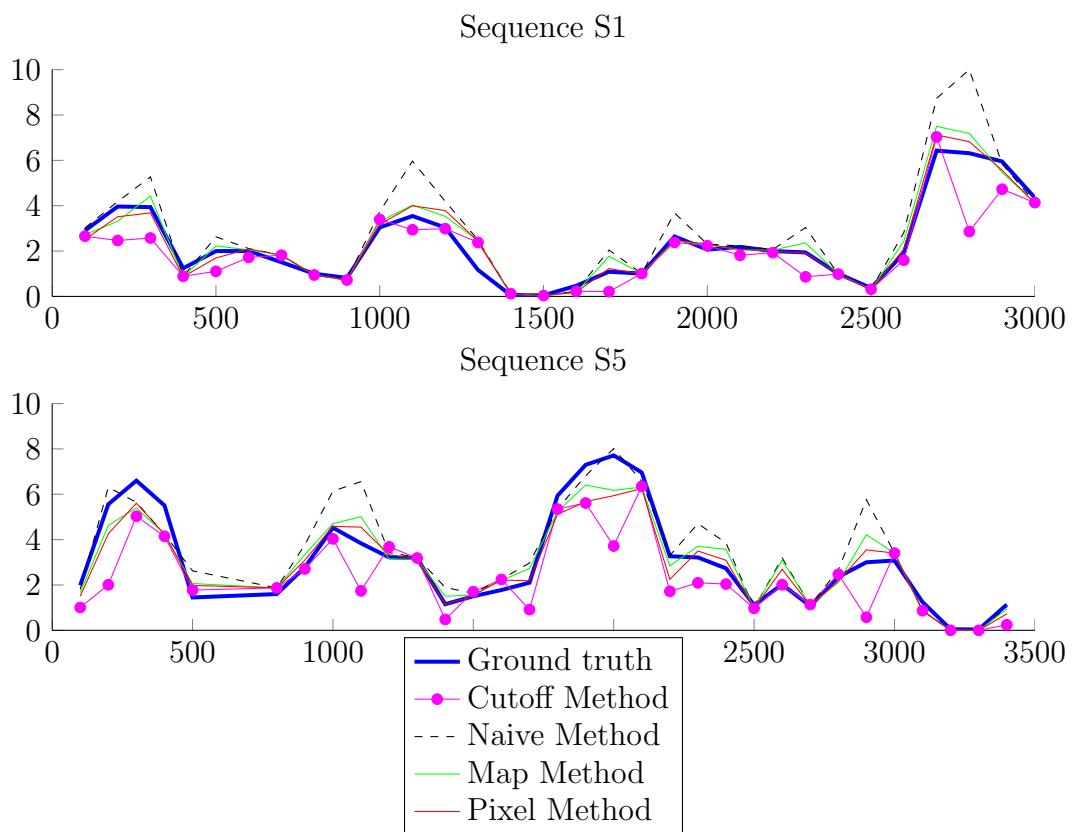


Figure 5.6: Results of multi camera crowd counting using **Views 3 and 4** of the PETS 2006 dataset.

		Map Method			Pixel Method			Naive Method			Cutoff Method		
Training	Testing	MAE	MSE	MRE	MAE	MSE	MRE	MAE	MSE	MRE	MAE	MSE	MRE
S5	S1	0.343	0.217	18.74%	0.269	0.145	14.34%	0.663	1.166	31.40%	0.545	0.780	26.23%
S1	S5	0.542	0.472	19.00%	0.500	0.472	16.03%	0.691	0.963	26.47%	0.888	1.739	28.82%
<i>All frames</i>		0.446	0.349	18.88%	0.388	0.314	15.21%	0.678	1.061	28.86%	0.722	1.275	27.57%

Table 5.1: Results of multi camera crowd counting using **Views 3 and 4** of the PETS 2006 dataset.

		Map Method			Pixel Method			Naive Method			Cutoff Method		
Training	Testing	MAE	MSE	MRE	MAE	MSE	MRE	MAE	MSE	MRE	MAE	MSE	MRE
13-59	13-57	0.879	1.715	3.65%	2.313	8.438	8.60%	7.378	66.919	31.51%	2.262	8.959	8.81%
13-57	13-59	1.443	3.526	8.27%	1.132	1.940	7.53%	5.953	50.568	32.00%	1.054	1.573	6.94%
<i>All frames</i>		1.173	2.660	6.06%	1.697	5.048	8.04%	6.635	58.388	31.77%	1.631	5.105	7.84%

Table 5.2: Results of multi camera crowd counting using **Views 1 and 2** of the PETS 2009 dataset.

5.3.1.2 PETS 2009 Results

Camera calibration is provided for multiple viewpoints of the PETS 2009 database, and Views 1 and 2 were selected due to the wide coverage and overlap that they provide. Sequences 13-57 and 13-59 were selected for evaluation because these were specifically captured for the purpose of evaluating crowd counting algorithms. These datasets contain much larger crowds than the PETS 2006 database, and two frames from this database are shown in Figure 5.1. Frames were annotated at regular intervals (every 20 frames) with the locations of pedestrians in the scene for evaluation purposes.

Cross validation with two folds is performed using the 13-57 and 13-59 sequences, and the results of this analysis are shown in Table 5.2. The proposed algorithms achieve a mean absolute error of 1.173 and 1.697 for the map and pixel methods, respectively. By comparison the naive method reports a MAE of 6.635 due to severe overestimation of the crowd size due to overlap regions. The cutoff method achieves a MAE of 1.631 which is comparable to the pixel method. The best performance across all three error metrics is observed for the Map method.

The results are plotted in Figure 5.7. The naive method overestimates the crowd size severely, while good performance ($MRE < 10\%$) is observed for the other methods.

5.3.2 Scene Invariant Crowd Counting Across a Multi-Camera Network

In this section, scene invariance is combined with multi camera crowd counting, and the proposed algorithm is evaluated on a three-camera environment using

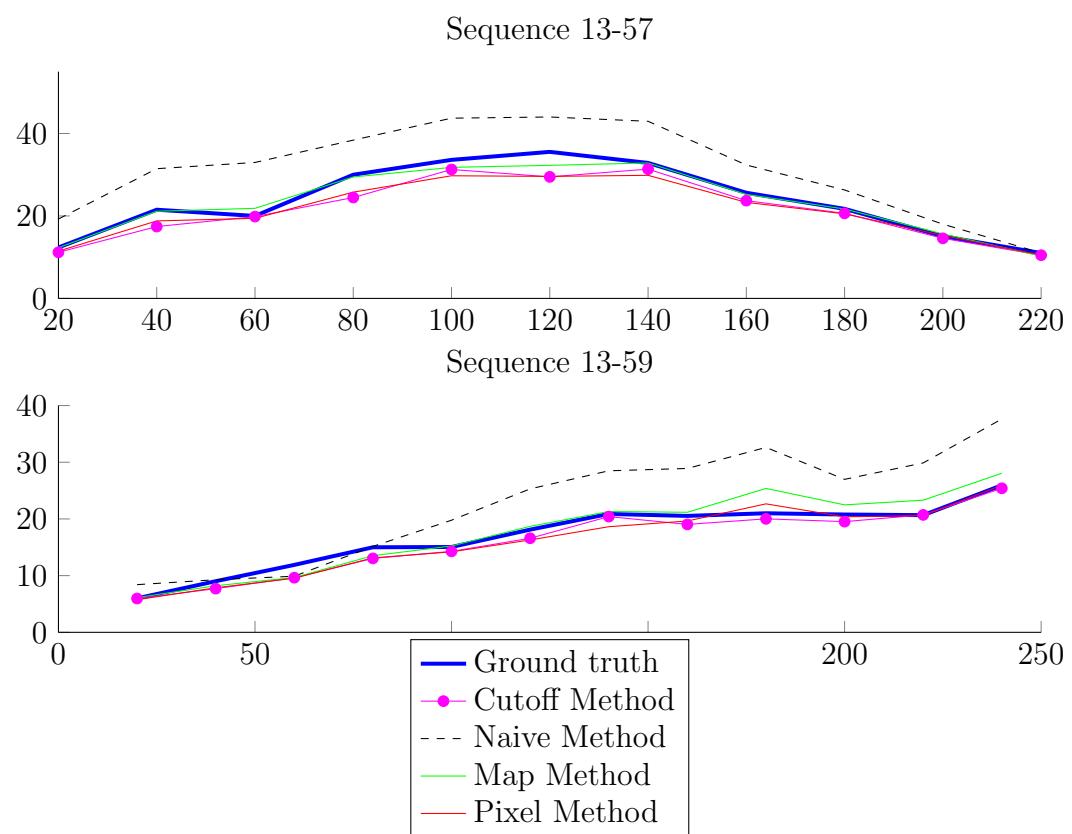


Figure 5.7: Results of multi camera crowd counting using **Views 1 and 2** of the PETS 2009 dataset.

the PETS 2009 database.

Training is performed using three QUT datasets (Cameras 3, 5 and 8) described in Section 4.3.1 (p. 216). Feature normalisation is performed as described in Section 4.2.3 to achieve scene invariance.

Testing is performed on Views 1, 2 and 3 of the PETS 2009 database, for sequences 13-57 and 13-59. The ROIs used for this evaluation are shown in Figure 5.3. A selection of pedestrians have been annotated to aid the reader in interpreting the relationship between the camera viewpoints. The ROIs used for the cutoff method are also shown in Figures 5.4(b) and 5.5.

Table 5.3 presents the results of this evaluation. The proposed Map and Pixel methods achieve mean absolute error rates of 1.756 and 2.665 respectively. Both approaches report a MRE < 10% indicating highly accurate performance. These approaches both outperform the cutoff method which achieved a MRE of 14.34%. This is due to imperfect separation provided by the ROI cutoff: humans are 3D objects and therefore it is difficult to achieve true separation using a hypothetical line on the ground plane. The compensation provided by the overlap map (Section 5.2.1) appears to provide superior crowd counting performance in a complicated three-camera environment such as this.

Results for this experiment are plotted in Figure 5.8. These results provide strong support for both the scene invariant and the multi camera crowd counting algorithms proposed in this thesis.

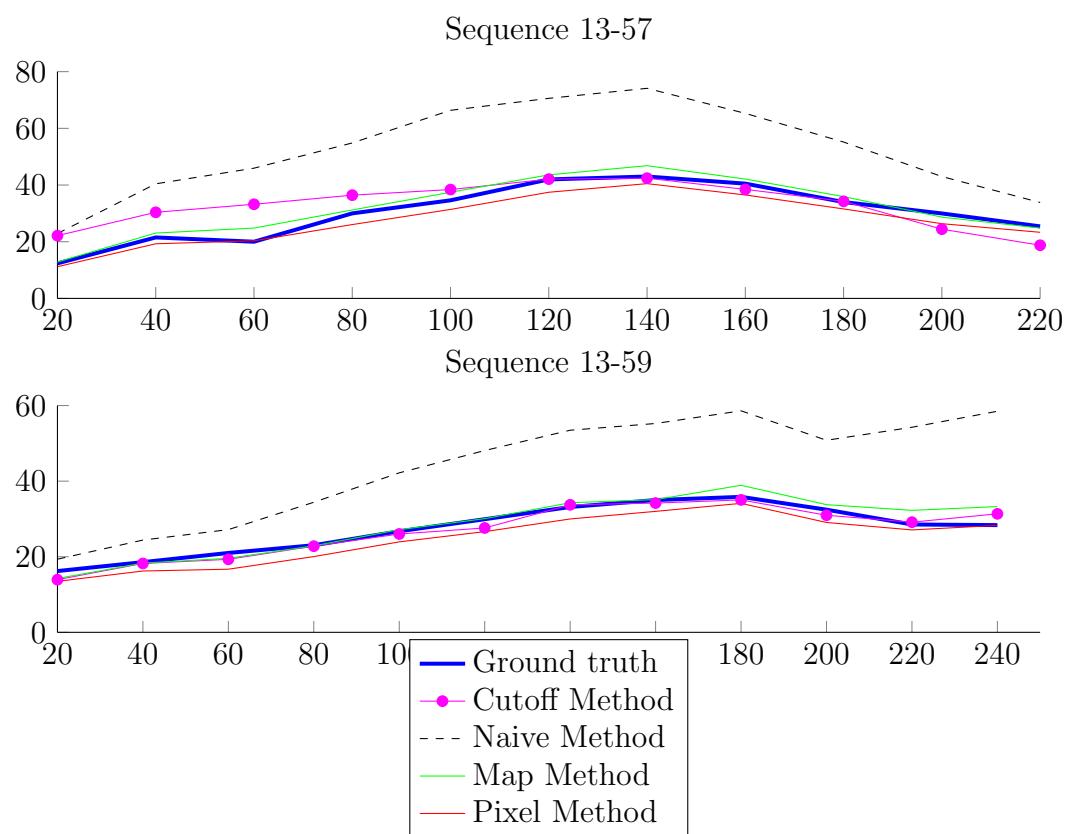


Figure 5.8: Results of multi camera crowd counting using **Views 1, 2 and 3** of the PETS 2009 dataset.

Training	Testing	Map Method			Pixel Method			Naive Method			Cutoff Method		
		MAE	MSE	MRE	MAE	MSE	MRE	MAE	MSE	MRE	MAE	MSE	MRE
QUT	13-57	1.977	5.533	6.95%	2.732	8.952	8.89%	21.739	532.523	74.37%	5.207	44.034	24.63%
QUT	13-59	1.554	4.717	5.81%	2.603	7.878	10.25%	16.477	335.754	56.98%	1.236	2.262	4.91%
<i>All frames</i>		1.756	5.107	6.36%	2.665	8.392	9.60%	18.993	429.861	65.30%	3.135	22.240	14.34%

Table 5.3: Results of multi camera crowd counting using **Views 1 2, and 3** of the PETSc 2009 dataset.

5.3.3 Conclusions

Both of the proposed approaches demonstrate the ability to count crowds in a multi camera environment. The map method ('Density Map Modification') alters the density map in regions of overlap, which modifies the values of the image features prior to regression. By contrast, the pixel method (Pixel Density Assignment) performs compensation on a pixelwise basis after regression has been performed.

These algorithms demonstrate accurate crowd counting results, with the pixel method performing slightly better on the PETS 2006 database, while the map method performs better on the PETS 2009 database. Both algorithms outperform the naive approach and provide similar or better performance than the ROI cutoff method (described in Section 5.2.4).

As noted in Section 5.2.2, the map method results in a modified crowd estimate for each viewpoint, $\hat{\mu}^v$, the summation of which over all viewpoints yields the total number of pedestrians in the scene (Equation 5.4). These estimates $\{\hat{\mu}^v\}$ do not provide an accurate indication of the number of people in each individual viewpoint, due to the modification of the density map prior to regression. If the number of people in a particular viewpoint is required, feature extraction and regression would need to be repeated using the unmodified density map S . The redundancy of this computation makes the map method less intuitive and attractive than the pixel method, which only modifies the crowding density after feature extraction and regression have been performed.

Both approaches require some additional computational overhead compared to the naive and cutoff methods. However, the accuracy is improved by doing so.

5.4 Summary

This chapter proposed a novel multi camera crowd counting algorithm built upon local features and scene invariant crowd counting. Camera calibration is incorporated into the system to compensate for people in regions of overlap and to scale features between viewpoints. Accurate crowd counting results were obtained using two datasets comprising five calibrated cameras.

The following contributions were made in this chapter:

- The construction of an overlap map which enables a system to quantify the amount of overlap in any given region of an image. This is based on a real world object of fixed dimensions, a cylinder model which is the approximate size of a human. This approach serves to estimate the overlap specifically for humans or objects of similar size.
- Two novel methods for compensating for camera overlap when counting crowds. Density Map Modification alters the density map in regions of overlap, while Pixel Density Assignment determines the crowding density at each pixel and then modifies it for multi camera environments. Both approaches make use of the overlap map.
- The ROI cutoff method is also described as an alternative method, which is computationally more efficient but slightly less accurate.
- Crowd counting experiments in multi camera environments demonstrate highly accurate crowd counting results on the PETS 2006 and PETS 2009 databases using the proposed algorithms.
- The combination of scene invariance and multi camera crowd counting demonstrates the efficacy of the algorithms proposed in this thesis. A mean

relative error of 6-8% was observed using a three-camera setup.

The proposed multi camera crowd counting algorithm has been demonstrated to be effective in counting crowds across multiple viewpoints, and can be deployed in large surveillance environments where a single camera is insufficient to observe the entire crowd.

Chapter 6

Queue Monitoring

6.1 Introduction

Estimating queue parameters is an important part of many business operations, including retail shops, public transport hubs and airports. When these queues are particularly large, as at airport check-in counters with multiple service stations, it is difficult for human operators to quantitatively assess queue parameters such as throughput rate, overall queue size, growth rate and wait time.

This chapter presents a queue monitoring algorithm which combines crowd counting and crowd flow monitoring in order to calculate the required parameters. There has been very little research in the computer vision field targeting the monitoring of queues. Although crowd counting and crowd flow monitoring have been actively researched in recent years, they have not previously been combined for the purpose of queue monitoring.

In developing this queue monitoring algorithm, a novel crowd flow monitoring

algorithm (virtual gate) has been developed, which monitors crowd flow automatically at a specified region within an image. Feature points corresponding to pedestrians are detected and accumulated as they pass through the virtual gate. During each window of time, the number of people who entered the gate is estimated. The queue monitoring system also makes use of the novel crowd counting algorithm discussed in Chapters 3-5 of this thesis.

The underlying algorithms used in this system can be applied to complex scenes and multi camera environments, resulting in a very flexible and powerful queue monitoring system. This chapter is structured as follows: Section 6.2 presents the proposed queue monitoring framework which is based on the combination of virtual gates and crowd counting; Section 6.3 describes the proposed virtual gate algorithm; Section 6.4 evaluates the queue monitoring framework on real world data from an international airport; and Section 6.5 presents the conclusions of this research.

6.2 Queue Monitoring Framework

The queue monitoring system developed for this project is comprised of two core components:

1. **Crowd Counting.** This module covers a holistic view of the scene (including support for multiple cameras) and enables the system to estimate the number of people waiting in a queue. This algorithm is described in Chapters 3-5.
2. **Virtual Gates.** Positioned at the entry and/or exit points of a queue, the purpose of a virtual gate is to detect the pedestrians as they pass across a line (or through a region of interest). This enables the system to estimate

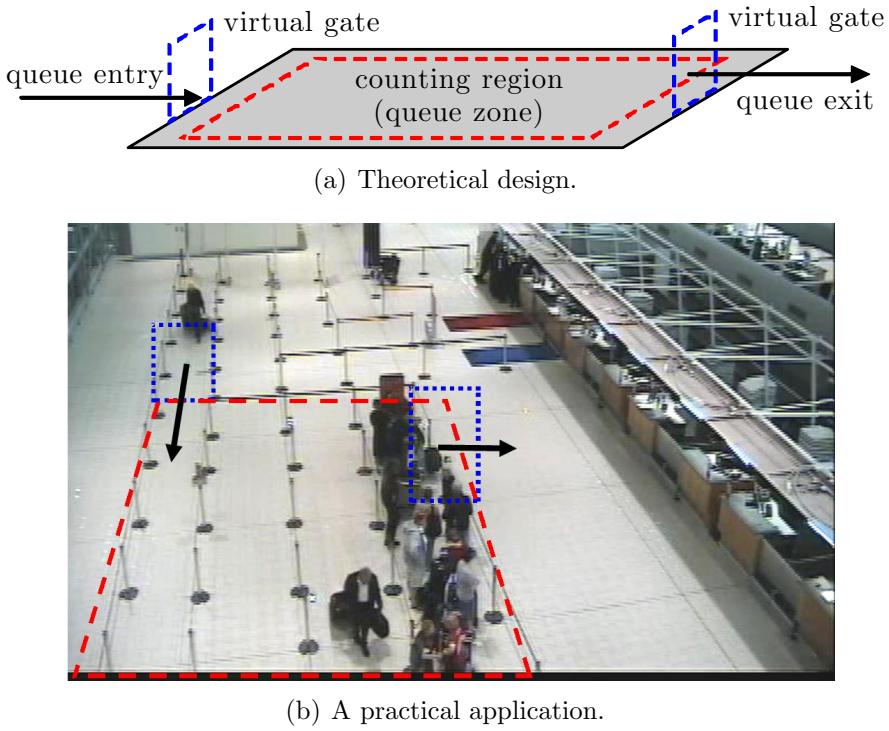


Figure 6.1: Design of the proposed queue monitoring system.

the number of pedestrians entering or exiting a queue, as well as the rates of entrances/exits over time. This algorithm is described in Section 6.3.

These components are illustrated in Figure 6.1. The counting region is placed over the queueing zone, with a virtual gate positioned at either end of the queue. The fusion of these modules is used to estimate queue parameters.

Let the size or length of a queue at time t be denoted Q_t , which represents the number of people waiting to be serviced. The rate of change of this queue length, q_t , is the *growth rate*:

$$q_t = \frac{\Delta Q_t}{\Delta t} = \frac{Q_t - Q_{(t-\Delta t)}}{\Delta t} \quad (6.1)$$

A typical value for Δt might be 3-5 minutes, as this will give a reasonable indication of how the queue size has changed in the short term, although any length of time can be used to estimate the growth rate depending on the target application.

The total number of arrivals at the end of a queue is denoted A_t , and this represents the cumulative count of pedestrians who have entered the queue since time $t = 0$. Its rate of change, a_t , is the *arrival rate*:

$$a_t = \frac{\Delta A_t}{\Delta t} = \frac{A_t - A_{(t-\Delta t)}}{\Delta t} \quad (6.2)$$

Similarly, the total number of people who have been serviced at the front of the queue is denoted S_t , and its rate of change, s_t , is the *service rate* (or throughput rate):

$$s_t = \frac{\Delta S_t}{\Delta t} = \frac{S_t - S_{(t-\Delta t)}}{\Delta t} \quad (6.3)$$

From an operational standpoint, the values of a_t and s_t are more interesting than A_t and S_t because they provide instantaneous data about the *current* state of the queue. By contrast, the measurements of A_t and S_t include the total number of people counted by the system since its inception (or an arbitrary reference point which we refer to as $t = 0$). Both Q_t and q_t provide instantaneous information about the crowd size and its growth rate respectively, and therefore contain operationally meaningful information.

Finally, the *wait time* is the average time expected for a person at the end of the queue to be serviced. The current service rate s_t can be used to estimate the wait time, T_w :

$$s_t = \frac{\Delta S_t}{\Delta t} \quad (6.4)$$

$$\approx \frac{Q_t}{T_w} \quad (6.5)$$

Thus, given the current estimate of queue length Q_t and service rate s_t , the wait time is expected to be:

$$T_w = \frac{Q_t}{s_t} \quad (6.6)$$

For example, a queue of length $Q_t = 50$ people serviced at a rate of $s_t = 5$ people per minute will take $T_w = \frac{50}{5} = 10$ minutes to process.

When all queue parameters (Q_t , A_t , S_t) are monitored, there is a redundancy of information because any parameter may be derived from the other two. This is because queue size is the difference between the total number of arrivals and the total number of people who have been serviced thus far:

$$Q_t = A_t - S_t \quad (6.7)$$

Equivalently, the growth rate is equal to the difference between arrival rate and service rate:

$$q_t = a_t - s_t \quad (6.8)$$

Therefore it is only strictly necessary to monitor two out of the following three parameters:

1. **Queue length**, Q_t (using crowd counting techniques)
2. **Number of arrivals**, A_t (using a virtual gate)
3. **Number of people serviced**, S_t (using a virtual gate)

This gives rise to three possible scenarios:

1. When **arrivals** are not monitored, they may be estimated from:

$$A_t = S_t + Q_t \quad (6.9)$$

This is the total number of people serviced plus those remaining to be serviced. Similarly, the arrival rate is:

$$a_t = s_t + q_t \quad (6.10)$$

2. When the number of people who have been **serviced** are not monitored, they may be estimated from:

$$S_t = A_t - Q_t \quad (6.11)$$

This is the total number of people who have arrived minus those still remaining in the queue to be serviced. Similarly, the service rate is:

$$s_t = a_t - q_t \quad (6.12)$$

3. When **queue size** is not monitored, the estimates obtained from Equations 6.7 and 6.8 may be used. However, the use of two virtual gates may prove problematic as historical errors in A_t , S_t are accumulated and this may affect the value of Q_t . This may be described as a gradual ‘drift’ over time

due to small inaccuracies in the virtual gates (or due to people who enter or exit the queue from unusual locations).

It is recommended that a crowd counting algorithm is used to directly count the crowd in order to avoid this drift. Alternatively, if the crowd size is derived from A_t , S_t (Equation 6.7) then it should be reset to 0 during periods of inactivity to mitigate the effects of drift.

The individual modules that comprise the queue monitoring algorithm operate independently of one another. The crowd counting algorithm is described in Chapters 3-5 and the virtual gate algorithm is presented in Section 6.3.

6.3 The Virtual Gate

The virtual gate is designed to count pedestrians passing through a region of interest or ‘gate’ over time. This may take place at the entrance or exit to a queue, for example. By comparison, ‘crowd counting’ is concerned with the number of pedestrians *within* a ROI at a single point in time. The similarities and differences between the two algorithms are summarised in Table 6.1.

The crowd counting algorithm employs background modelling to detect both moving and stationary pedestrians, while the virtual gate uses optical flow to measure *moving* pedestrians only. Crowd counting is performed in a single frame, while the virtual gate measures crowd flow across an entire video sequence. Therefore, while crowd counting is evaluated over multiple frames, the virtual gate algorithm is evaluated over multiple sequences. For each sequence, the number of pedestrians entering the gate is annotated, and the system’s estimate is compared to this ground truth.

Crowd Counting	Virtual Gate
Counts the number of people located within a wide ROI.	Counts the number of people passing through a small ROI.
Measured at a single point in time.	Measured over a longer period of time.
Counts both moving and stationary pedestrians.	Counts moving pedestrians only.
Detected using adaptive background modelling (motion detection).	Detected using optical flow.
Error is measured by comparing estimate to ground truth in a single frame.	Error is measured by comparing estimate to ground truth across a sequence.
Evaluation is performed across multiple frames.	Evaluation is performed across multiple sequences.
Local features are extracted by dividing an image into foreground segments.	Local features are extracted by dividing a sequence into sub-sequences, or <i>windows</i> .
Holistic features are extracted from the entire image.	Holistic features are extracted from the entire sequence.

Table 6.1: The comparison between crowd counting and virtual gate algorithms presented in this thesis.

As discussed in Section 2.4, existing crowd flow monitoring algorithms have typically fallen into two categories:

1. **Detection and tracking.** These approaches utilise object or feature tracking [7, 39, 173, 198]. In crowded scenes where the camera is sufficiently close to the scene, face detection has been used to track incoming pedestrians [47, 205] as well as head and shoulder contours [175].
2. **Regression of optical flow.** These approaches adopt a statistical approach more suitable for crowded scenes. They typically calculate the optical flow fields within a gate and accumulate the motion vectors over time [20, 27, 98]. A regression function is learned to estimate the number of people entering based on the optical flow.

In a crowded environment such as a queue, human detection, face detection and object tracking are challenging due to the size of the crowd and the distance of the camera from the scene (Figure 6.1(b)). In many cases these algorithms are also computationally expensive. Regression based approaches are therefore most applicable to crowded environments.

This chapter draws upon both methods and proposes a combined approach: *feature points* are detected in an image (such as corners and edges) and their motion is accumulated over time. This accumulated motion serves as input to a regression model. The proposed algorithm is based on the intuition that the number of people passing through a gate is related to the number of feature points crossing a counting line, although the relationship may not necessarily be linear. The proposed system establishes a region of interest (ROI) around the counting line (Figure 6.2) and calculates the summation of optical flow of these feature points. Therefore a combination of feature detection and regression is used to obtain the estimated count.

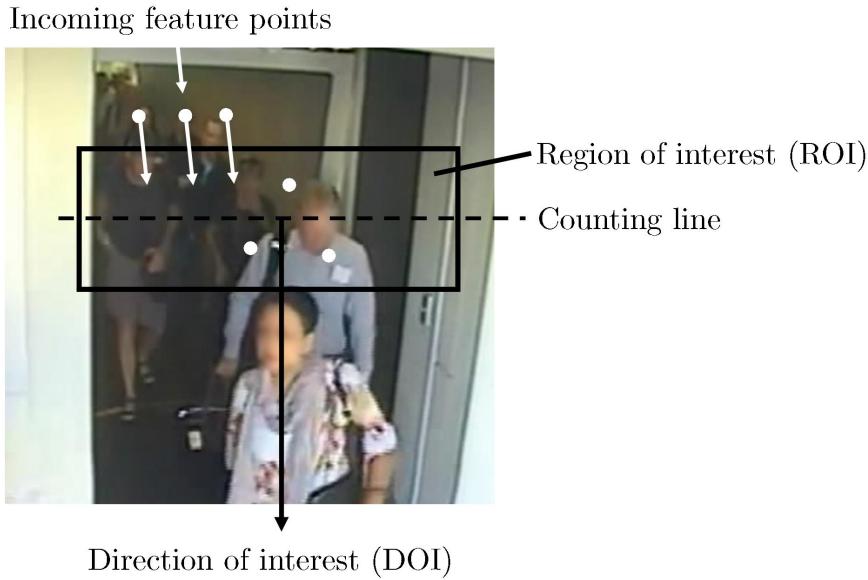


Figure 6.2: The components of a virtual gate.

This section is structured as follows: Section 6.3.1 presents an overview of the proposed virtual gate algorithm; Section 6.3.2 introduces the use of optical flow histograms to separate noise from meaningful flow information; Section 6.3.3 describes the feature selection process used in this evaluation; and Section 6.3.4 describes the ground truth annotation method used to train the system.

6.3.1 Virtual Gate Overview

The virtual gate is depicted in Figure 6.2. It is comprised of a counting line surrounded by a region of interest (ROI), and a direction of interest (DOI) in which pedestrians move. The set of pixels belonging to the ROI is denoted R , and the unit vector pointing in the DOI is denoted d .

The proposed algorithm accumulates optical flow *in the direction of interest* at a discrete set of detected feature points. Each video sequence is divided into a set of sub-sequences, or *windows*, in which the optical flow is accumulated (Figure

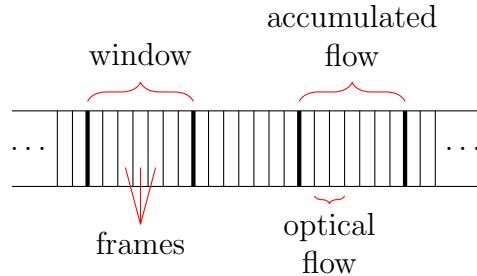


Figure 6.3: Visual representation of frames and windows. Optical flow based features are computed between frame pairs and accumulated across each window.

6.3). Regression is then applied to each window independently to estimate the number of people passing through the virtual gate. This approach is analogous to the use of local features within a crowd counting algorithm (Chapter 3), whereby an image is divided into a set of segments and the number of people within each segment is estimated.

The features within each window are calculated as follows. The optical flow field at time t is denoted \mathbf{v}_t , and the optical flow at a pixel \mathbf{p} is denoted $\mathbf{v}_t(\mathbf{p})$. The component of this flow which points in the direction of interest \mathbf{d} is referred to as the *aligned* optical flow, and is computed using the dot product:

$$\hat{\mathbf{v}}_t(\mathbf{p}) = \mathbf{v}_t(\mathbf{p}) \cdot \mathbf{d} \quad (6.13)$$

The set of feature points detected *within the ROI* at time t is denoted $F_{t,f}$, where the subscript f represents the type of feature under consideration (such as corners or edges). At each frame in the video we calculate the *total aligned flow* as follows:

$$a_{t,f} = \sum_{\mathbf{p} \in F_{t,f}} \hat{\mathbf{v}}_t(\mathbf{p}) \quad (6.14)$$

The video sequence is split into a series of time windows enumerated by n . The

set of frames belonging to the n th window is denoted W_n , and each window is taken to be the same length. Across each time window W_n the total aligned flow is accumulated:

$$\alpha_{n,f} = \sum_{t \in W_n} a_{t,f} \quad (6.15)$$

Although feature points are not explicitly tracked, this summation will be roughly proportional to the number of feature points crossing the counting line. This is because the summation of aligned flow for each feature point is equal to the distance it travels through the gate (i.e. the width of the ROI). Figure 6.4 illustrates this: a feature point passing through the gate is described by a series of optical flow vectors (displacements), the aligned components of which sum to the width w of the ROI.

Consequently each feature passing through the gate contributes approximately the same quantity of aligned flow, so that the total aligned flow over a time window, $\alpha_{n,f}$, is proportional to the number of features crossing the gate (with proportionality constant w). Thus the summation of aligned flow over a time window is a measure of the number of passing feature points. Furthermore, the use of flow accumulation rather than explicit feature tracking reduces the computational overhead involved with tracking.

6.3.2 Optical Flow Histograms

This section describes how optical flow histograms can be used to further improve system performance by separating the effects of potential noise and true motion.

In practice the values of optical flow are not always accurate and may be subject

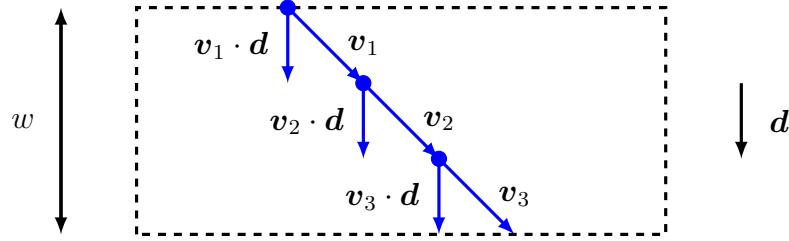


Figure 6.4: A feature point passes through the ROI. Optical flow vectors describe its displacement between consecutive frames. The component of the flow in direction d contributes to the *aligned* flow. These components sum to w approximately.

to error or noise. Consider a feature point belonging to a background object which does not move. The true value of optical flow is 0 but in practice may be assigned a small fractional value such as 0.02. In order to separate the effects of noise and true motion, a histogram based on flow magnitude is proposed. Aligned optical flow (Equations 6.13-6.14) is calculated within different histogram bins as follows:

1. Each pixel \mathbf{p} is assigned to a histogram bin b based on the magnitude of $\hat{\mathbf{v}}_t(\mathbf{p})$. Typical bin ranges are $[0, 0.05)$, $[0.05, 0.25)$ and $[0.25, \infty)$ and these values are used in this thesis. The set of pixels belonging to bin b is denoted H_b .
2. The total aligned flow for bin b and for feature f is denoted:

$$a_{t,f,b} = \sum_{\mathbf{p} \in F_{t,f} \cap H_b} \hat{\mathbf{v}}_t(\mathbf{p}) \quad (6.16)$$

3. The total aligned flow across a time window W_n for bin b and feature f is then calculated as follows:

$$\alpha_{n,f,b} = \sum_{t \in W_n} a_{t,f,b} \quad (6.17)$$

The set of all features and histogram bins, $\{\alpha_{n,f,b}\}_{f,b}$, are collected into a feature vector \mathbf{x}_n which describes the time window W_n . Note that when multiple features and histogram bins are used, these features are *concatenated* to form the larger feature vector. A regression model is then trained to learn the relationship between the feature vectors $\{\mathbf{x}_n\}$ and the ground truth values $\{g_n\}$. This is described in Section 6.3.4.

6.3.3 Feature Selection

In this section a number of features are proposed for use with the virtual gate algorithm. A number of image features were previously used for crowd counting (Chapter 3) and these were categorised into *size*, *shape*, *edge* and *keypoint* features. A similar selection of features is adopted for the virtual gate algorithm.

The following features are used in this chapter:

1. **All Pixels.** The use of all pixels treats every pixel within the ROI as if it is a feature point. Consequently, all pixels within the ROI are used to accumulate the total aligned flow in Equation 6.14. This approach is similar to existing methods in the literature which use regression of optical flow [98].

This feature is analogous to the ‘size’ feature in the crowd counting algorithm, because all pixels within a segment are used, and all pixels contribute to the value of the feature.

2. **Edges.** As described in Section 3.2.3.4, edges have been commonly used in crowd counting systems. Canny edge detection [32] is used to detect edges within the ROI and these pixels are used to accumulate the total aligned optical flow.

3. **Corners.** As discussed in Section 3.2.3.5, keypoints are often indicative of human crowding. Corners are chosen here because of their use in popular feature trackers, such as the Kanade-Lucas-Tomasi (KLT) tracker [117, 186]. In order to achieve rapid processing speed, Rosten’s FAST algorithm [160, 161] is used for corner detection within the proposed framework.

The above features are analogous to the size, edge and keypoint features which were used within the crowd counting algorithm of Chapter 3. Shape based features (Section 3.2.3.3) were also used for crowd counting; these features are derived from the perimeter pixels of each foreground segment. Foreground segmentation was achieved using adaptive background modelling techniques, which is not employed within the proposed virtual gate. Instead, the virtual gate makes use of optical flow, for which there is no equivalent shape feature. Consequently, three types of feature are evaluated in this chapter: all pixels, edges and corners.

Figure 6.5 depicts the relationship between these features and the number of people passing through the gate within each window of time. Although the relationship is approximately linear, the presence of noise and local non-linearities warrants the use of multiple features and non-linear regression techniques such as GPR.

6.3.4 Ground Truth Annotation and System Training

Ground truth is annotated via a graphical user interface to identify the time stamp of each pedestrian passing through the gate. These annotations are then assigned automatically to windows so that the number of people passing through the gate during each window is obtained automatically. A window length of 30 seconds was used for the proposed system.

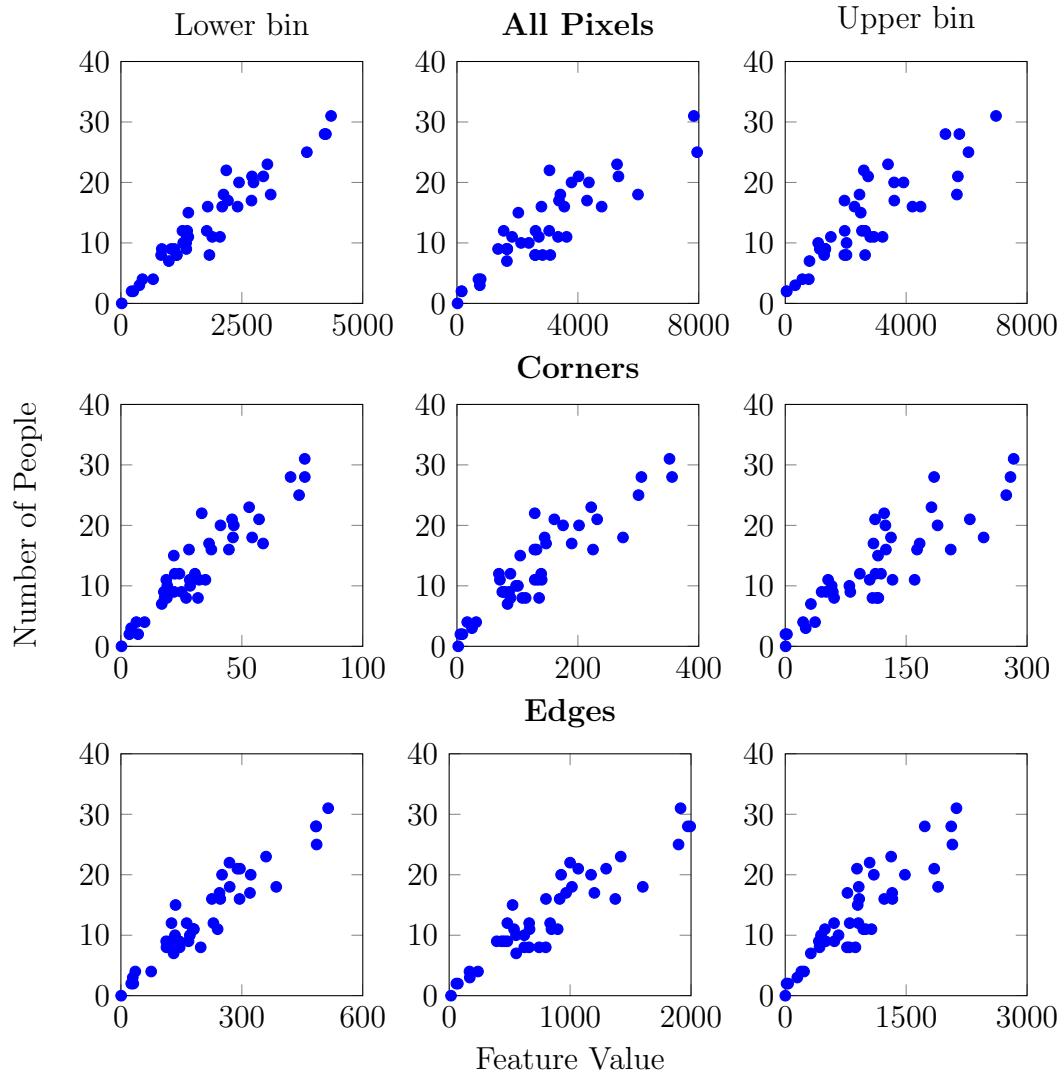


Figure 6.5: Relationship between features and number of people passing through the gate for **Camera C** (as described in Section 6.4.1). The lower histogram bin with range $[0, 0.05]$ is shown in the left hand column; the middle histogram bin with range $[0.05, 0.25]$ is shown in the middle column; and the upper histogram bin with range $[0.25, \infty)$ is shown in the right hand column.

Let the number of annotated people passing through the gate during time window W_n be denoted g_n . A corresponding set of features, concatenated into the feature vector \mathbf{x}_n , is extracted for each window. These features and targets form the training data set, $\{\mathbf{x}_n, g_n\}$, and a regression model is used to learn the relationship between the two.

As described in Section 3.2.5, Gaussian process regression (GPR) does not place any prior assumptions on the relationship between the feature space and target values, and is therefore ideal for crowd monitoring applications such as this. GPR was shown to achieve consistently superior performance for crowd counting (Sections 3.3.3.2 and 4.3.2.2) compared to linear regression, neural networks and K -nearest neighbours.

For comparison, all of the above regression models are used within the virtual gate framework, and the results of this analysis are presented in Section 6.4.2.2.

6.4 Experimental Results

In this section the proposed virtual gate algorithm and queue monitoring framework are evaluated using a number of datasets obtained from the Brisbane International Airport. Section 6.4.1 presents these benchmark datasets, Section 6.4.2 evaluates the performance of the proposed virtual gate algorithm on a variety of viewpoints and Section 6.4.3 evaluates the proposed queue monitoring framework specifically on queue footage.

	Camera A	Camera B	Camera C	Camera D	Camera E
Camera ID	2122	2341	2355	2365	2729
Sequences	$10 \times 3:00$	$8 \times 3:00$	$8 \times 3:00$	$8 \times 3:00$	$20 \times 3:00$
Throughput	323	419	574	503	401
Resolution	600×800	720×576	600×800	720×576	600×800
Frame Rate					
Original	30	30	30	30	30
Downsampled	6	6	6	6	6

Table 6.2: The benchmark datasets used to evaluate the proposed virtual gate algorithm. The total number of sequences and their duration is listed, as well as the throughput across all sequences (number of pedestrians passing through the gate).

6.4.1 Benchmark Datasets

Existing crowd flow monitoring algorithms have been evaluated on a variety of different datasets which are not publicly available, making a comprehensive comparison difficult. Those datasets which are publicly available such as PETS 2009 [149] do not contain sufficient data to evaluate the virtual gate across multiple sequences, as is required for a rigorous analysis.

To evaluate the proposed virtual gate algorithm, five benchmark datasets were collected and annotated from the Brisbane International Airport. These datasets are summarised in Table 6.2, and are referred to as Cameras A-E for brevity. The datasets are divided into a number of individual sequences, each of length 3 minutes. The algorithm is evaluated using a cross validation procedure as described in Section 6.4.2.

Camera A is positioned at the exit to the secondary examination (customs bag search) as shown in Figure 6.6. Cameras B, C and D are located above airport concourses and capture free flowing pedestrian traffic, as shown in Figures 6.7-6.9. Finally, Camera E is positioned above the arrival exit on the public side as



Figure 6.6: Camera A



Figure 6.7: Camera B

shown in Figure 6.10.

To evaluate the proposed queue monitoring framework, surveillance footage was obtained from the airport's baggage check-in counter. This dataset is referred to



Figure 6.8: Camera C

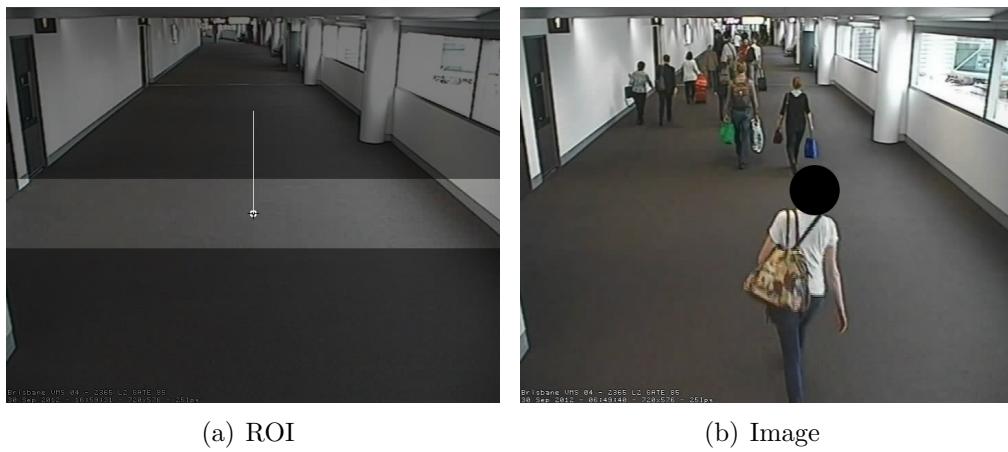


Figure 6.9: Camera D

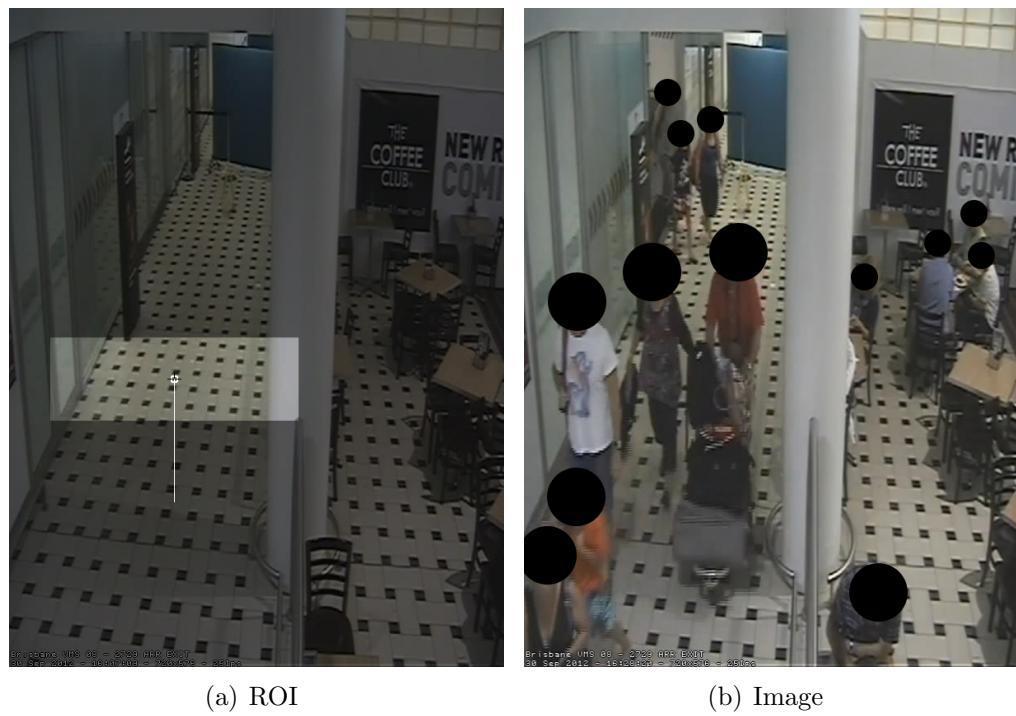


Figure 6.10: Camera E

	Sequence A	Sequence B
Length	12:00	20:00
Frames	9000	15000
Queue Length	26 to 32	19 to 28
Arrivals	35	37
Services	27	57
Resolution	704×576	704×576
Frame Rate		
Original	12.5	12.5
Downsampled	4.17	4.17

Table 6.3: The benchmark datasets used to evaluate the proposed queue monitoring algorithm.

as the ‘Queue dataset’ and is comprised of two sequences called ‘Sequence A’ and ‘Sequence B’, as summarised in Table 6.3. Sequence A is of length 12 minutes, containing 35 passenger arrivals, 27 passengers serviced and 26-32 people in the queue. Sequence B is of length 20 minutes, containing 37 passenger arrivals, 57 passengers services and 19-28 people in the queue.

The footage contains queues located at the baggage check-in counter, containing both human and non-human objects (i.e. luggage). This makes crowd counting and crowd flow monitoring particularly challenging. For each experiment, a cross validation procedure was followed, as described in Section 3.3.3.

6.4.2 Virtual Gate

In this section the performance of the virtual gate is assessed. A cross validation procedure is followed, as described in Section 3.3.3. For each dataset, this procedure can be summarised as follows:

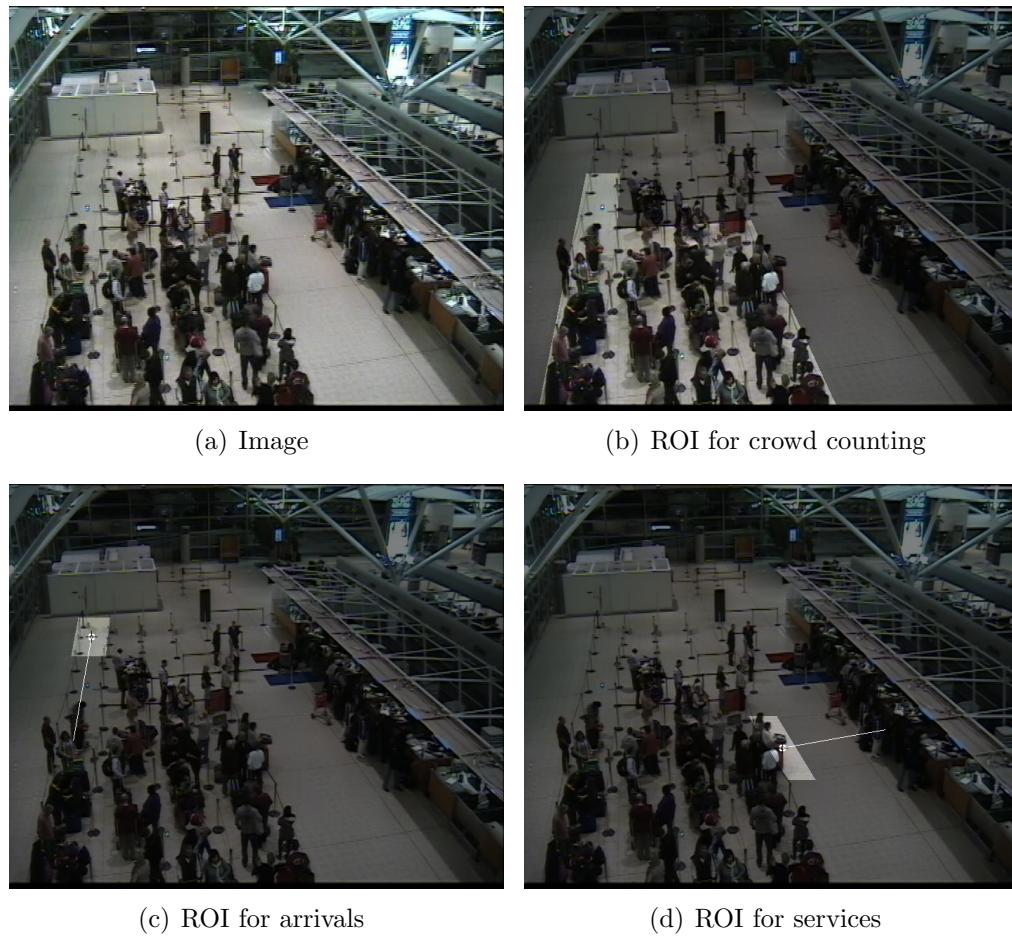


Figure 6.11: Queue dataset

1. The dataset is annotated with ground truth data: the number of pedestrians passing through the gate during each 3 minute sequence is annotated, as well as the number of pedestrians passing through the gate during each 30 second window.
2. One sequence is set aside for testing. The proposed virtual gate algorithm is trained using ground truth from the *other* sequences. Testing is then performed on the selected test sequence.
3. This procedure is repeated for all sequences within the dataset, by rotating the training and testing sets as depicted in Figure 3.19 (p. 161). Error metrics are averaged across all sequences: mean absolute error (MAE), mean square error (MSE) and mean relative error (MRE) are reported.

Each dataset is evaluated separately using the above procedure. Error metrics are then combined across all sequences using average rank, with equal weighting given to each dataset. This procedure is used to evaluate feature selection in Section 6.4.2.1 and regression models in Section 6.4.2.2.

6.4.2.1 Feature Evaluation

The features used in this evaluation are referred to as ‘All’ (all pixels), ‘Edges’ and ‘Corners’, as explained in Section 6.3.3. Seven different combinations of these features are assessed, as shown in Tables 6.4–6.8. Furthermore, the following system parameters are evaluated in this section:

1. **Optical Flow Histograms**, as described in Section 6.3.2, are indicated under the ‘Histogram’ heading. The ‘✓’ symbol indicates the use of three histogram bins, with ranges $[0, 0.05]$, $[0.05, 0.25]$ and $[0.25, \infty)$. The omis-

sion of this symbol indicates the absence of optical flow histograms, or equivalently, a single histogram bin with range $[0, \infty)$.

2. **Windows**, as described in Section 6.3.1, are indicated under the ‘Windows’ heading. The ‘✓’ symbol indicates that each 3 minute sequence has been divided into six windows of length 30 seconds, which are evaluated separately (during both training and testing). The omission of this symbol indicates that each sequence is analysed holistically. The use of windows is analogous to the use of local features in Chapter 3.

This gives rise to 28 different combinations of features and parameters, as listed in Tables 6.4-6.8. For each cross validation experiment, the feature-parameter combinations are ranked from 1 (best) to 28 (worst) in terms of MAE, MSE and MRE. The regression model used in these experiments is GPR.

The results for Camera A are tabulated in Table 6.4. The top-ranking feature vector is ‘Corners, All’ with histograms and windows being used. The omission of histograms causes the ranking to fall from 1 to 13, and the omission of windows causes it to fall to 21. The mean relative error is between 9.86% and 12.30% for the best performing feature sets. These are indicated in bold.

The results for Camera B are tabulated in Table 6.5. The top-ranking feature vectors include ‘Edges’ and ‘Edges, All’. Once again, the omission of histograms or windows reduce performance significantly. The mean relative error is less than 10% for these sequences.

Camera C is summarised in Table 6.6, with ‘All’ pixels attaining the highest ranking in terms of MAE and MSE; and ‘Corners, Edges’ attaining the highest ranking in terms of MRE. For these sequences the mean relative error is less than 10% and the use of histograms and windows are crucial for good performance.

Camera D is summarised in Table 6.7. ‘Corners, Edges’ (with histograms and windows) achieved the top ranking by all three error metrics, with a MRE of 10.79%. Finally, Camera E is presented in Table 6.8, for which ‘All’ pixels (with histograms and windows) achieved top ranking and a MRE of 10.94%.

These results demonstrate that the use of time windows and optical flow histograms improve the performance of the virtual gate on every dataset tested, and in some cases their inclusion is critical to good performance. In order to compare feature vectors across all datasets, the results are pooled by comparing the *average rank* in Table 6.9. The top-ranking feature in each column is underlined, and the top three features are indicated in bold.

The best performing feature vector is ‘Edges, All’. This feature vector achieves an average rank of 3.20 (out of 28) in terms of MAE, 4.20 in terms of MSE and 4.00 in terms of MRE. Table 6.9 also presents results using linear regression on the right hand side, for completeness. Once again the top-ranking feature is ‘Edges, All’. Due to the consistently good performance of this feature vector across all datasets, it is selected for use in the subsequent analyses.

The inclusion of ‘Corners’ does not confer any additional advantage. This is most likely due to the sparseness of these keypoints in an image: relatively few corners are detected inside the narrow ROI compared to edges (or all pixels). Nonetheless, relatively good performance is still observed with the ‘Corners, All’ feature vector as well as ‘Corners, Edges, All’.

Figure 6.12 plots the estimate of the proposed algorithm (using the ‘Edges, All’ feature vector) against the ground truth for each sequence in Camera A. Both the GPR and linear regression estimates are shown. Similarly, Figure 6.13 plots the Camera B sequences. Note that a negative value occurs because a pedestrian passes in the opposite direction, and the system estimates a negative value by

extrapolating beyond the training range as desired. Figure 6.14 plots the sequences from Camera C, Figure 6.15 plots the sequences from Camera D and Figure 6.16 plots the sequences from Camera E. These figures indicate that the proposed algorithm can perform accurate crowd flow monitoring across a wide range of conditions. The underestimation in some sequences (such as Figure 6.15, Sequence 6) is due to greater level of occlusion than the other sequences. This is depicted in Figure 6.17.

Features	Histogram	Windows	Error					
			MAE		MSE		MRE	
Corners	✓	✓	4.832	(5)	33.524	(5)	15.07%	(7)
	✓		31.487	(28)	1260.211	(28)	95.81%	(28)
		✓	5.034	(7)	38.696	(6)	15.67%	(9)
			5.414	(10)	42.003	(10)	16.16%	(16)
Edges	✓	✓	5.985	(19)	45.724	(14)	19.42%	(19)
	✓		22.626	(26)	807.727	(26)	71.01%	(25)
		✓	5.012	(6)	40.918	(8)	14.80%	(5)
			5.462	(12)	42.176	(11)	16.07%	(15)
All	✓	✓	3.332	(2)	19.728	(3)	9.86%	(1)
	✓		16.718	(23)	561.187	(23)	59.02%	(24)
		✓	5.432	(11)	46.956	(17)	15.56%	(8)
			5.696	(18)	51.317	(19)	15.83%	(12)
Corners, Edges	✓	✓	5.684	(17)	42.557	(12)	18.77%	(18)
	✓		21.950	(25)	707.936	(24)	76.44%	(26)
		✓	5.053	(8)	40.866	(7)	15.07%	(6)
			5.366	(9)	40.920	(9)	15.76%	(10)
Corners, All	✓	✓	3.288	(1)	17.076	(1)	10.56%	(2)
	✓		12.805	(21)	291.772	(21)	55.75%	(22)
		✓	5.466	(13)	44.979	(13)	15.90%	(13)
			7.454	(20)	134.117	(20)	21.96%	(20)
Edges, All	✓	✓	3.583	(4)	20.126	(4)	13.31%	(4)
	✓		24.668	(27)	897.832	(27)	76.57%	(27)
		✓	5.571	(16)	48.149	(18)	16.33%	(17)
			5.546	(15)	46.037	(15)	15.81%	(11)
Corners, Edges, All	✓	✓	3.384	(3)	18.111	(2)	12.30%	(3)
	✓		19.524	(24)	724.837	(25)	57.13%	(23)
		✓	5.477	(14)	46.520	(16)	16.03%	(14)
			13.308	(22)	518.226	(22)	35.39%	(21)

Table 6.4: Comparison of features, histograms and windows on **Camera A**.

Features	Histogram	Windows	Error					
			MAE		MSE		MRE	
Corners	✓	✓	3.912	(3)	27.612	(2)	11.83%	(6)
	✓		36.422	(20)	2259.019	(20)	78.90%	(20)
		✓	11.314	(14)	194.692	(17)	69.00%	(18)
			12.755	(17)	185.069	(16)	60.68%	(16)
Edges	✓	✓	3.684	(2)	22.065	(1)	7.43%	(1)
	✓		47.967	(22)	3418.854	(22)	87.82%	(25)
		✓	10.303	(13)	153.620	(13)	66.06%	(17)
			11.743	(15)	158.119	(14)	50.33%	(13)
All	✓	✓	5.970	(7)	55.347	(11)	20.13%	(7)
	✓		51.631	(25)	3782.209	(24)	87.25%	(23)
		✓	6.626	(11)	52.762	(10)	38.82%	(12)
			23.398	(18)	1552.223	(18)	37.68%	(11)
Corners, Edges	✓	✓	4.232	(5)	29.135	(3)	8.47%	(3)
	✓		50.065	(24)	4340.804	(28)	87.06%	(22)
		✓	9.414	(12)	132.737	(12)	59.29%	(15)
			12.143	(16)	168.183	(15)	54.05%	(14)
Corners, All	✓	✓	4.175	(4)	42.920	(6)	10.40%	(5)
	✓		53.948	(27)	3796.523	(25)	96.31%	(27)
		✓	6.145	(9)	50.808	(8)	35.60%	(9)
			45.666	(21)	3149.742	(21)	87.69%	(24)
Edges, All	✓	✓	3.666	(1)	31.582	(4)	8.27%	(2)
	✓		48.166	(23)	3433.225	(23)	85.38%	(21)
		✓	6.292	(10)	52.113	(9)	37.29%	(10)
			53.932	(26)	3800.052	(26)	98.18%	(28)
Corners, Edges, All	✓	✓	4.414	(6)	42.909	(5)	8.87%	(4)
	✓		32.740	(19)	1890.042	(19)	76.57%	(19)
		✓	5.999	(8)	47.644	(7)	34.97%	(8)
			54.440	(28)	3854.808	(27)	94.23%	(26)

Table 6.5: Comparison of features, histograms and windows on **Camera B**.

Features	Histogram	Windows	Error					
			MAE		MSE		MRE	
Corners	✓	✓	8.293	(13)	108.367	(13)	12.34%	(8)
	✓		58.385	(22)	4868.251	(23)	105.32%	(26)
		✓	9.007	(14)	125.663	(14)	17.66%	(14)
			46.652	(18)	2734.472	(17)	77.15%	(16)
Edges	✓	✓	5.721	(2)	61.257	(2)	9.68%	(4)
	✓		49.587	(20)	3541.611	(20)	78.85%	(18)
		✓	7.629	(10)	91.381	(7)	13.33%	(12)
			75.696	(28)	6784.929	(28)	107.42%	(27)
All	✓	✓	5.271	(1)	36.085	(1)	12.28%	(7)
	✓		60.262	(23)	4569.810	(21)	93.64%	(22)
		✓	7.903	(12)	90.795	(5)	13.93%	(13)
			71.498	(25)	6582.336	(27)	102.77%	(25)
Corners, Edges	✓	✓	6.380	(4)	91.795	(8)	8.12%	(1)
	✓		39.785	(16)	2221.816	(15)	84.84%	(21)
		✓	7.503	(8)	98.267	(11)	11.52%	(6)
			69.811	(24)	5788.151	(24)	152.63%	(28)
Corners, All	✓	✓	6.778	(5)	67.552	(3)	13.32%	(11)
	✓		45.543	(17)	2836.234	(18)	77.64%	(17)
		✓	7.616	(9)	98.211	(10)	13.15%	(10)
			73.413	(27)	6433.622	(26)	102.22%	(24)
Edges, All	✓	✓	6.255	(3)	68.505	(4)	8.94%	(3)
	✓		38.859	(15)	2335.429	(16)	72.19%	(15)
		✓	7.862	(11)	90.977	(6)	13.05%	(9)
			72.252	(26)	6210.541	(25)	101.53%	(23)
Corners, Edges, All	✓	✓	7.420	(6)	103.781	(12)	8.59%	(2)
	✓		48.559	(19)	3278.461	(19)	82.69%	(20)
		✓	7.493	(7)	96.528	(9)	11.49%	(5)
			54.247	(21)	4581.324	(22)	79.00%	(19)

Table 6.6: Comparison of features, histograms and windows on **Camera C**.

Features	Histogram	Windows	Error					
			MAE		MSE		MRE	
Corners	✓	✓	6.542	(4)	85.055	(2)	10.97%	(3)
	✓		55.851	(22)	5027.973	(22)	77.43%	(22)
		✓	11.488	(18)	160.775	(19)	47.64%	(20)
			10.301	(13)	161.975	(20)	24.33%	(12)
Edges	✓	✓	7.693	(5)	106.050	(6)	11.93%	(4)
	✓		73.995	(28)	6575.781	(28)	172.85%	(27)
		✓	11.634	(20)	156.863	(16)	46.14%	(19)
			9.986	(11)	155.758	(15)	23.45%	(9)
All	✓	✓	8.468	(8)	114.825	(8)	14.80%	(7)
	✓		57.612	(23)	5401.593	(23)	81.13%	(23)
		✓	10.953	(14)	128.484	(10)	41.14%	(16)
			8.193	(7)	101.138	(4)	23.25%	(8)
Corners, Edges	✓	✓	6.100	(1)	75.708	(1)	10.79%	(1)
	✓		66.114	(24)	5807.248	(26)	131.49%	(25)
		✓	11.630	(19)	158.523	(17)	45.87%	(18)
			10.174	(12)	159.511	(18)	23.94%	(11)
Corners, All	✓	✓	6.448	(2)	101.223	(5)	10.86%	(2)
	✓		68.447	(26)	6134.276	(27)	124.84%	(24)
		✓	11.278	(17)	141.976	(14)	40.35%	(15)
			9.509	(10)	133.242	(11)	24.49%	(13)
Edges, All	✓	✓	7.975	(6)	109.793	(7)	14.36%	(6)
	✓		66.988	(25)	5771.340	(24)	133.40%	(26)
		✓	11.041	(15)	136.842	(12)	40.27%	(14)
			9.128	(9)	126.469	(9)	23.71%	(10)
Corners, Edges, All	✓	✓	6.538	(3)	92.903	(3)	12.14%	(5)
	✓		69.613	(27)	5772.124	(25)	189.59%	(28)
		✓	11.150	(16)	138.926	(13)	41.16%	(17)
			52.713	(21)	4892.769	(21)	73.54%	(21)

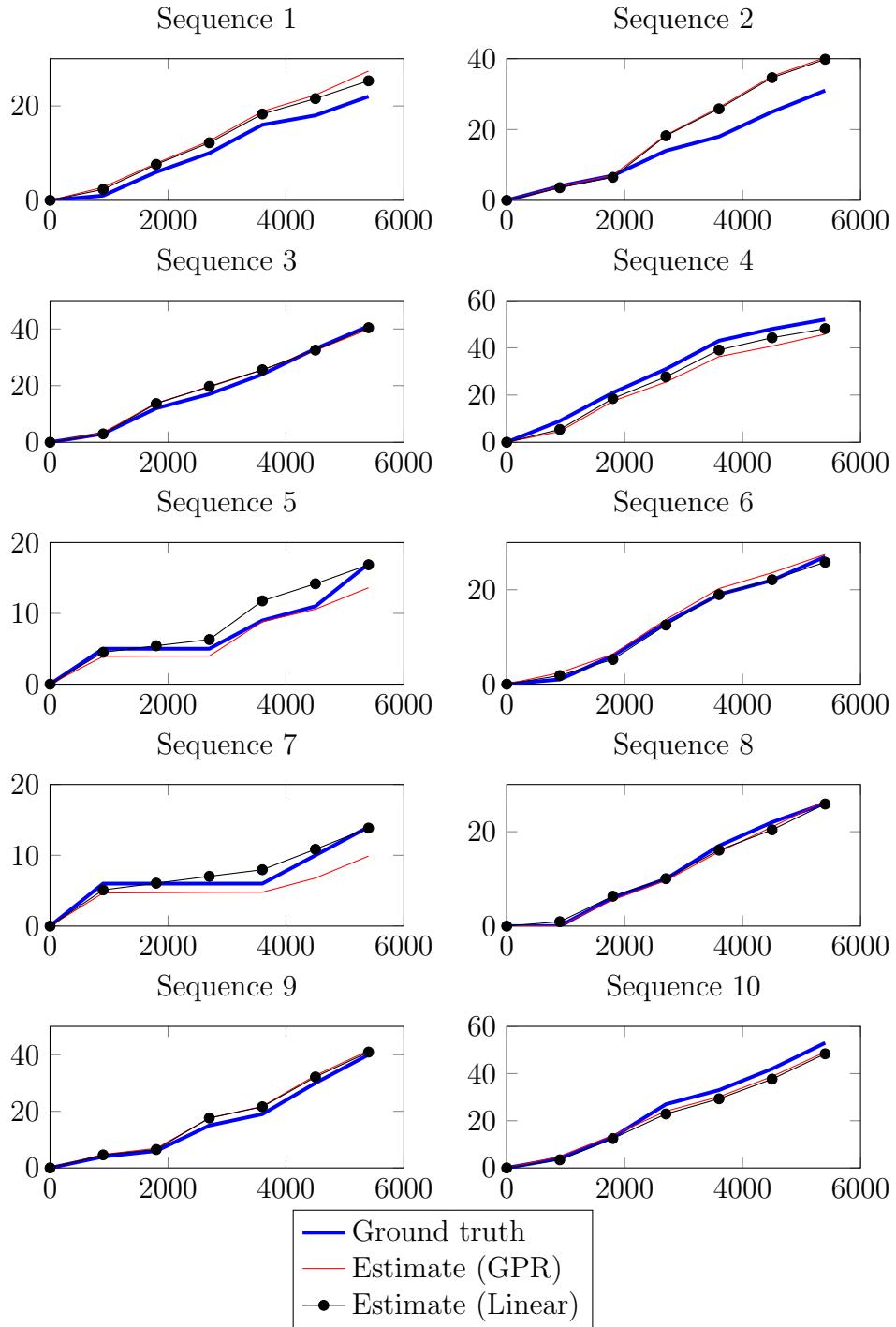
Table 6.7: Comparison of features, histograms and windows on **Camera D**.

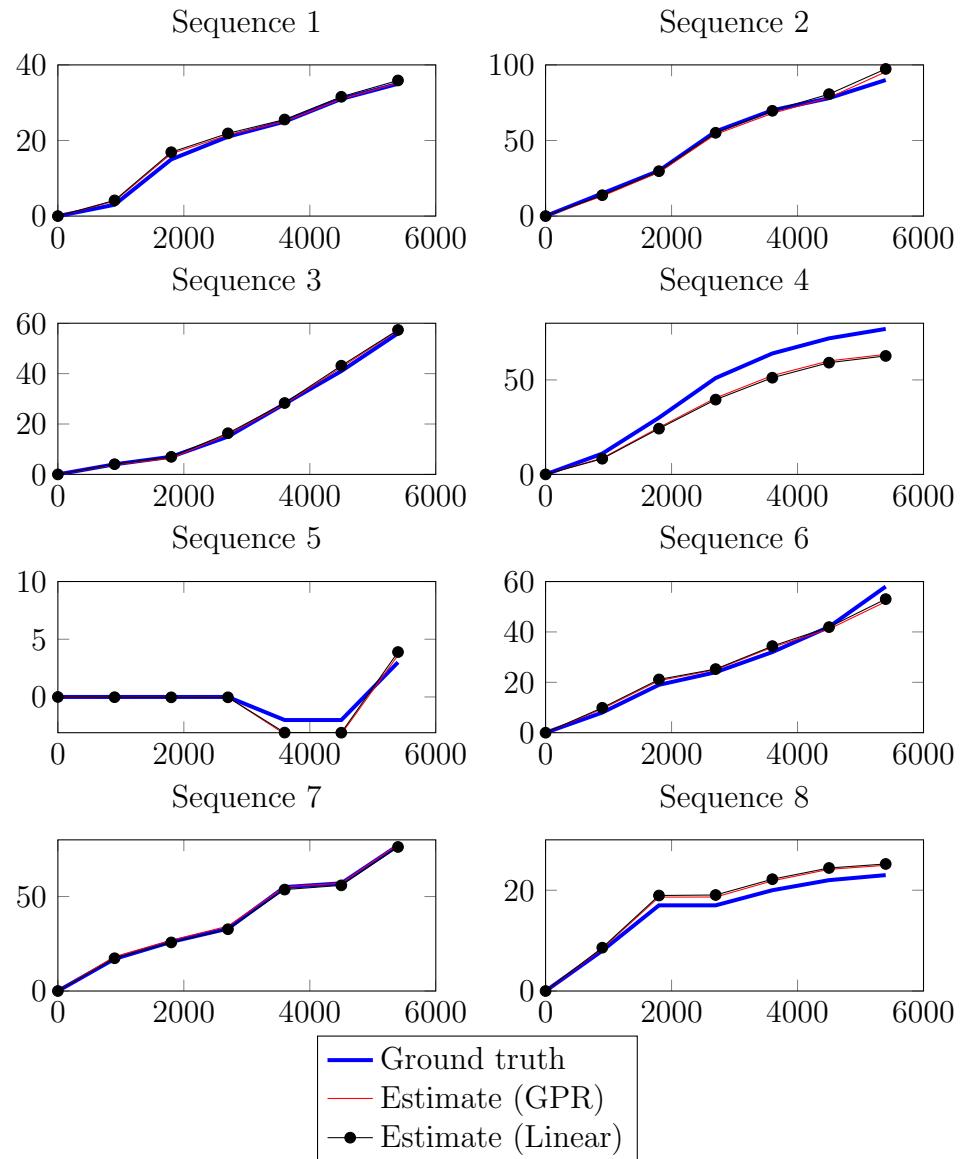
Features	Histogram	Windows	Error			
			MAE	MSE	MRE	
Corners	✓	✓	2.168 (14)	8.306 (11)	19.54% (16)	
	✓		7.873 (24)	154.855 (21)	63.86% (26)	
		✓	2.026 (9)	9.054 (16)	12.63% (3)	
			11.678 (28)	292.062 (24)	80.61% (28)	
Edges	✓	✓	2.144 (12)	8.979 (15)	19.17% (15)	
	✓		9.410 (26)	428.534 (27)	34.86% (24)	
		✓	1.956 (8)	8.916 (14)	14.11% (7)	
			11.090 (27)	579.415 (28)	46.75% (25)	
All	✓	✓	1.354 (1)	2.874 (1)	10.94% (1)	
	✓		3.253 (18)	66.527 (19)	18.87% (14)	
		✓	1.872 (6)	6.441 (8)	15.37% (9)	
			6.992 (23)	397.834 (26)	26.28% (19)	
Corners, Edges	✓	✓	2.312 (16)	8.696 (13)	33.44% (23)	
	✓		3.482 (19)	74.315 (20)	18.01% (13)	
		✓	1.905 (7)	6.642 (9)	13.08% (4)	
			6.253 (21)	252.641 (23)	30.33% (21)	
Corners, All	✓	✓	2.169 (15)	6.104 (7)	27.87% (20)	
	✓		2.154 (13)	8.670 (12)	13.45% (6)	
		✓	1.803 (5)	5.157 (4)	17.80% (12)	
			8.983 (25)	228.191 (22)	66.65% (27)	
Edges, All	✓	✓	1.390 (2)	3.433 (2)	13.40% (5)	
	✓		1.480 (3)	3.727 (3)	11.10% (2)	
		✓	2.034 (10)	7.106 (10)	14.63% (8)	
			6.950 (22)	298.441 (25)	32.23% (22)	
Corners, Edges, All	✓	✓	2.060 (11)	5.887 (6)	24.64% (18)	
	✓		3.133 (17)	35.581 (17)	16.05% (10)	
		✓	1.795 (4)	5.238 (5)	16.59% (11)	
			3.645 (20)	52.533 (18)	23.84% (17)	

Table 6.8: Comparison of features, histograms and windows on **Camera E**.

Features	Histogram	Windows	Average Rank					
			GPR			Linear		
			MAE	MSE	MRE	MAE	MSE	MRE
Corners	✓	✓	7.80	6.60	8.00	9.40	10.60	9.40
	✓		23.20	22.80	24.40	18.80	19.00	17.00
		✓	12.40	14.40	12.80	19.40	19.60	21.80
			17.20	17.40	17.60	20.20	20.40	21.40
Edges	✓	✓	8.00	7.60	8.60	8.40	8.20	7.60
	✓		24.40	24.60	23.80	19.20	18.40	17.40
		✓	11.40	11.60	12.00	17.00	15.20	18.60
			18.60	19.20	17.80	17.20	16.80	17.80
All	✓	✓	3.80	4.80	4.60	5.60	7.00	6.20
	✓		22.40	22.00	21.20	11.60	11.00	11.40
		✓	10.80	10.00	11.60	11.00	10.40	16.40
			18.20	18.80	15.00	11.80	11.60	15.20
Corners, Edges	✓	✓	8.60	7.40	9.20	9.80	10.80	8.20
	✓		21.60	22.60	21.40	24.40	24.00	24.40
		✓	10.80	11.20	9.80	17.00	16.00	16.60
			16.40	17.80	16.80	18.40	18.00	16.00
Corners, All	✓	✓	5.40	4.40	8.00	6.20	7.20	6.60
	✓		20.80	20.60	19.20	21.40	21.80	17.80
		✓	10.60	9.80	11.80	10.80	10.80	13.60
			20.60	20.00	21.60	16.40	16.00	16.60
Edges, All	✓	✓	3.20	4.20	4.00	5.00	5.60	5.60
	✓		18.60	18.60	18.20	15.40	16.00	12.80
		✓	12.40	11.00	11.60	13.00	12.60	13.00
			19.60	20.00	18.80	14.80	14.80	15.80
Corners, Edges, All	✓	✓	5.80	5.60	6.40	7.20	8.40	5.20
	✓		21.20	21.00	20.00	24.40	25.80	23.20
		✓	9.80	10.00	11.00	14.00	12.60	11.80
			22.40	22.00	20.80	18.20	17.40	18.60

Table 6.9: Average rank across **all datasets**, when all combinations of features, histograms and windows are ranked from 1 to 28 on each dataset. Values shown are not actual error rates, but rather an average ranking.

Figure 6.12: Cross validation results on **Camera A**.

Figure 6.13: Cross validation results on **Camera B**.

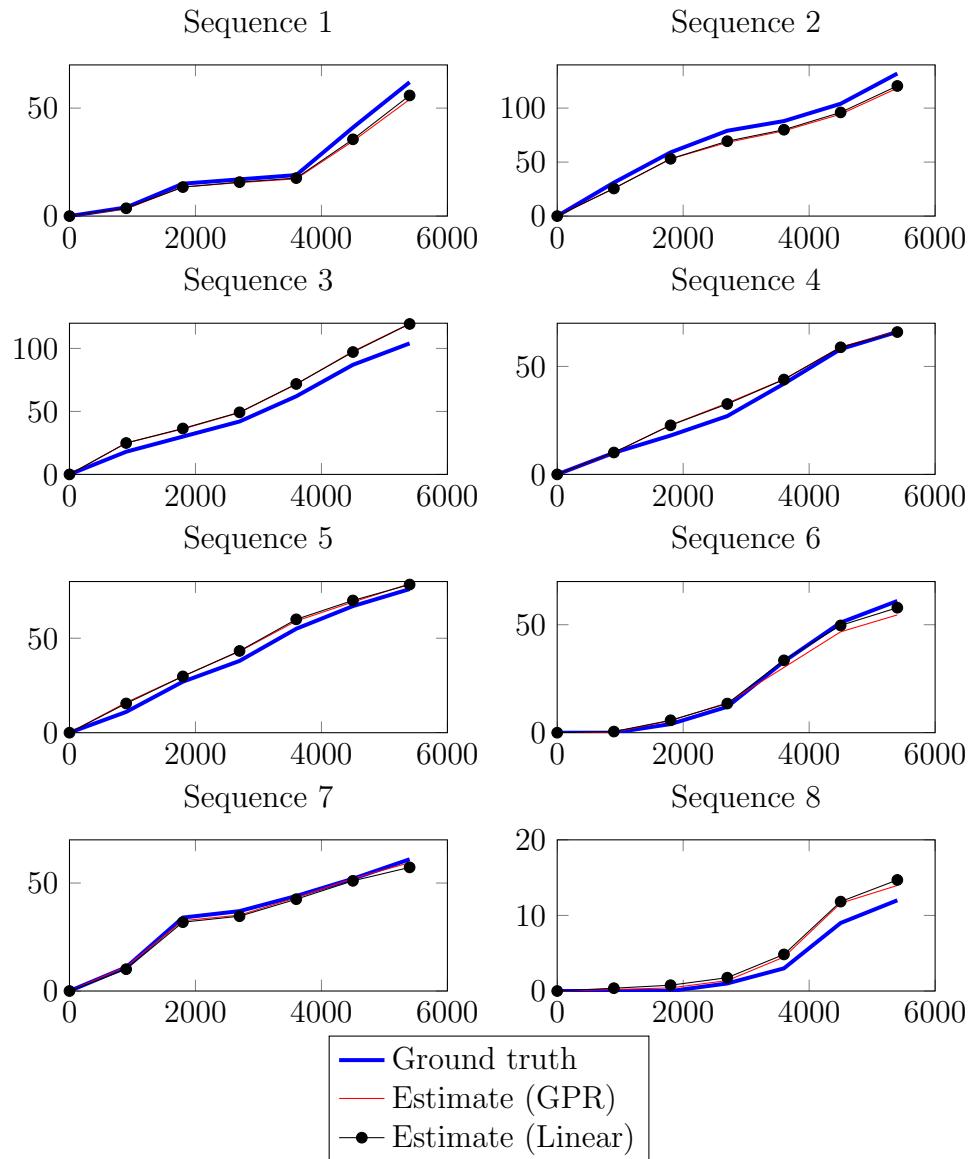
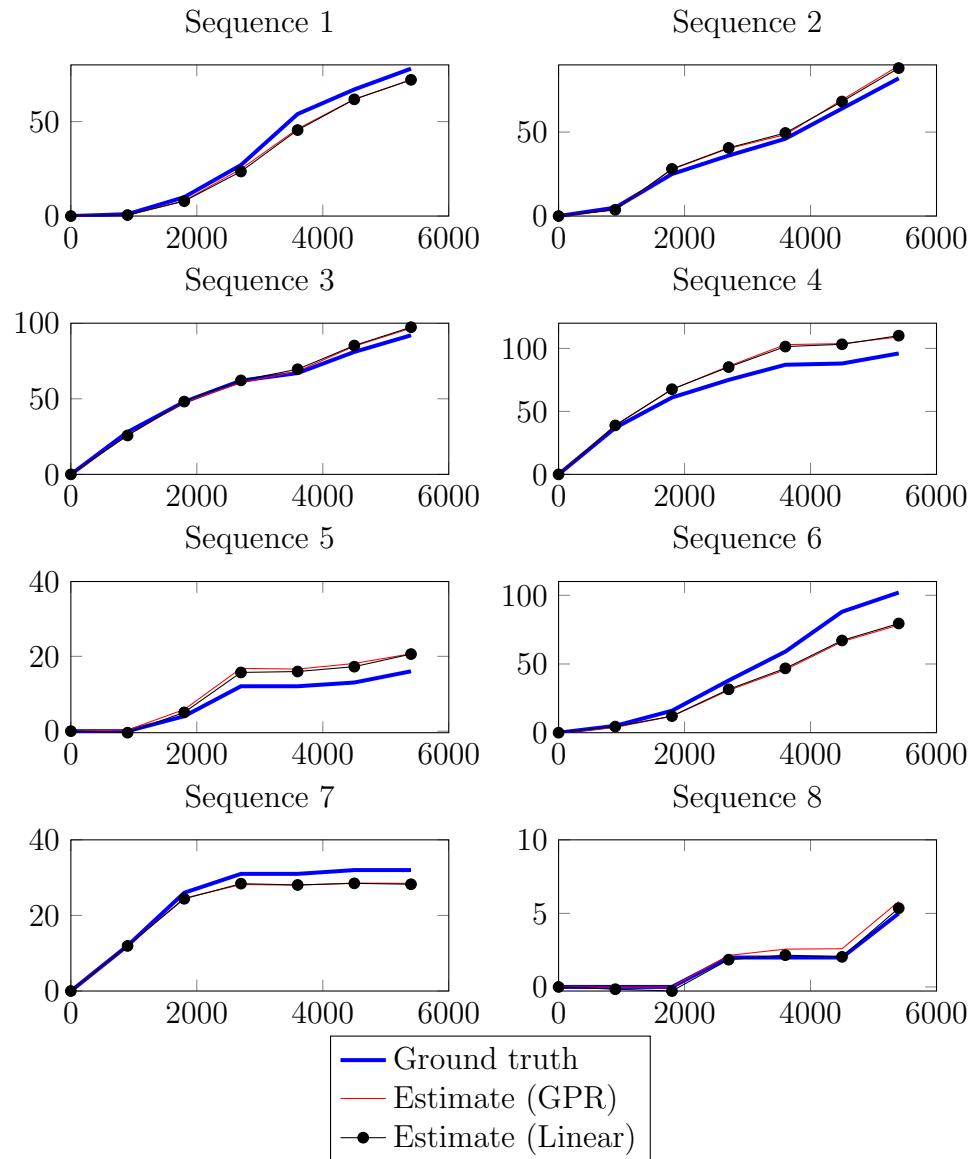


Figure 6.14: Cross validation results on **Camera C**.

Figure 6.15: Cross validation results on **Camera D**.

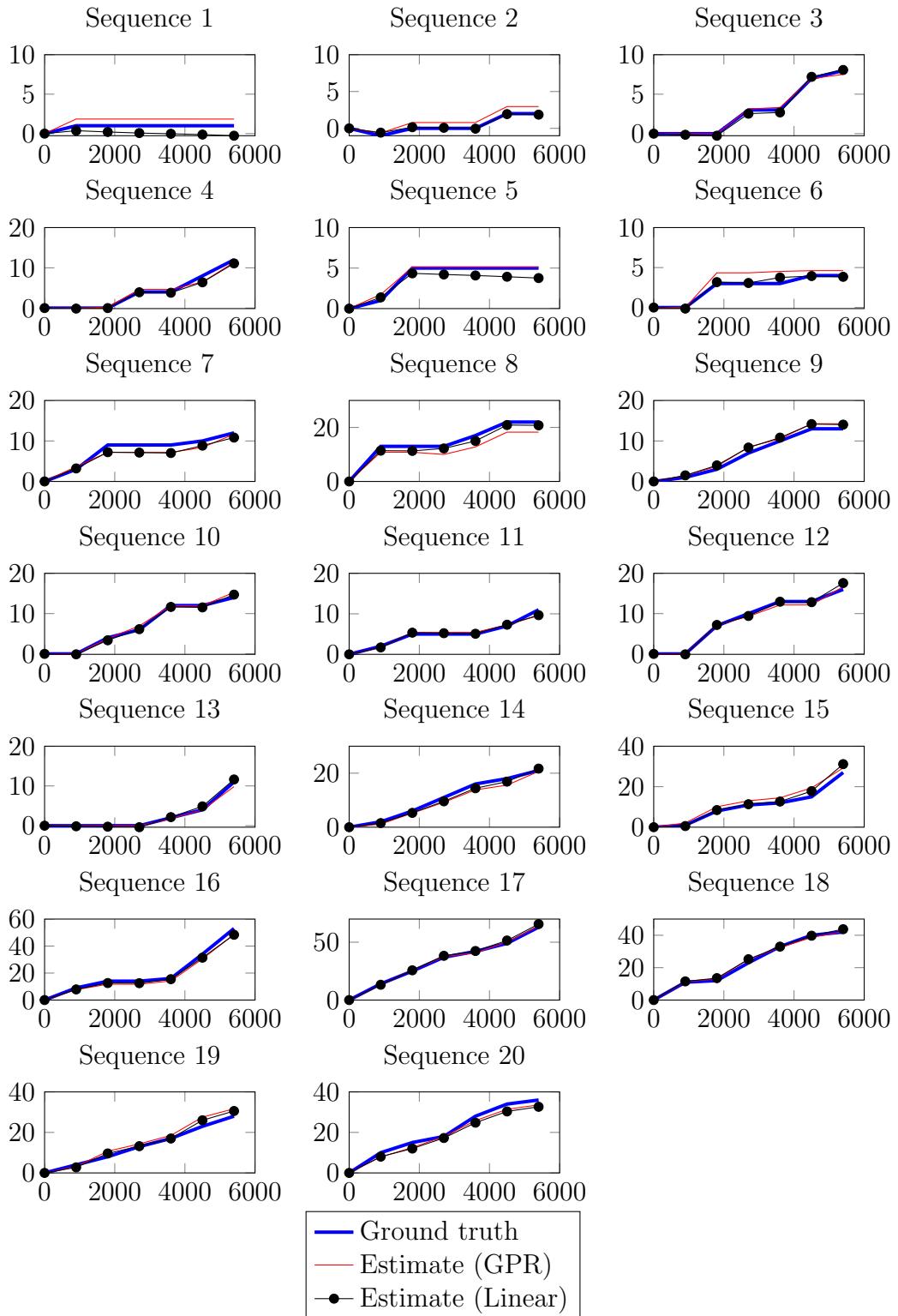
Figure 6.16: Cross validation results on **Camera E**.



Figure 6.17: Sequence 6, Frame 4200 from Camera D. A higher level of occlusion is observed for this sequence.

6.4.2.2 Regression Models

In this section, the K -fold cross validation experiments are repeated to evaluate a number of regression models within the context of the proposed virtual gate algorithm. For comparison, we use Gaussian process regression (GPR), least squares linear regression (Linear), K -Nearest Neighbours (KNN) with $K = 1, 2, 4, 8, 16, 32$, and a neural network (NN) with Sigmoid activation function and one hidden input layer (containing 4, 8, 16 or 32 neurons). In total there are 12 regression models with various parameters. The ‘Edges, All’ feature vector is used in conjunction with histograms and windows, due to the consistently good performance of this configuration in Section 6.4.2.1.

Table 6.10 summarises the results for Camera A for various regression models. Error rates are ranked from 1 to 12 in each column. The linear and GPR models provide the most accurate performance. Table 6.11 presents the results for Camera B, for which GPR and linear regression again offer best performance. The linear model performs best on Camera C (Table 6.12); K -nearest neighbours (with $K = 4$) performs best on Camera D (Table 6.13) and GPR performs best on Camera E (6.14). Across all viewpoints, GPR and linear regression are included in the top three ranking regression models in every case.

Results across all datasets are pooled in Table 6.15 and the *average rank* is reported. The linear model outperforms the other models with an average rank of 1.60 (out of 12) in terms of both MAE and MRE. The GPR model follows closely with an average rank of 2.00 in terms of MAE and 1.80 in terms of MRE.

These results indicate that the linear model provides the most accurate performance within the proposed virtual gate algorithm, followed very closely by Gaussian process regression. For this reason the linear regression model adopted in the subsequent analyses.

Regression Model	Error					
	MAE		MSE		MRE	
GPR	3.583	(2)	20.126	(2)	13.31%	(2)
Linear	2.375	(1)	12.811	(1)	7.03%	(1)
KNN (K=1)	5.200	(6)	38.400	(6)	17.02%	(9)
KNN (K=2)	5.100	(5)	34.650	(5)	15.40%	(4)
KNN (K=4)	5.475	(8)	43.481	(7)	16.44%	(6)
KNN (K=8)	5.763	(9)	52.058	(9)	16.89%	(8)
KNN (K=16)	6.537	(10)	63.844	(10)	21.80%	(10)
KNN (K=32)	8.566	(12)	115.437	(12)	27.50%	(12)
NN (4)	4.580	(4)	30.169	(4)	14.14%	(3)
NN (8)	6.596	(11)	81.107	(11)	23.16%	(11)
NN (16)	4.533	(3)	26.994	(3)	16.84%	(7)
NN (32)	5.345	(7)	48.745	(8)	15.83%	(5)

Table 6.10: Comparison of regression models on **Camera A**.

Regression Model	Error					
	MAE		MSE		MRE	
GPR	3.666	(1)	31.582	(1)	8.27%	(1)
Linear	4.087	(2)	36.491	(3)	10.06%	(2)
KNN (K=1)	6.750	(8)	74.250	(8)	30.84%	(4)
KNN (K=2)	7.500	(10)	77.813	(9)	39.46%	(6)
KNN (K=4)	5.188	(3)	48.297	(4)	53.24%	(8)
KNN (K=8)	7.078	(9)	80.771	(10)	83.32%	(10)
KNN (K=16)	11.891	(11)	186.320	(11)	123.90%	(11)
KNN (K=32)	20.844	(12)	574.365	(12)	206.81%	(12)
NN (4)	6.201	(6)	61.512	(6)	24.14%	(3)
NN (8)	5.671	(5)	64.468	(7)	63.15%	(9)
NN (16)	5.403	(4)	34.615	(2)	43.96%	(7)
NN (32)	6.534	(7)	55.272	(5)	32.35%	(5)

Table 6.11: Comparison of regression models on **Camera B**.

Regression Model	Error					
	MAE		MSE		MRE	
GPR	6.255	(3)	68.505	(4)	8.94%	(2)
Linear	5.638	(1)	55.576	(1)	8.80%	(1)
KNN (K=1)	8.000	(6)	85.500	(5)	10.69%	(3)
KNN (K=2)	8.625	(9)	113.563	(8)	13.10%	(5)
KNN (K=4)	7.594	(5)	97.758	(6)	13.64%	(6)
KNN (K=8)	8.328	(7)	156.201	(10)	22.90%	(9)
KNN (K=16)	12.734	(11)	337.364	(11)	40.04%	(11)
KNN (K=32)	20.336	(12)	853.446	(12)	66.81%	(12)
NN (4)	10.029	(10)	155.991	(9)	32.72%	(10)
NN (8)	8.590	(8)	102.301	(7)	18.32%	(7)
NN (16)	7.038	(4)	62.649	(3)	13.03%	(4)
NN (32)	6.067	(2)	61.641	(2)	21.11%	(8)

Table 6.12: Comparison of regression models on **Camera C**.

Regression Model	Error					
	MAE		MSE		MRE	
GPR	7.975	(3)	109.793	(3)	14.36%	(2)
Linear	7.837	(2)	105.357	(2)	13.15%	(1)
KNN (K=1)	9.375	(8)	123.625	(8)	29.50%	(7)
KNN (K=2)	8.500	(5)	110.000	(4)	27.47%	(5)
KNN (K=4)	7.688	(1)	91.438	(1)	24.92%	(4)
KNN (K=8)	9.719	(9)	127.398	(9)	30.34%	(8)
KNN (K=16)	14.828	(11)	237.719	(11)	66.54%	(10)
KNN (K=32)	32.074	(12)	1083.641	(12)	170.92%	(12)
NN (4)	8.293	(4)	117.780	(7)	49.00%	(9)
NN (8)	12.506	(10)	231.953	(10)	97.89%	(11)
NN (16)	8.664	(7)	110.673	(5)	20.44%	(3)
NN (32)	8.636	(6)	114.843	(6)	28.52%	(6)

Table 6.13: Comparison of regression models on **Camera D**.

Regression Model	Error					
	MAE		MSE		MRE	
GPR	1.390	(1)	3.433	(1)	13.40%	(2)
Linear	1.563	(2)	4.009	(2)	14.05%	(3)
KNN (K=1)	2.850	(5)	14.250	(5)	17.30%	(5)
KNN (K=2)	1.975	(3)	8.463	(3)	12.99%	(1)
KNN (K=4)	2.325	(4)	13.488	(4)	16.88%	(4)
KNN (K=8)	3.031	(8)	23.741	(9)	19.34%	(7)
KNN (K=16)	3.450	(10)	36.440	(10)	18.26%	(6)
KNN (K=32)	4.734	(11)	79.228	(12)	20.47%	(8)
NN (4)	2.870	(6)	14.816	(6)	35.23%	(11)
NN (8)	5.289	(12)	71.887	(11)	56.36%	(12)
NN (16)	3.153	(9)	19.129	(7)	27.12%	(10)
NN (32)	2.933	(7)	21.350	(8)	25.50%	(9)

Table 6.14: Comparison of regression models on **Camera E**.

Regression Model	Average Rank		
	MAE	MSE	MRE
GPR	2.00	2.20	1.80
Linear	1.60	1.80	1.60
KNN (K=1)	6.60	6.40	5.60
KNN (K=2)	6.40	5.80	4.20
KNN (K=4)	4.20	4.40	5.60
KNN (K=8)	8.40	9.40	8.40
KNN (K=16)	10.60	10.60	9.60
KNN (K=32)	11.80	12.00	11.20
NN (4)	6.00	6.40	7.20
NN (8)	9.20	9.20	10.00
NN (16)	5.40	4.00	6.20
NN (32)	5.80	5.80	6.60

Table 6.15: Average rank across **all datasets**, when regression models are ranked from 1 to 12 on each dataset. Values shown are not actual error rates, but rather an average ranking.

Sequence	MAE	MSE	MRE
A	2.969	12.059	10.38%
B	4.015	21.795	13.93%
<i>Average</i>	3.492	16.927	12.15%

Table 6.16: Results of queue length estimation using the crowd counting module.

6.4.3 Queue Monitoring

In this section the proposed queue monitoring algorithm is evaluated using the queue dataset described in Section 6.4.1. This dataset is depicted in Figure 6.11. It is comprised of two sequences, as summarised in Table 6.3. The footage contains queues located at the baggage check-in counter, containing both human and non-human objects.

In order to verify the operation of the crowd counting system in a queueing environment, cross validation is performed by training the system on Sequence A and testing it on Sequence B; and then training the system on Sequence B and testing it on Sequence A. Table 6.16 reports the performance of the algorithm on this dataset. Mean relative errors of 10.38% and 13.93% were observed for Sequences A and B respectively. This is consistent with the operational requirements established in Section 3.3.1 (p. 151).

Screenshots of the system operating on this dataset are shown in Figure 6.18. The counting module operates by segmenting a crowd into groups and estimating the number of pedestrians in each group, as shown in Figure 6.18(b). When the queue forms a large group, the system identifies it and estimates its size on a holistic level (Figure 6.18(c)).

The virtual gate algorithm is evaluated by monitoring arrivals at the end of the

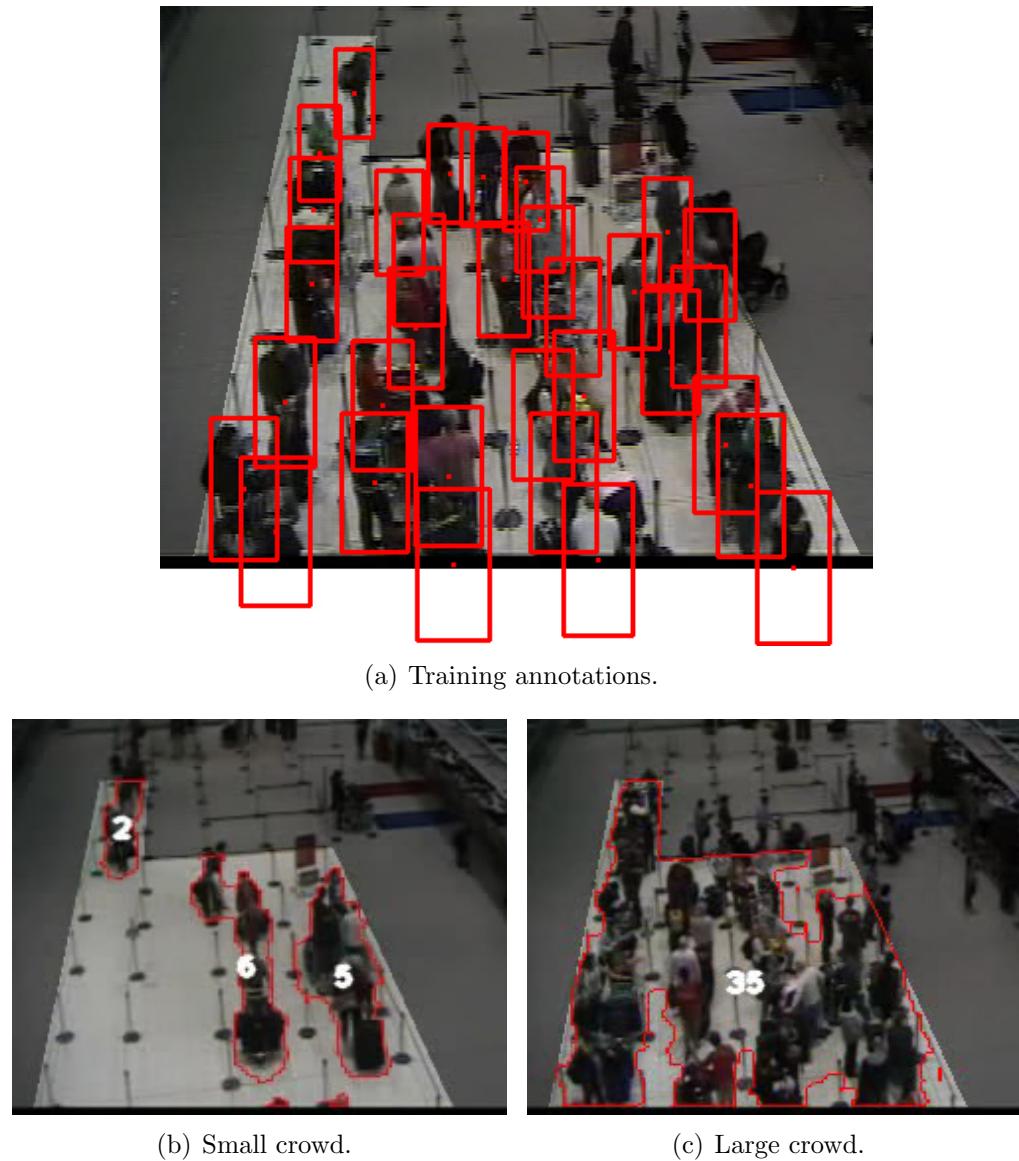


Figure 6.18: Queue length monitoring using the crowd counting module.

Virtual Gate	Error		
	MAE	MSE	MRE
Arrivals	1.296	3.115	23.69%
Services	1.368	2.891	32.04%

Table 6.17: Results of cross validation on the Queue dataset.

queue (Figure 6.11(c)) and services at the front of the queue (Figure 6.11(d)). The feature vector ‘Edges, All’ was selected; time windows of length 30 seconds were used; and three histogram bins were used as described in Section 6.3.2. The linear regression model was used due to its superior performance in Section 6.4.2.2.

The dataset is divided into 11 sequences of length 3 minutes, and the cross validation procedure of Section 6.4.2 was adopted. Table 6.17 reports the error rates for each of the virtual gates (arrivals and services) across these 11 sequences. Although the MRE is higher on this dataset than those reported in Section 6.4.2, this is due to the smaller crowds observed in these sequences; the mean absolute error across all sequences was only 1.296 for arrivals and 1.368 for services. The sequences are plotted in Figure 6.19.

Figure 6.20 depicts the long term performance of the proposed virtual gate algorithm. The 3 minute sequences are concatenated to show cumulative performance across sequences A and B for each virtual gate.

Finally, Figures 6.21 and 6.22 depict all three modules working together. The number of arrivals (A_t) and services (S_t) were estimated directly using the virtual gate algorithm while the queue length was estimated using the crowd counting module. The relationship between each of these modules can be seen in Sequence B (Figure 6.22); the queue length (Q_t) decreases as the number of services surpasses the number of arrivals.

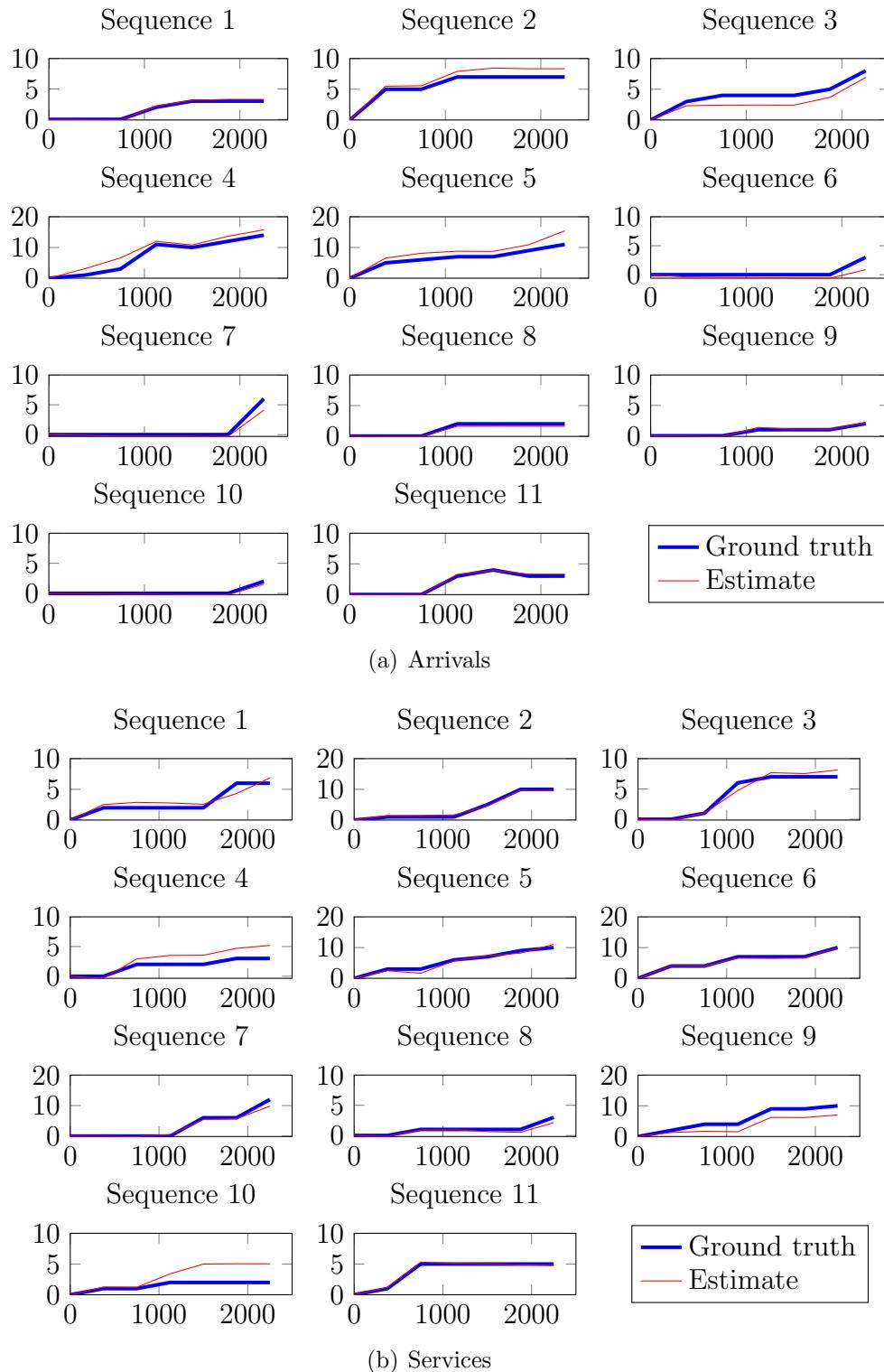


Figure 6.19: Cross validation results on the **Queue Dataset** using the virtual gate.

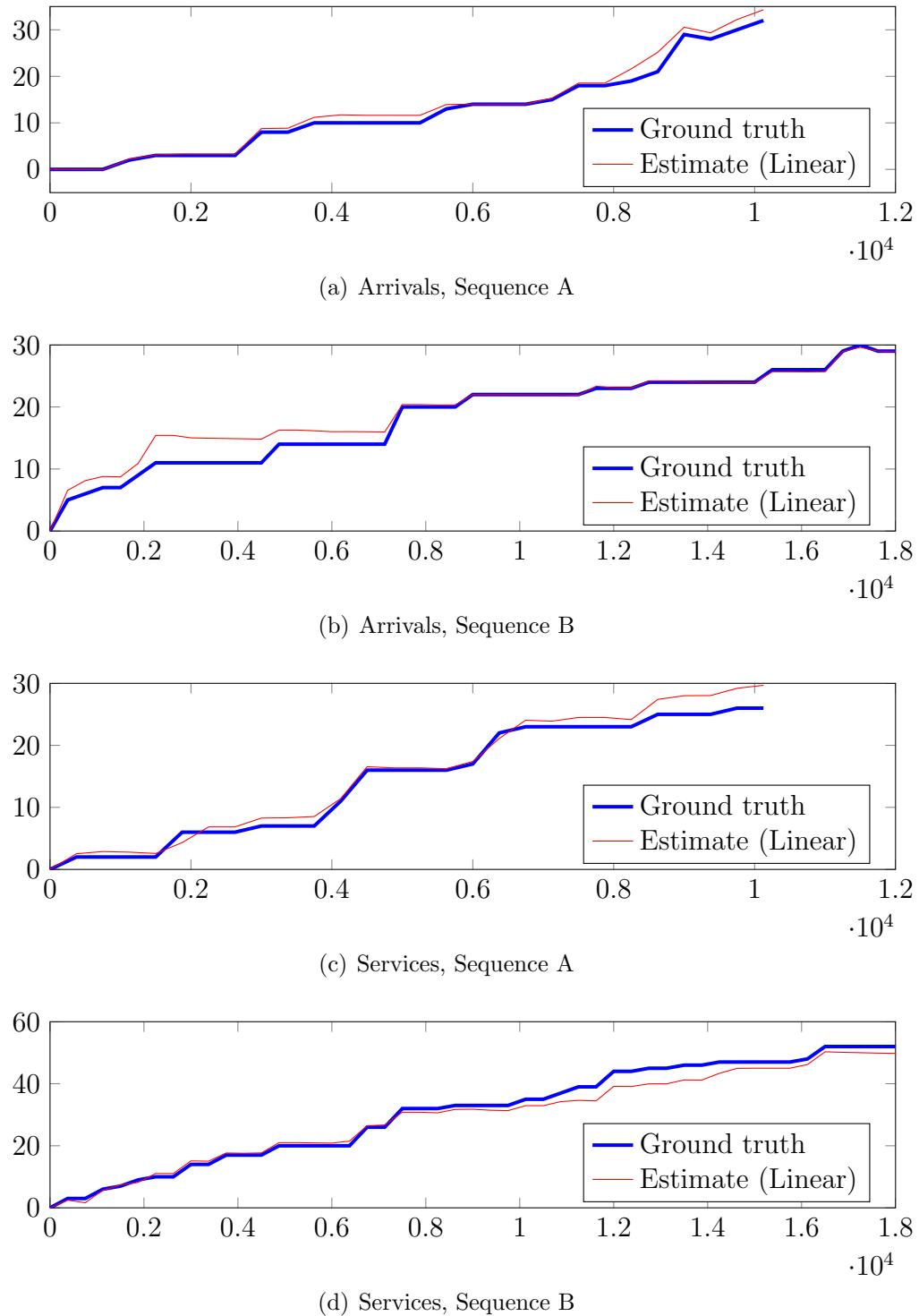


Figure 6.20: Cumulative performance of the virtual gate algorithm across Sequences A and B. Both arrivals and services are shown.

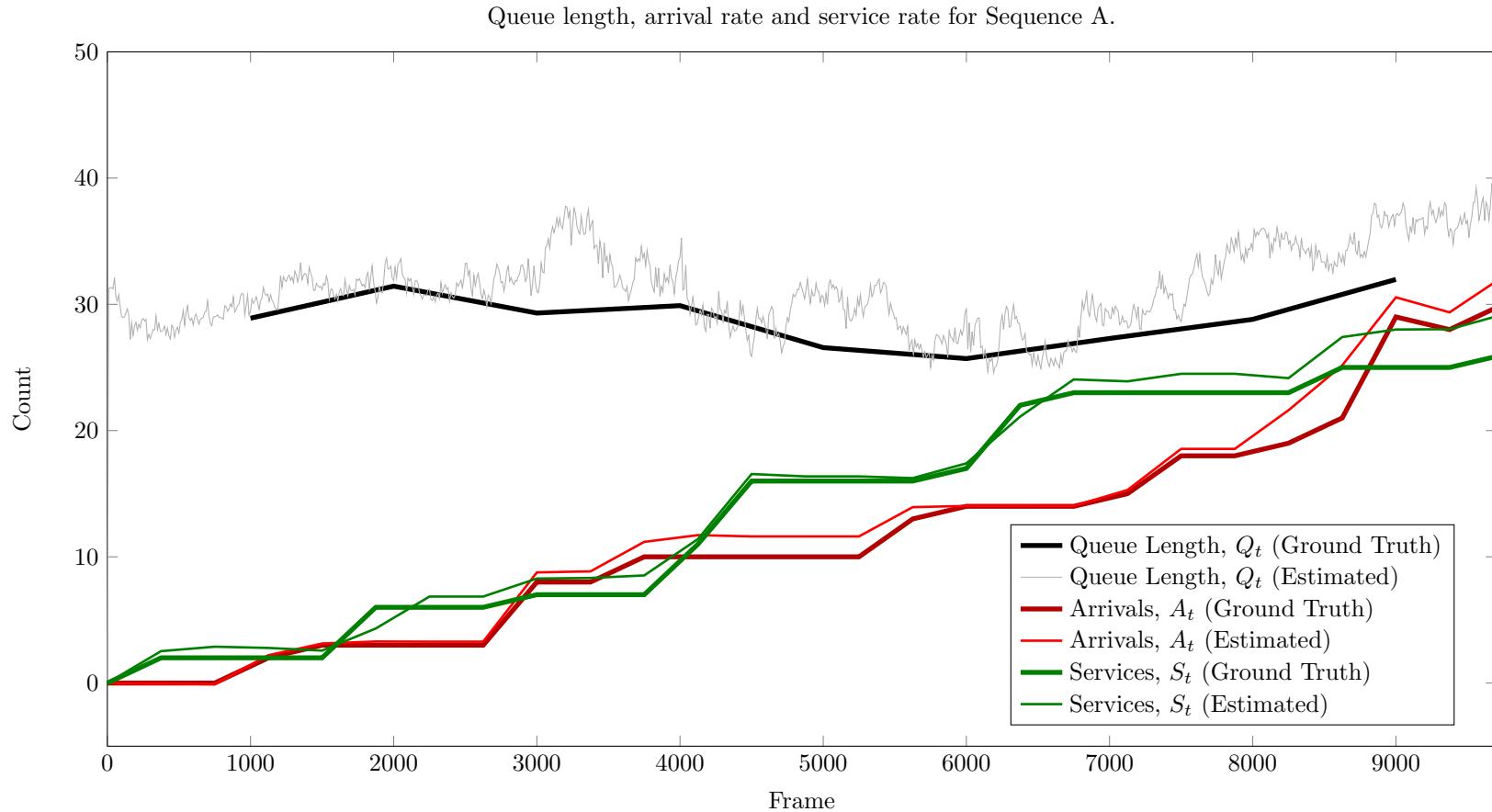


Figure 6.21: Results of the proposed queue monitoring algorithm operating on **Sequence A**. Queue length (Q_t) is estimated using the crowd counting module; arrivals (A_t) and services (S_t) are estimated using the virtual gate modules.

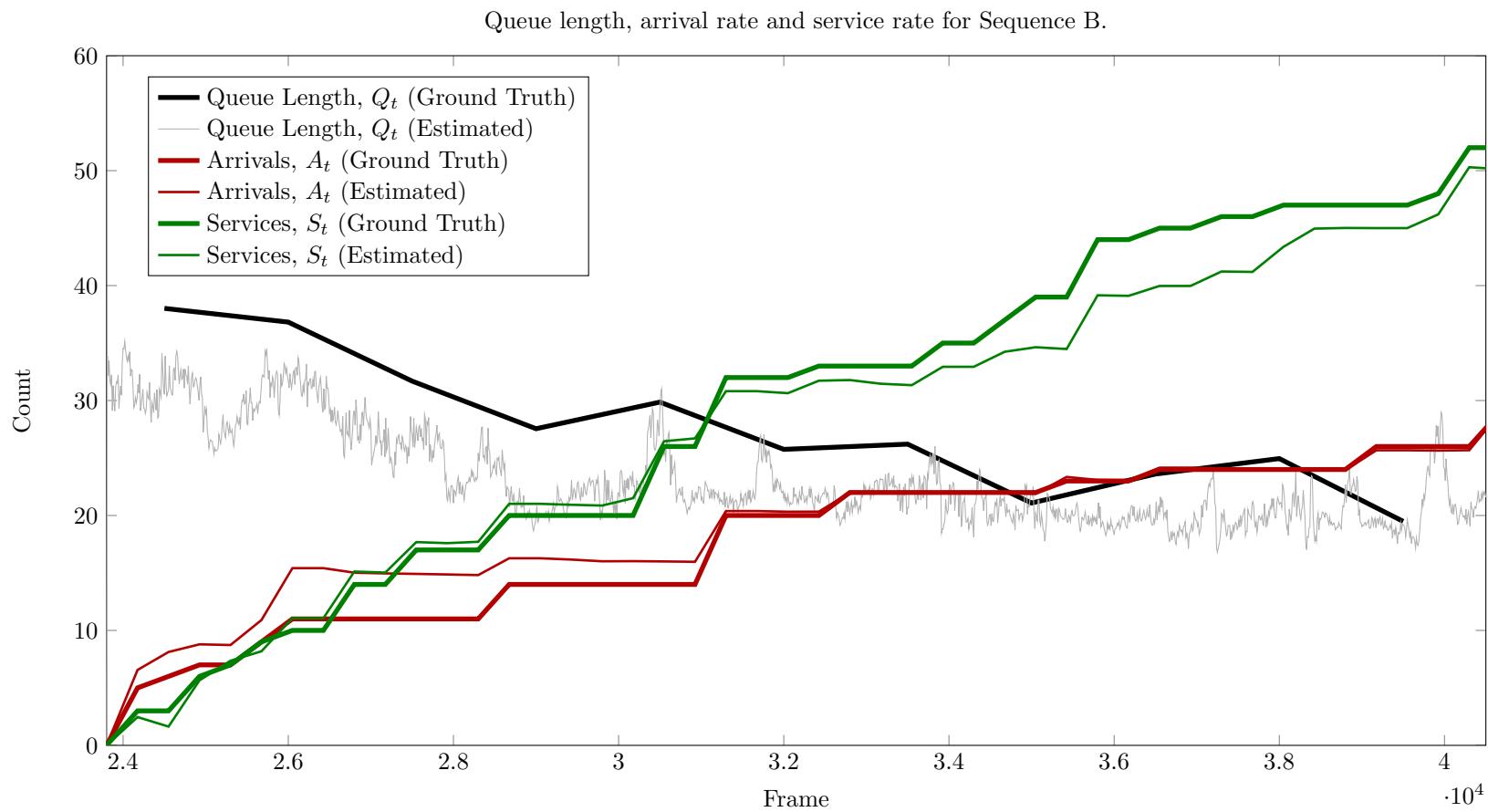


Figure 6.22: Results of the proposed queue monitoring algorithm operating on **Sequence B**. Queue length (Q_t) is estimated using the crowd counting module; arrivals (A_t) and services (S_t) are estimated using the virtual gate modules.

These experiments demonstrate that the proposed crowd counting system and virtual gate algorithm can be used to accurately monitor queue properties in a crowded environment. Some errors are observed in the crowd counting module, due primarily to the difficulty of counting crowds with large items of baggage. Because of the camera distance and the size of the crowd, prior approaches to abandoned luggage detection via object tracking or foreground analysis [176] are unsuitable in this environment. Instead we rely on the extracted features to provide an approximation of crowding levels: since the mean error is 12.15%, it is sufficiently accurate to correctly notify an operator of an excessively large queue or an abnormally empty scene.

Errors are also observed in the virtual gate module, due to variations in the amount of baggage carried by each passenger, occlusions between people entering the queue, and the distance of the camera from the scene. These problems could be mitigated by using a second camera positioned closer to the scene, specifically dedicated to queue monitoring rather than general surveillance. An overhead camera would be ideal for this analysis as it removes occlusions. These approaches are easily integrated with the framework proposed in this chapter.

6.5 Summary

This chapter has proposed a novel queue monitoring algorithm by combining the existing technologies of crowd counting and virtual gates. These modules are combined to calculate queue statistics such as arrival rate, service rate and queue size, as described in Section 6.2.

The following contributions are noted:

- The novel combination of crowd counting and virtual gate technologies to monitor queue statistics such as arrival rate, service rate, queue size and growth rate.
- A queue monitoring framework which establishes the relationship between queue statistics and allows these statistics to be derived from others in the absence of adequate video coverage. Derived statistics such as growth rate and wait time are also included in this framework.
- A virtual gate algorithm which makes use of feature points passing through a region of interest. The accumulation of optical flow at these points and the introduction of optical flow histograms are both demonstrated to improve accuracy.
- An evaluation on real world queue data. Unlike existing crowd datasets which are captured in unordered public spaces such as shopping malls and outdoor walkways, the datasets used in this analysis were captured from six cameras at Brisbane International Airport. The queue analysis was performed on surveillance footage obtained from the baggage check-in counter.

The evaluation of this system indicates that the proposed algorithm can accurately monitor queue statistics over time with a high level of accuracy. This technology has many applications for business operators, such as airports and supermarkets, in which queue management plays an important role.

Chapter 7

Anomaly Detection

7.1 Introduction

Automated visual surveillance in public environments is a rapidly growing area of research. As many CCTV cameras are not actively monitored, it is desirable to automatically detect abnormal events for security or review purposes. In ordered and public environments such as airports, shops, train stations and other public spaces, abnormal events are of interest to security personnel as they may be indicative of dangerous or disruptive events.

Anomalous events may also have a negative impact on the operation of the crowd monitoring algorithms described in Chapters 3-6. These crowd monitoring algorithms are designed to monitor human crowds under relatively normal circumstances, however when these assumptions are violated the operation of the system may be compromised. This might occur, for example, if a bicycle or vehicle enters a scene designed for pedestrians only.

This chapter proposes an *anomaly detection* algorithm designed to detect violations of these assumptions, as well as unusual situations which may be of interest to security personnel. The proposed algorithm is based on a novel representation of local motion patterns, called *textures of optical flow*.

In ordered and public environments such as airports, shops, train stations and public spaces, the abnormal events which we typically seek to detect include:

1. Pedestrians moving with excessive speed, for example running or skateboarding (Figure 7.1(a)).
2. Spatial abnormalities, such as intruders in restricted areas or unusual locations (Figure 7.1(b))
3. Non-human objects, e.g. vehicles or bikes (Figure 7.1(c)).

Current approaches to abnormality detection in uncrowded scenes usually involve object tracking followed by trajectory analysis [91, 138, 179], while in crowded scenes research has focused more generally on motion analysis at a local level [106, 125] or holistic level [11, 192].

In this chapter, the focus is placed on motion representation for the purpose of localised abnormality detection, rather than global detection schemes such as LDA [133, 192] or MRF [106]. These approaches may improve a system's performance, but also tend to "mask the limitations of the underlying visual representation" [125].

Existing approaches to anomaly detection using motion analysis generally require the computation of dense optical flow at full resolution, which is difficult to perform in real time with substantial accuracy. They also do not capture the motion *patterns* present in a scene, but instead reduce the optical flow in some way

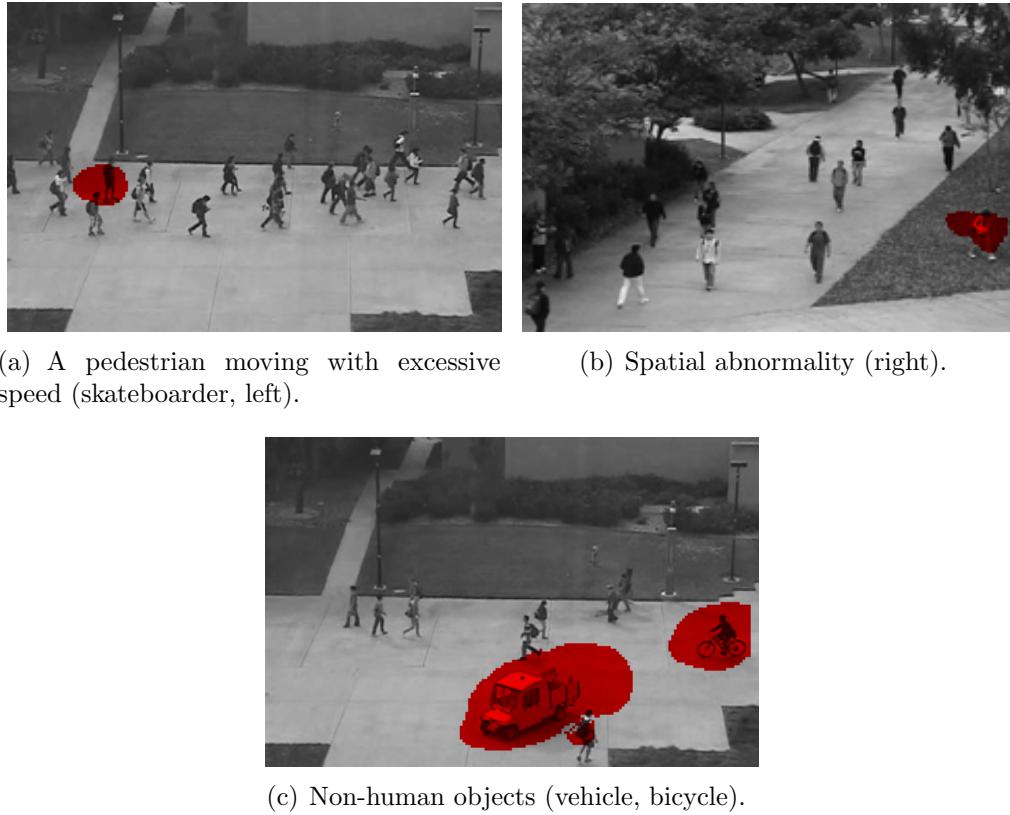


Figure 7.1: Typical abnormalities in crowded pedestrian scenes.

(through quantisation, dimensionality reduction, or histogram binning). This type of approach does not necessarily capture *interesting* interactions from a surveillance perspective, and novelty detection is performed as a subsequent process on the reduced data. However, if a visual representation is insufficient, an abnormal event will not be detected at a later stage of the algorithm, regardless of the statistical model used.

In this chapter, a novel visual representation called textures of optical flow is proposed, for the detection of anomalous objects and events in crowded scenes. The proposed technique is based on robust dense optical flow extracted from a subsampled image sequence. Textural features are then applied to these vector fields. The proposed technique is based on the extension of traditional greyscale textural features to robust dense optical flow fields.

While humans in surveillance footage exhibit a combination of various flow vectors due to the independent movement of their limbs, abnormal objects tend to generate consistently anomalous flow patterns across their entire surface, which may be detected using textures of optical flow.

Traditionally, textures have been applied only to discrete image data rather than to real-valued optical flow vector fields. This chapter describes a novel class of textural features applicable to such fields for the purpose of anomaly detection in crowded scenes. A statistical model of normal motion patterns is built so that anomalies are detected as outliers from this model. The approach is based on 3D volumes called spatio-temporal patches, and anomalies are detected as volumes of insufficient likelihood.

The remainder of this chapter is structured as follows: Section 7.2 describes the proposed visual representation called textures of optical flow (ToF); Section 7.3 outlines the proposed anomaly detection algorithm which utilises ToF; Section 7.4 presents experimental results on a publicly-available dataset; and Section 7.5 presents the conclusions from this research.

7.2 Textures of Optical Flow

Existing approaches to anomaly detection have used gradients or optical flow to detect abnormal events in crowded scenes. However, these approaches are generally best suited to detecting regions of unusually high velocity (i.e. running, a vehicle in a pedestrian area), as substantial information is lost through quantisation, dimensionality reduction, or histogram binning. In many cases, however, the objects to be detected do not necessarily travel at a substantially faster speed than regular pedestrians. For example, a skateboarder, cyclist or a slow-moving

vehicle should be detected regardless of their velocity.

Even an average pedestrian generates motion of a high velocity, due to the periodic movement of their limbs, although this is restricted to small regions near their extremities at any one time. By contrast, abnormal objects often generate these anomalous flow patterns across their entire surface. For example, the flow field of a cyclist or vehicle passing through a pedestrian scene is relatively smooth and laminar.

In determining a robust set of features to detect such flow patterns, we draw on the well known textural statistics of Haralick [79] which measure image properties such as homogeneity and contrast; and devise a similar set of features applicable to optical flow fields. The classic textural features are based on the grey level co-occurrence matrix (GLCM), which measures the quantity of cooccurring pixel values in a discrete-valued greyscale image I , at a specified offset $\boldsymbol{\delta} = (\delta_x, \delta_y)$.

For generality let us consider the textural features of 3D *volumes*, rather than 2D images, in order to detect abnormalities. Therefore we extend the standard GLCM from two dimensions to an arbitrary number of dimensions. The GLCM for 3D data was recently considered by Tsai [187] for a hyperspectral remote imaging applicaton. More generally, if we let \mathbf{p} denote a discrete point in n -dimensional space (pixel, voxel, etc.), and $\boldsymbol{\delta}$ denote an offset such that $\mathbf{p}' = \mathbf{p} + \boldsymbol{\delta}$, the values of the GLCM are calculated by [79]:

$$G(x, y) = \sum_{\mathbf{p} \in I} \begin{cases} 1 & \text{if } I(\mathbf{p}) = x \text{ and } I(\mathbf{p}') = y \\ 0 & \text{otherwise} \end{cases} \quad (7.1)$$

where I is an n -dimensional volume containing discrete-valued greyscale data. The normalised GLCM, analogous to a pdf, measures the frequency of such co-

occurrences:

$$P(x, y) = \frac{1}{K} G(x, y) \quad (7.2)$$

where K is a normalisation constant:

$$K = \sum_{x,y} G(x, y) \quad (7.3)$$

Haralick defines a number of textural statistics based on this distribution, of which homogeneity, f_H , and contrast, f_C , are most relevant here because they pertain to the smoothness (or roughness) of a field:

$$f_C = \frac{1}{K} \sum_{x,y} (x - y)^2 G(x, y) \quad (7.4)$$

$$f_H = \frac{1}{K} \sum_{x,y} \frac{1}{1 + (x - y)^2} G(x, y) \quad (7.5)$$

As $G(x, y)$ is defined for discrete-valued fields only (see Equation 7.1), these features cannot be applied directly to a continuous optical flow field. By substituting Equation 7.1 into Equations 7.4-7.5, they may be rewritten:

$$f_C = \frac{1}{K} \sum_{x,y} (x - y)^2 \sum_{\mathbf{p} \in I} \begin{cases} 1 & \text{if } I(\mathbf{p}) = x \text{ and } I(\mathbf{p}') = y \\ 0 & \text{otherwise} \end{cases} \quad (7.6)$$

$$= \frac{1}{K} \sum_{\mathbf{p} \in I} \sum_{x,y} \begin{cases} (x - y)^2 & \text{if } I(\mathbf{p}) = x \text{ and } I(\mathbf{p}') = y \\ 0 & \text{otherwise} \end{cases} \quad (7.7)$$

$$= \frac{1}{K} \sum_{\mathbf{p} \in I} [I(\mathbf{p}) - I(\mathbf{p}')]^2 \quad (7.8)$$

And, similarly:

$$f_H = \frac{1}{K} \sum_{\mathbf{p} \in I} \frac{1}{1 + [I(\mathbf{p}) - I(\mathbf{p}')]^2} \quad (7.9)$$

Both Equations 7.8 and 7.9 are the summation of some quantity of interest: a difference measure in the case of contrast, and a similarity measure in the case of homogeneity. More generally, these belong to a class of features described by:

$$f = \sum_{\mathbf{p} \in I} \rho(I(\mathbf{p}), I(\mathbf{p}')) \quad (7.10)$$

where ρ is the quantity of interest, and is expressed as a function of $I(\mathbf{p})$ and $I(\mathbf{p}')$. In this form these equations are applicable to continuous fields.

When considering optical flow fields in real-world scenes, low values of contrast (f_C) and high values of homogeneity (f_H) are the normal state of being when zero motion is present across a region. But these properties are also generated by abnormal objects moving smoothly, particularly wheeled objects. Therefore a feature which captures both the smoothness of the flow *and* the presence of motion is most desirable for a surveillance application. As we are dealing with a vector field \mathbf{v} , we propose a quantity of interest ρ that captures smoothness in terms of both magnitude and direction. A natural measure of similarity between two vectors is the dot product,

$$\rho(\mathbf{v}(\mathbf{p}), \mathbf{v}(\mathbf{p}')) = \mathbf{v}(\mathbf{p}) \cdot \mathbf{v}(\mathbf{p}') \quad (7.11)$$

$$= |\mathbf{v}(\mathbf{p})| |\mathbf{v}(\mathbf{p}')| \cos \theta \quad (7.12)$$

as this incorporates both the magnitude and the angle, θ , between the vectors.

Therefore we define *uniformity* across a real vector field \mathbf{v} :

$$\phi_{\delta} = \sum_{\mathbf{p} \in \mathbf{v}} \rho(\mathbf{v}(\mathbf{p}), \mathbf{v}(\mathbf{p}')) \quad (7.13)$$

$$= \sum_{\mathbf{p} \in \mathbf{v}} \mathbf{v}(\mathbf{p}) \cdot \mathbf{v}(\mathbf{p} + \boldsymbol{\delta}) \quad (7.14)$$

$$= \sum_{\mathbf{p} \in \mathbf{v}} u(\mathbf{p})u(\mathbf{p} + \boldsymbol{\delta}) + v(\mathbf{p})v(\mathbf{p} + \boldsymbol{\delta}) \quad (7.15)$$

where u and v represent the horizontal and vertical components of \mathbf{v} , respectively.

Although it is common in normal pedestrian scenes for neighboring pixels to have similar values of optical flow, this is not necessarily true for pixels at a larger fixed relative distance from one another due to the non-rigid movement of the human body. Therefore $\boldsymbol{\delta}$ can be set to any given offset, and need not be restricted to small or unit values (as is commonly used; see, for example, Marana [130]).

7.3 Anomaly Detection Algorithm

The anomaly detection algorithm proposed in this chapter makes use of textures of optical flow (ToF), as described in Section 7.2. An overview of the algorithm is depicted in Figure 7.2. Robust optical flow forms the basis of the approach, and this is described in Section 7.3.1. A video sequence is then divided into 3D volumes, or spatio temporal patches, which are described in Section 7.3.2. From each volume, local features are extracted based on motion information, spatial information and textures of optical flow, using the procedure outlined in Section 7.3.3. A statistical model of normal behaviour patterns is used to jointly model this data, so that anomalies are detected as statistical outliers. The classification stage is detailed in Section 7.3.4.

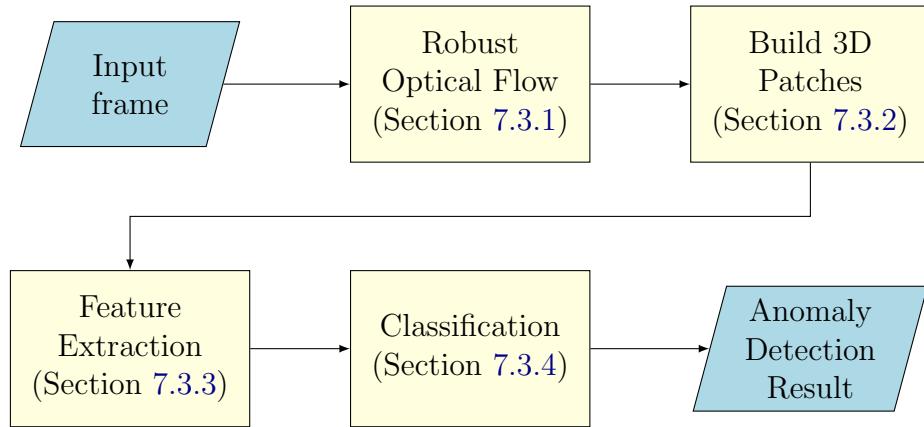


Figure 7.2: Overview of the anomaly detection system.

7.3.1 Robust Optical Flow

The calculation of uniformity (Section 7.2) is based on spatially-related optical flow vectors. In order to achieve a meaningful result, these vectors need to be sufficiently accurate. Optical flow estimation is a noisy process and traditional methods such as Horn and Schunck [86] and Lucas and Kanade [117] are unlikely to compute the flow with sufficient robustness for our purpose.

Various robust flow methods [180] have been proposed that produce more accurate optical flow fields. However, without GPU acceleration these state-of-the-art algorithms are generally not capable of real-time performance, typically taking several seconds or minutes per frame (Section 2.2.1).

A popular method for robust estimation of optical flow, proposed by Black and Anandan [22], has been used previously in crowd monitoring research [11]. We adopt this algorithm here, as it provides sufficient accuracy on our datasets. However, another algorithm could also be used in its place with this system as the technology continues to improve.

Black and Anandan's optical flow algorithm does not fulfill real-time requirements

for larger images, but full resolution is not required for the detection of abnormal flow patterns. This is because such anomalies are usually generated at the object level (humans, bikes, cars) and not at the pixel-level. Such objects are still visible in smaller images, therefore each image is downsampled to a lower resolution prior to processing.

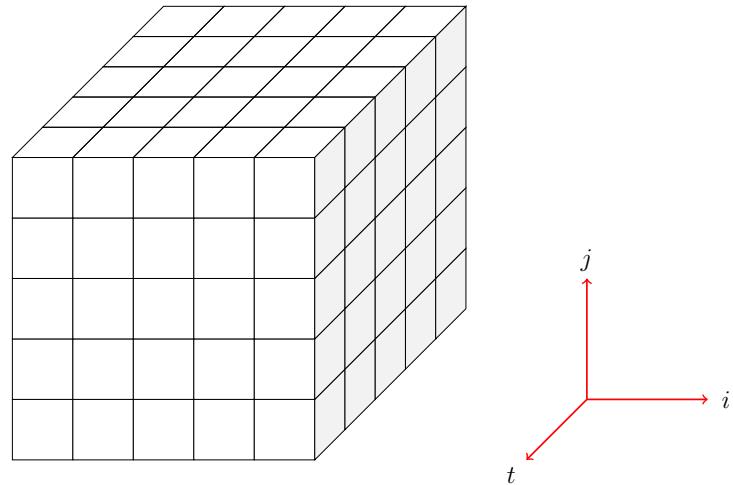
For the experiments presented in this chapter all images are standardised to 180×120 pixels. This approach places higher value on an accurate flow field at low resolution than it does on a noisier approximation at high resolution. This is an appropriate methodology as the proposed algorithm requires accurate flow for the calculation of uniformity.

7.3.2 Spatio-Temporal Patches

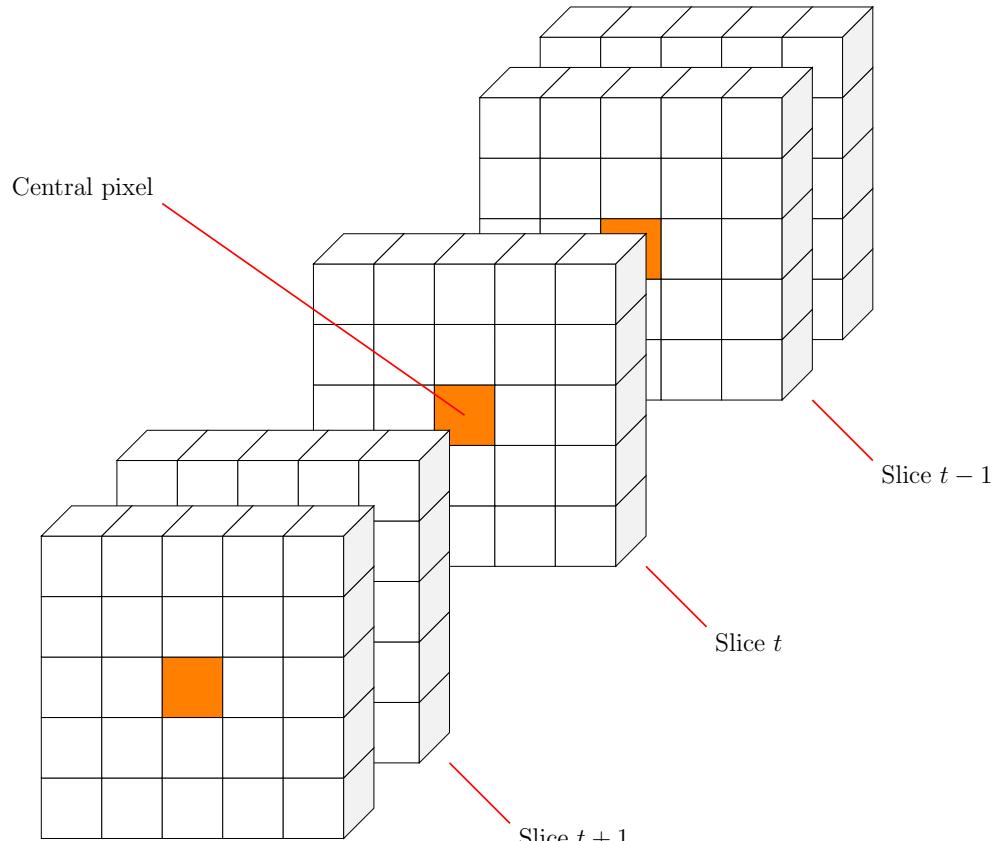
The normal frames in a training dataset are used to train a model of normal behaviours. This is done in local regions of the image using low-level features extracted from 3D volumes, or ‘spatio-temporal patches’. The features are used to classify the patch as either normal or abnormal.

Two types of classification models are considered in this chapter: the Gaussian Mixture Model (GMM) which is used to evaluate the spatio-temporal patch as a single unit, and the Hidden Markov Model (HMM) which is used to evaluate the patch as a temporal sequence of ‘slices’, as depicted in Figure 7.3. Correspondingly, two types of spatio temporal patches are considered in this research:

1. A **cuboid**, or rectangular prism, as depicted in Figure 7.3(a). This patch is treated as a single unit with one feature vector describing the whole volume. This vector is used for **GMM** classification.



(a) Rectangular prism.



(b) Visualised as a set of slices.

Figure 7.3: Spatio temporal patches may be visualised as a rectangular prism for GMM classification, or as a set of connected slices for HMM classification.

2. A **temporal sequence of slices** as depicted in Figure 7.3(b). A central pixel is tracked using optical flow as it moves through the scene, and a rectangular slice is constructed around this pixel in each frame. A feature vector is extracted from each temporal slice, so that the patch is described by a sequence of feature vectors. This sequence is used for **HMM** classification.

For GMM classification, the spatio-temporal patch P is constructed around a central pixel (i, j, t) , forming a *rectangular prism*. Low level statistics are extracted across the patch to populate a single feature vector \mathbf{f} . This feature vector is then used to determine the likelihood of the patch using a statistical model (GMM), and to subsequently classify the pixel at the center of the patch as either normal or abnormal.

For HMM classification, the spatio temporal patch is visualised as a temporal sequence of ‘slices’. A feature vector is extracted from each slice. The observation sequence is then classified as a normal or abnormal sequence using the Hidden Markov Model.

This procedure generates an *abnormality mask*, in which every pixel is marked as either normal or abnormal based on its surrounding patch, to which further analysis may be applied. For example, binary morphology and connected component labelling may be used to localise anomalous objects, or to initialise object tracking of anomalous objects.

The size of the spatio-temporal patch is determined by the *pedestrian template* introduced in Section 3.2.2 (p. 106). The pedestrian template is a rectangular model which approximates the size of person at each location in an image. The dimensions of the rectangle, $w \times h$, vary as a function of the vertical image coordinate, j , around which it is constructed. This is described in Equations 3.13-3.14 (p. 111), and repeated here for reference:

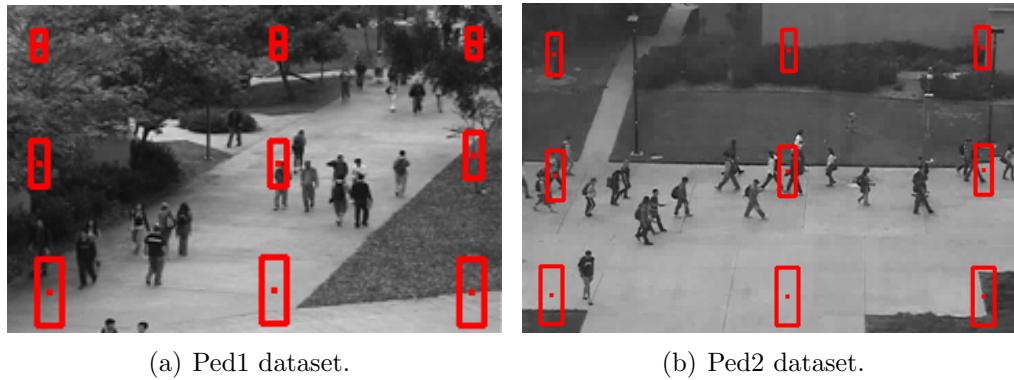


Figure 7.4: Patches of variable size are used based on the size of the pedestrian template centred at each location in the scene. Various patch sizes are shown on each image.

$$w = b_w j + c_w \quad (7.16)$$

$$h = b_h j + c_h \quad (7.17)$$

where $\{b_w, c_w, b_h, c_h\}$ represent the linear coefficients. Figure 7.4 depicts the variable patch sizes for two benchmark datasets, Ped1 and Ped2.

Therefore at any location within the scene, the width and height of the spatio temporal patch is equal to the pedestrian template width w and height h . The ‘depth’ of the patch is set to 21 frames (which corresponds to 2.1 seconds at 10 fps) as this is sufficient to capture the motion patterns present in a scene. For the cuboid model, this is the length or depth of the rectangular prism; whereas for the temporal sequence, this is the number of slices used.

7.3.3 Feature Extraction

A spatio temporal patch is centred around every pixel in a video sequence, from which a number of features are extracted. The feature extraction procedure used

for cuboids is detailed in Section 7.3.3.1, and the procedure for temporal sequences is described in Section 7.3.3.2. Section 7.3.3.3 describes an efficient method for extracting these features which makes use of integral images.

7.3.3.1 Feature Extraction for Cuboids

From each patch P a feature vector \mathbf{f} is extracted, containing spatial coordinates, motion information and textures of optical flow (Section 7.2).

Spatial abnormalities occur when a motion pattern is observed in an unusual region of the scene. A typical approach is to train a model for each location in the image [3, 8, 106], although this requires sufficient training data for every location. It is unlikely that all of the normal possible behaviours will be observed in all locations during training, leaving this approach susceptible to improper generalisation.

Alternatively, a global model assumes that normal behaviour has a constant definition across the whole scene, effectively neglecting spatial information [11]. This is inadequate because the patterns of motion expected to occur in one region cannot necessarily be expected to occur in another.

In order to address these problems, we model the motion, location and textual information using a joint distribution. Location information is encoded by including the i and j coordinates of the patch center in the feature vector.

Velocity information across a patch, P , is directly incorporated using a summation of optical flow in each of the horizontal and vertical directions:

$$s_u = \sum_{\mathbf{p} \in P} u(\mathbf{p}) \quad (7.18)$$

$$s_v = \sum_{\mathbf{p} \in P} v(\mathbf{p}) \quad (7.19)$$

This motion feature, in conjunction with spatial information (patch coordinates), serve to model the expected velocities in each region of the scene.

Lastly we consider the uniformity ϕ_δ as defined in Section 7.2, across the patch P . The offset parameter δ dictates the relative offset when computing the dot product in Equation 7.14. Varying the offset parameter provides a powerful set of descriptors of motion uniformity at various scales. Using multiple values within the same feature vector achieves a ‘multi-scale analysis’ whereby uniformity is considered using both small and large offsets at the same time. This is useful for detecting objects of various sizes. For example, a skateboarder who is relatively small will have a high uniformity at fine scales (1-2 pixels in any direction); while the optical flow of a large vehicle will exhibit a correlation at both small and large offsets.

In order to achieve this multi-scale analysis, a combination of multiple offsets are used. The offsets are expressed as a fraction of the pedestrian template (a rectangle of size $w \times h$, whose dimensions are a linear function of the vertical image coordinate j). A number of example feature vectors are listed in Table 7.1 with diagrams depicting the uniformity offsets.

The offset is only applied in the i and j directions (horizontal and vertical). A temporal offset is not used because a moving object changes its position over time, therefore uniformity in this dimension is of little interest.

Finally, the uniformity feature and velocity features are normalised by dividing

by the area of the patch, wh . This achieves normalisation at any location in the scene so that the features are not patch-size-dependent.

7.3.3.2 Feature Extraction for Temporal Sequences

A temporal sequence of slices is depicted in Figure 7.3(b). The central pixel is tracked using optical flow as it moves through a scene, and a feature vector is extracted from the rectangular ‘slice’ surrounding this pixel in every frame¹.

The location of the central pixel, (i, j) , is used as the location feature. This may change from frame to frame, depending on the magnitude of the optical flow at this location in the image. A rapidly moving object will see substantial change in position, for example.

The velocity feature is calculated by applying Equations 7.18-7.19 to each slice, S :

$$s_u = \sum_{\mathbf{p} \in S} u(\mathbf{p}) \quad (7.20)$$

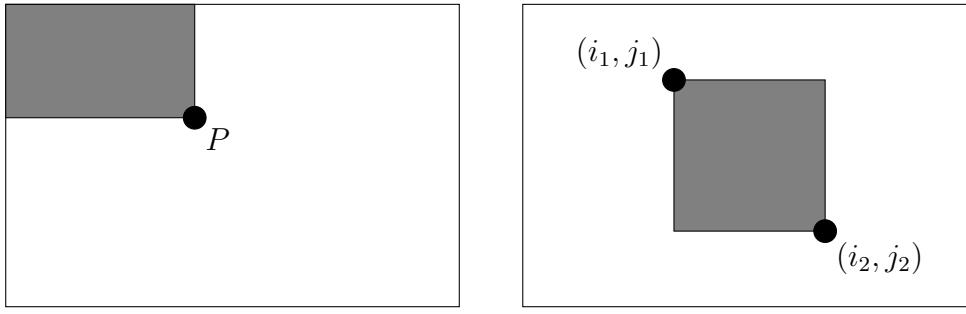
$$s_v = \sum_{\mathbf{p} \in S} v(\mathbf{p}) \quad (7.21)$$

Finally the uniformity feature is calculated for each slice using the offsets listed in Table 7.1. The feature is divided by the area of the slice, wh , to achieve normalisation across the scene. An efficient method for calculating patch based features is described in the following section.

¹The pixel is tracked by displacing it by the value of the optical flow from one frame to the next.

Location	Motion	Uniformity Offsets	Uniformity Offset Diagram
i, j	s_u, s_v	$(\frac{w}{8}, 0), (0, \frac{h}{8})$	
i, j	s_u, s_v	$(\frac{w}{4}, 0), (0, \frac{h}{4})$	
i, j	s_u, s_v	$(\frac{w}{2}, 0), (0, \frac{h}{2})$	
i, j	s_u, s_v	$(\frac{w}{8}, 0), (\frac{w}{4}, 0), (\frac{w}{2}, 0)$	
i, j	s_u, s_v	$(\frac{w}{8}, \frac{h}{8}), (\frac{w}{4}, \frac{h}{4}), (\frac{w}{2}, \frac{h}{2})$	
i, j	s_u, s_v	$(0, \frac{h}{8}), (0, \frac{h}{4}), (0, \frac{h}{2})$	
i, j	s_u, s_v	$(-\frac{w}{8}, \frac{h}{8}), (-\frac{w}{4}, \frac{h}{4}), (-\frac{w}{2}, \frac{h}{2})$	

Table 7.1: Feature vectors used with the proposed system. The notation $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$ represent unit vectors in the i and j dimensions respectively. The width and height of the pedestrian template (w, h) vary linearly as a function of j .



(a) The integral image at point P is the sum of all elements above and to the left of that point (inclusive).

(b) Summing features across a patch can be performed efficiently using the integral image.

Figure 7.5: The integral image can be used to efficiently extract features from spatio temporal patches.

7.3.3.3 Efficient Implementation of Feature Extraction

Exhaustively extracting features from spatio temporal patches (centred around every pixel in a video sequence) is inefficient with iterative methods. In order to speed computation we can make use of summed area tables (SAT) or *integral images*, as popularised by Viola and Jones [191] for face detection using simple rectangular features.

Consider, for example, the uniformity feature ϕ_{δ} with offset parameter $\delta = (\delta_i, \delta_j)$. The first step is to calculate a dot product image D :

$$D(i, j) = \mathbf{v}(i, j) \cdot \mathbf{v}(i + \delta_i, j + \delta_j) \quad (7.22)$$

The uniformity across a slice S which spans from (i_1, j_1) to (i_2, j_2) , as depicted in Figure 7.5(b), can be calculated exhaustively using Equation 7.14:

$$\phi_{\delta} = \sum_{(i,j) \in S} \mathbf{v}(i, j) \cdot \mathbf{v}(i + \delta_i, j + \delta_j) \quad (7.23)$$

$$= \sum_{(i,j) \in S} D(i,j) \quad (7.24)$$

$$= \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} D(i,j) \quad (7.25)$$

However this iteration is costly to implement across all spatio temporal patches in a video sequence. Therefore we make use of the integral image of D , which is denoted I_D . Each pixel in I_D is the summation of pixels above and to the left of that pixel in image D (inclusive), as depicted in Figure 7.5(a). The integral image is defined by:

$$I_D(i,j) = \sum_{i'=0}^i \sum_{j'=0}^j D(i',j') \quad (7.26)$$

Which can be efficiently calculated in a single pass:

$$I_D(i,j) = D(i,j) + I_D(i-1,j) + I_D(i,j-1) - I_D(i-1,j-1) \quad (7.27)$$

The summation across a patch spanning from (i_1, j_1) to (i_2, j_2) , as shown in Figure 7.5(b), can be easily computed as:

$$\phi_{\delta} = \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} D(i,j) \quad (7.28)$$

$$= I_D(i_2, j_2) - I_D(i_1-1, j_2) - I_D(i_2, j_1-1) + I_D(i_1-1, j_1-1) \quad (7.29)$$

This avoids the costly procedure of iterating across all pixels between (i_1, j_1) and (j_1, j_2) . It is straightforward to extend this approach from a 2D slice to a 3D cuboid (since the cuboid is simply a set of connected slices; the summation is simply extended into the third dimension).

7.3.4 Classification

A statistical model of normal behaviours is learned from a large database of regular surveillance footage under normal circumstances. The features from each patch in the training dataset are used to train the system. Two models are considered in this work:

1. A **Gaussian Mixture Model** (GMM) is used to classify a **cuboid** as a single entity.
2. A **Hidden Markov Model** (HMM) is used to classify a **temporal sequence of slices**.

Anomalies are detected as statistical outliers using a fixed threshold, L_T . If the likelihood of any patch falls below this threshold then the region is classified as abnormal. The frame is labelled abnormal on a holistic level if one or more patches in the frame is abnormal.

Section 7.3.4.1 discusses the Gaussian Mixture Model, and Section 7.3.4.2 discusses the Hidden Markov Model. Section 7.3.4.3 describes some methods for making the classification procedure more efficient during both training and testing.

7.3.4.1 Gaussian Mixture Model

The Gaussian mixture model (GMM) is a probabilistic model obtained from a mixture of Gaussian distributions. The GMM has the density function:

$$p(\mathbf{x}) = \sum_{k=1}^K \alpha_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \Sigma_k) \quad (7.30)$$

where α_k , $\boldsymbol{\mu}_k$ and Σ_k denote the mixture weight, mean and covariance of component k respectively.

In some applications a diagonal covariance matrix is sometimes used for simplicity or efficiency. A full covariance matrix is used here to properly model the relationship between the features. This is because the feature vector contains a variety of unrelated information (such as location and motion) which we wish to jointly model as accurately as possible.

The training data set consists of N data samples, $\{\mathbf{x}_i\}_{i=1}^N$, which are used to build the model. The parameters of the GMM are learned using the expectation-maximisation (EM) algorithm [21]. This procedure is summarised in Section 2.7.1.

The procedure must be initialised with an approximate pre-clustering of the data as described in Section 7.3.4.3.

When testing the algorithm, the likelihood of each input is calculated (Equation 7.30) and compared to a threshold of acceptability L_T . When the likelihood falls below this threshold the patch is classified as abnormal.

The number of mixture components is determined automatically using the Bayesian Information Criterion (BIC), as described in Section 7.3.4.3.

7.3.4.2 Hidden Markov Model

A hidden Markov model (HMM) is a stochastic model in which a series of unobserved states, $Q = \{q_t\}_{t=1}^T$ ($q_t \in [1, N]$), are visited in a sequential order, for which a corresponding sequence of outputs $O = \{\mathbf{o}_t\}_{t=1}^T$ are observed [21, 152]. The observations are a probabilistic function of the hidden states. In the proposed system, a Gaussian mixture model output distribution is used:

$$p(\mathbf{o}_t | q_t = j) = \sum_{\ell=1}^M c_{j\ell} \mathcal{N}(\mathbf{o}_t | \boldsymbol{\mu}_{j\ell}, \boldsymbol{\Sigma}_{j\ell}) \quad (7.31)$$

where $c_{j\ell}$, $\boldsymbol{\mu}_{j\ell}$, $\boldsymbol{\Sigma}_{j\ell}$ denote the mixture coefficient, mean and covariance of mixture ℓ in state j , respectively. The full set of HMM parameters is collectively denoted λ , and this includes the output distribution parameters $\{c_{j\ell}, \boldsymbol{\mu}_{j\ell}, \boldsymbol{\Sigma}_{j\ell}\}$, the initial state probabilities $\pi = \{\pi_i\}$ and the state transition probabilities $A = \{a_{i,j}\}$, which are described in Section 2.7.2.

In order to train the HMM, the parameters λ are chosen so as to maximise the likelihood of the observed sequences, $p(O|\lambda)$. This Baum-Welch procedure is described in Section 2.7.2 and additional details are provided in Appendix C.

When performing anomaly detection, the likelihood of each input sequence is calculated (Equations 2.83-2.85, p. 93) and compared to a threshold of acceptability L_T . When the likelihood falls below this threshold the patch is classified as abnormal.

In this research, $M = 1$ mixtures are used, and the number of states N is determined automatically using the Bayesian Information Criterion (BIC) as described in Section 7.3.4.3.

7.3.4.3 Efficient Implementation of Classification

During testing, spatio temporal patches are constructed around every pixel in the video sequence. To avoid redundancy during training, however, only non-overlapping patches are used. This reduces the number of data points that need to be modelled.

The mixture model is learned using the expectation-maximisation (EM) algorithm, as summarised in Section 7.3.4.1 for GMMs and 7.3.4.2 for HMMs. Some common problems with EM include:

1. A strong reliance on the initial clustering parameters;
2. Slow convergence of the algorithm; and
3. An ambiguity in the appropriate number of mixture components to use.

To address these issues, the following steps are taken:

1. K-Means++

The K-Means++ algorithm is used as an initial ‘hard clustering’ to seed the EM algorithm, as this achieves better speed and accuracy than regular K-Means [12].

(a) GMM:

In the case of the GMM, the feature vectors are assigned to clusters which represent the initial *mixture component* to which they belong.

(b) HMM:

In the case of the HMM, the feature vectors for each slice (observations) are assigned to clusters which represent the initial *state* to which they belong.

2. Foreground Masking

Faster convergence is achieved by only considering patches with sufficient motion to be of interest; thus a patch is only considered if it contains foreground pixels which are detected using an adaptive background model [209]. This is also done during testing for consistency and to mask out uninteresting regions.

(a) Cuboid:

If a cuboid contains a foreground pixel anywhere within the prism (Figure 7.3(a)) it is included in the training dataset; otherwise, it is discarded.

(b) Temporal Sequence:

If a temporal sequence of slices contains a foreground pixel at any of its *central pixels* (Figure 7.3(b)) then it is included in the training dataset; otherwise, it is discarded.

This substantially reduces the number of data samples and allows training to occur within a reasonable time frame.

3. Automatic Model Selection

The EM algorithm is run several times using an increasing number of mixture components (GMM) or states (HMM). The final number is selected automatically by using the Bayesian Information Criterion (BIC) [172].

The BIC is based on the log likelihood of the training data, with a penalty for the number of free parameters in the model (which increases as the number of states or mixtures increases). This penalty avoids overfitting to a training dataset at the expense of generality. The BIC is calculated as follows:

$$\text{BIC} = -\log L + k \log N \quad (7.32)$$

where L is the maximised likelihood, k is the number of free parameters in the model and N is the number of training data points. Whichever model configuration (i.e. number of states or mixtures) yields the lowest BIC is selected.

The above procedure automates the selection of training samples to reduce the size of the training dataset without sacrificing important information. This improves the efficiency of the algorithm. The above procedure also automates the selection of the model configuration (number of states or mixtures).

7.4 Experimental Results

The proposed algorithm is evaluated using two large publicly available benchmark datasets introduced by Mahadevan [125]. This section presents experimental results on these datasets and compares the proposed algorithm to existing approaches. Section 7.4.1 introduces the benchmark datasets and testing protocol used to evaluate the algorithm; Section 7.4.2 presents results for the proposed algorithm using various configurations; and Section 7.4.3 compares these results to other algorithms.

7.4.1 Benchmark Datasets and Testing Protocol

To assess the performance of the proposed algorithm it is evaluated on the UCSD anomaly detection datasets [125], ‘Ped1’ and ‘Ped2’. These datasets feature bidirectional pedestrian traffic from two different camera viewpoints. An image from dataset Ped1 is shown in Figure 7.4(a), and Ped2 is shown in Figure 7.4(b). The abnormalities include non-human objects, anomalous pedestrian motions, and

	Ped1	Ped2
Original Resolution	360×240	238×158
Down-sampled Resolution	180×120	180×120
Training Sequences	34	16
Testing Sequences	36	14

Table 7.2: The benchmark datasets used to evaluate the proposed anomaly detection algorithm. The total number of sequences is listed, and every frame has been annotated with ground truth (normal or abnormal).

spatial abnormalities (people moving off the main walkway, which does not occur during the training sequences).

Several short clips of 200 frames are used for training and testing the algorithm. In total there are 100 sequences and 30 minutes of footage. The frame rate of all sequences is 10 fps. Clips in the training set contain normal pedestrian activity, while the testing set has been annotated with frame-level ground truth (a binary flag indicating whether an anomaly is present). The threshold L_T is used to detect abnormal patches, and a frame is classified as abnormal if it contains any abnormal patches. The computer detection is compared to ground truth at each frame, and L_T is varied to generate an ROC curve; the equal-error rate (EER) and area under curve (AUC) are reported.

Table 7.2 provides a summary of the benchmark datasets.

7.4.2 Evaluation of Proposed Algorithm

The proposed algorithm is evaluated using the Ped1 and Ped2 datasets. Both types of spatio temporal patch are evaluated: the cuboid (with GMM classification) and temporal sequence (with HMM classification), as described in Section

7.3. The patch size is equal to the pedestrian template at any location in the image; the offsets used for calculating uniformity are expressed as a fraction of this template. Table 7.1 lists the full feature vectors used in testing the system.

The results of the proposed algorithm are presented in Table 7.3 for both the Ped1 and Ped2 datasets. The uniformity offsets are stated in each row, while the location and motion features are omitted for brevity (see Table 7.1 for the full feature vectors). The performance is also averaged across both datasets and this is reported under the heading ‘Average’.

On the Ped1 dataset, GMM classification achieves an EER of 18.9-23.0% while HMM classification achieves an EER of 18.1-21.6% (depending on the offsets used). These results indicate that performance is very similar for the two classifiers, with a small improvement seen with HMMs.

On the Ped2 dataset, GMM classification achieves an EER of 13.2-18.1% while the HMM achieves 16.4-21.6%. In this case, GMM classification outperforms the HMM noticeably.

In order to compare feature sets, results are averaged across both datasets and reported. The best two feature sets are indicated in bold for each column (Table 7.3). The most consistent performance is observed when the uniformity offsets $(\frac{w}{4}, 0), (0, \frac{h}{4})$ are used. This feature set does not provide the best performance in any one experiment, but it does provide the most consistent results. Using this feature set, the ROC curve for each dataset is depicted in Figures 7.6(a) and 7.6(b).

Although some individual frames were not detected correctly, most abnormal events were identified, with the erroneous frames occurring at the start and end of each event. This is depicted in Figure 7.7. These frames are somewhat ambiguous

because it is not always clear when an event starts and finishes. In practice it is more important that the event is correctly detected than it is for a precise timestamp to be identified. The longest error occurs during sequence 10, when a very slow-moving bicycle is not initially detected. However, the bicycle is eventually detected at the end of the sequence (Figure 7.8).

Representative frames from the system are shown in Figure 7.1. Anomalous pixels are indicated in red when the pixel's surrounding patch is detected to be abnormal.

To provide further insight into the contribution of each feature type, these experiments were repeated using *reduced* feature sets in which a single type of feature was omitted for each experiment. Table 7.4 presents these results. The uniformity offsets $(\frac{w}{4}, 0), (0, \frac{h}{4})$ were selected with a GMM classifier, and each feature type (location, motion, uniformity) was omitted in turn. This is indicated by the table's empty cells.

For comparison, the full feature vector is shown in the final row of Table 7.4 and the numbers are duplicated from Table 7.3. On the Ped1 dataset, these results indicate that the omission of any feature degrades the EER of the system by 3.1-6.0%. It is concluded that all features (location, motion and uniformity) contribute to the performance of the algorithm on this dataset.

On the Ped2 dataset, the performance of the system is significantly degraded when either motion or uniformity are omitted, confirming the value of these features. However, the omission of location improves the EER slightly by 1.1%. This is because the Ped2 dataset contains fewer instances of spatial anomalies compared to Ped1. If the detection of spatial anomalies is not required or desired, then the location feature may be optionally omitted.

Uniformity Offsets	Ped 1				Ped 2				Average			
	GMM		HMM		GMM		HMM		GMM		HMM	
	EER	AUC	EER	AUC	EER	AUC	EER	AUC	EER	AUC	EER	AUC
$(\frac{w}{8}, 0), (0, \frac{h}{8})$	19.6%	0.875	18.5%	0.875	16.1%	0.895	18.7%	0.884	17.8%	0.885	18.6%	0.879
$(\frac{w}{4}, 0), (0, \frac{h}{4})$	20.1%	0.864	18.7%	0.874	14.0%	0.921	17.6%	0.907	17.0%	0.892	18.2%	0.891
$(\frac{w}{2}, 0), (0, \frac{h}{2})$	21.3%	0.858	18.8%	0.874	17.1%	0.901	18.5%	0.865	19.2%	0.879	18.7%	0.869
$(\frac{w}{8}, 0), (\frac{w}{4}, 0), (\frac{w}{2}, 0)$	18.9%	0.873	21.6%	0.852	15.9%	0.896	21.0%	0.835	17.4%	0.884	21.3%	0.844
$(\frac{w}{8}, \frac{h}{8}), (\frac{w}{4}, \frac{h}{4}), (\frac{w}{2}, \frac{h}{2})$	22.5%	0.841	18.1%	0.874	18.1%	0.883	14.6%	0.924	20.3%	0.862	16.3%	0.899
$(0, \frac{h}{8}), (0, \frac{h}{4}), (0, \frac{h}{2})$	22.1%	0.846	19.6%	0.855	13.2%	0.913	21.6%	0.879	17.6%	0.879	20.6%	0.867
$(-\frac{w}{8}, \frac{h}{8}), (-\frac{w}{4}, \frac{h}{4}), (-\frac{w}{2}, \frac{h}{2})$	23.0%	0.830	18.8%	0.863	13.8%	0.907	19.5%	0.893	18.4%	0.868	19.1%	0.878

Table 7.3: Anomaly detection results of the proposed system on the UCSD anomaly detection datasets [125]. Equal error rate (EER) and area under curve (AUC) of the ROC are reported. Average performance across both datasets is also reported, and the best two results in each column are indicated in bold.

Features			Ped 1		Ped 2	
Location	Motion	Uniformity Offsets	EER	AUC	EER	AUC
	s_u, s_v	$(\frac{w}{4}, 0), (0, \frac{h}{4})$	26.1%	0.814	12.9%	0.942
i, j		$(\frac{w}{4}, 0), (0, \frac{h}{4})$	23.2%	0.833	18.9%	0.889
i, j	s_u, s_v		25.9%	0.810	24.5%	0.830
i, j	s_u, s_v	$(\frac{w}{4}, 0), (0, \frac{h}{4})$	20.1%	0.864	14.0%	0.921

Table 7.4: Anomaly detection results with reduced feature sets. In each experiment a different type of feature is omitted. The full feature vector is shown in the bottom row. The uniformity offsets $(\frac{w}{4}, 0), (0, \frac{h}{4})$ were selected, with a GMM classifier.

7.4.3 Comparison to Other Algorithms

In this section the performance of the proposed algorithm is compared to several other visual representations: the social force model ‘SF’ [133], the MPPCA model of optical flow [100], the normalised combination SF-MPPCA [125], the pixel monitoring approach of Adam [3], and the mixture of dynamic textures ‘MDT’ [125]. Results using these visual representations are presented in Table 7.5. The equal error rate (EER) lies between 25-40% for dataset ‘Ped1’ and 25-42% for ‘Ped2’ using these approaches, as reported by Mahadevan [125].

Table 7.5 also includes the results from the proposed algorithm with uniformity offsets $(\frac{w}{4}, 0), (0, \frac{h}{4})$ and GMM classification. These show a significant improvement in EER, with 20.1% and 14.0% for the Ped1 and Ped2 datasets respectively.

More comprehensive results for the proposed system were presented in Table 7.3 (Section 7.4.2); in these experiments, the EER lies between 18.1-23.0% for Ped1 and 13.2-21.6% for Ped2, indicating improved performance (compared to these other methods) regardless of the uniformity offset used.

The proposed algorithm also achieves reasonable processing speeds when oper-

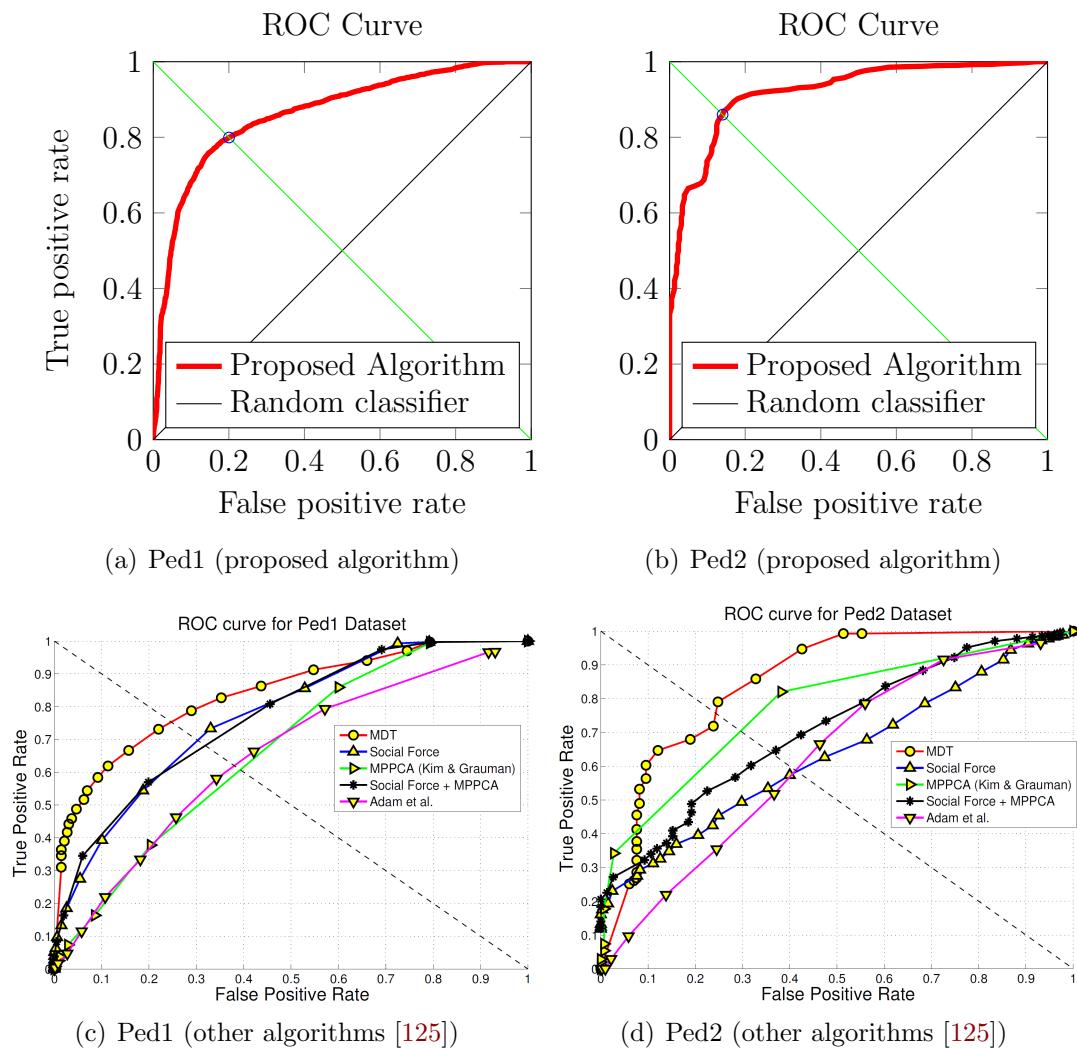


Figure 7.6: ROC curves for the UCSD anomaly detection datasets.

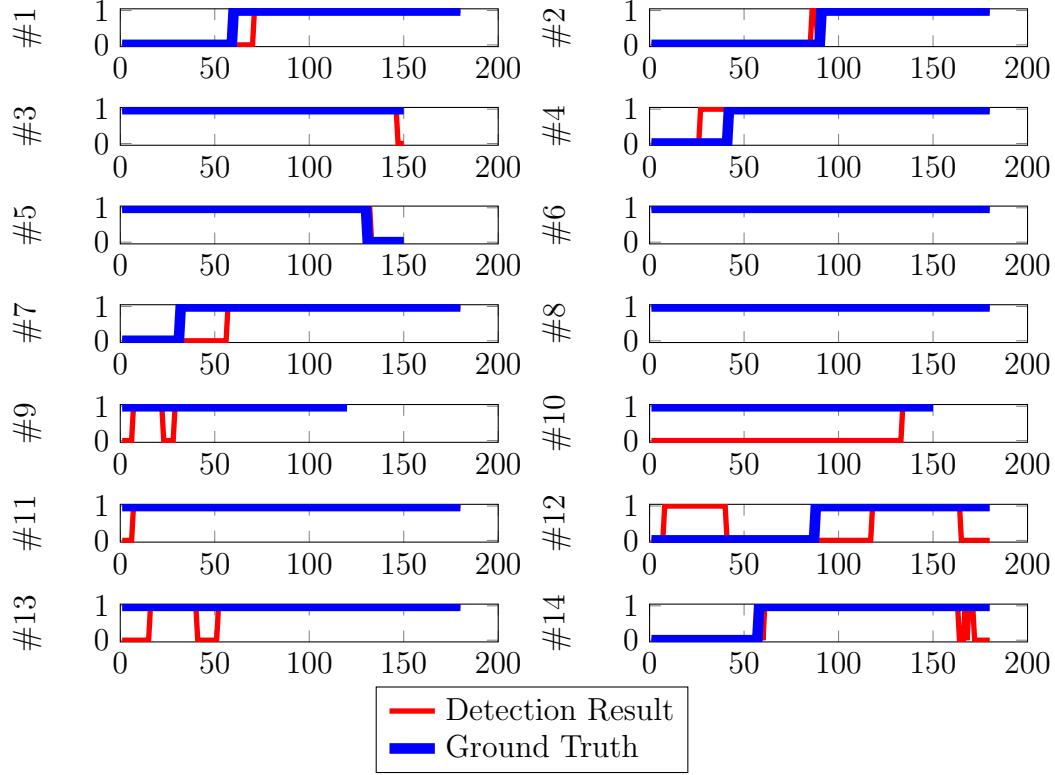


Figure 7.7: Detection results for the Ped2 dataset (14 sequences) at the equal error rate. Normal frames are represented by the value 0, abnormal frames are represented by the value 1.

Visual Representation	EER	
	Ped 1	Ped 2
Social Force (SF) [133]	31%	42%
MPPCA [100]	40%	30%
SF + MPPCA [125]	32%	36%
Adam [3]	38%	42%
Mixture of Dynamic Textures (MDT) [125]	25%	25%
Proposed, $(\frac{w}{4}, 0), (0, \frac{h}{4})$, GMM	20.1%	14.0%

Table 7.5: Anomaly detection results on the UCSD datasets for various algorithms [125]. Equal error rate (EER) and area under curve (AUC) of the ROC are reported.



(a) Slow bicycle not detected at the beginning of the sequence. (b) Slow bicycle detected at the end of the sequence.

Figure 7.8: Partial detection of a slow-moving bicycle in sequence 10 of the Ped2 dataset [125].

ating on a desktop PC. When both GMM and HMM classification are used at the same time, the system operates at approximately 4-5 fps depending on the number of mixtures and states selected. When the GMM is used on its own, the system operates at an average speed of 9.4 fps. This is very close to the source video frame rate of 10 fps. The code has not been fully optimised, as it runs on a single core and does not make use of the GPU, which can be used to significantly improve processing speed.

The processing rates of competing techniques are generally not stated, although many of them (such as [11]) rely on high-resolution optical flow algorithms which are known to be slow. Mahadevan [125] reported a processing speed of 0.04 fps on the UCSD anomaly detection datasets.

7.5 Summary

This chapter proposed a novel anomaly detection algorithm based on local features which are extracted from spatio temporal patches in a video sequence.

Unlike previous methods which use feature discretisation, histogram binning or dimensionality reduction, the proposed approach uses textures of optical flow which are specifically formulated to detect anomalous motion patterns in video sequences.

An investigation was conducted to determine the best features and classification models for anomaly detection. Results demonstrate that the proposed approach consistently outperforms existing algorithms on two benchmark datasets.

The following contributions are noted:

- A novel visual representation called textures of optical flow or uniformity, which is specifically formulated to detect anomalous motion patterns in crowded scenes. The proposed representation measures the uniformity of a flow field in order to detect anomalous objects such as bicycles and vehicles; and can be combined with spatial information to detect other forms of abnormality.
- An investigation into optimal features and classification models to be used in conjunction with the proposed system. It was demonstrated that the proposed approach outperforms state-of-the-art anomaly detection algorithms on two large, publicly-available datasets.
- A number of optimisations were proposed to speed up computation of the proposed algorithm. These included the use of integral images to rapidly calculate patch based features; the use of the K-Means++ algorithm to improve pre-clustering; and foreground masking to reduce the number of data samples without sacrificing relevant motion information.

The proposed approach outperforms the baseline methods introduced by Mahadevan [125]. The proposed algorithm may be indicative of abnormal events which

violate the assumptions behind other crowd monitoring algorithms, as discussed in Chapters 3-6. It can also be used to detect anomalous motion patterns which may be indicative of dangerous or disruptive events.

Chapter 8

Conclusion

8.1 Introduction

This thesis has presented a number of novel algorithms to automatically monitor human crowds using computer vision techniques. Contributions have been made in the fields of crowd counting, crowd flow monitoring, queue monitoring and anomaly detection. These contributions are summarised as follows:

1. Crowd Counting.

This thesis presented a novel approach to crowd counting based on *local features*, which were demonstrated to outperform traditional holistic features. Camera calibration was utilised to achieve *scene invariance* by scaling features appropriately between viewpoints. Additionally, *multi camera* crowd counting was achieved by using camera calibration to compensate for regions of overlap within a multi camera network.

2. Crowd Flow Monitoring.

This thesis proposed a novel algorithm which combines salient keypoint detection and regression of optical flow to count the number of people passing through a ‘virtual gate’. Temporal windows and optical flow histograms were also shown to improve performance.

3. Queue Monitoring.

This thesis proposed a novel algorithm which combines crowd counting and virtual gates to monitor queue parameters automatically. The proposed framework can monitor parameters such as queue length, growth rate, arrival rate and service rate.

4. Anomaly Detection.

This thesis proposed a novel visual representation called *textures of optical flow* which captures the properties of motion patterns in crowded environments by applying traditional textural features directly to an optical flow field. Results demonstrated that the proposed approach outperforms existing algorithms on benchmark anomaly detection sequences.

The original contributions of this thesis are discussed in further detail in Section 8.2. Directions for future research are discussed in Section 8.3.

8.2 Summary of Contribution

In Chapter 3, a novel approach to crowd counting was presented, which replaced holistic features with localised feature extraction. Instead of estimating a crowd globally, the problem was broken down into groups detected using a foreground detection algorithm, and features were extracted from each segment.

The concept of a *pedestrian template* was introduced to model the approximate size of a pedestrian located at each position in an image. The pedestrian template served two functions:

1. It enabled feature normalisation across an image to account for the effects of perspective. Each pixel was weighted by the inverse of the area of the pedestrian template centred at that pixel
2. It simplified the ground truth annotation process. Given a set of ‘dot’ annotations at each pedestrian’s centre, the bounding box of each pedestrian can be approximated by the pedestrian template at that location in the image plane. This enables localised blob annotations to be performed by the system automatically, and therefore expedites the training process.

An automated procedure for constructing the pedestrian template model was proposed: the system used a pedestrian detection algorithm to learn the relative sizes of objects at different locations in the image.

A novel method of refining the estimates using group tracking was also described. Unlike previous approaches which ignore temporal correlations, or apply holistic smoothing, the proposed approach identified groups as they moved through a scene and tracked them over time. This enabled smoothing to be applied on a local level.

A comprehensive analysis of various crowd counting algorithms across five datasets was performed, and the following conclusions were reached:

- The use of local features consistently outperformed holistic features.
- A greater quantity of local features generally improved performance compared to fewer features. The best performance was observed with the fol-

lowing feature vectors: ‘size, shape, edges, keypoints’ (all features) and ‘shape, edges, keypoints’. (The omission of size did not improve or detract from the overall performance.)

- The use of Gaussian process regression consistently outperformed the other regression models tested (linear regression, K -nearest neighbours and neural networks).
- The proposed tracking algorithm improved crowd counting accuracy, except on the Mall dataset, due to its low frame rate of 2 fps.
- The proposed crowd counting system outperformed a number of existing approaches, both local and holistic. These included Kong [103], Chan [36], Lempitsky [110], Conte [55], Albiol [6] and Chen [42].

In Chapter 4, a novel *scene invariant* crowd counting algorithm was proposed, to overcome the limitations of existing single-camera methods. The system uses camera calibration to construct a 3D cylindrical pedestrian template which can be projected into the image plane around any pixel. The method uses real-world reference coordinates, and therefore enables scene invariant crowd counting to be performed by normalising features across viewpoints. This allows the system to be trained on one or more viewpoints and then deployed on one or more other viewpoints for counting, without any additional training required.

Three new benchmark datasets for crowd counting were introduced, collected from the Queensland University of Technology (QUT), with ground truth annotations and camera calibration parameters. These sequences feature challenging reflections, shadows and lighting fluctuations, and were used to demonstrate the efficacy of the proposed methodology. In total, seven calibrated datasets were used to demonstrate the scene invariance of the proposed algorithm. The following conclusions were reached:

- The proposed algorithm can successfully scale features across viewpoints and achieve accurate scene invariant crowd counting results.
- The use of multiple image features (size, shape, edges, keypoints) outperformed the use of fewer features across a wide range of datasets.
- The use of Gaussian process regression consistently outperformed the use of linear regression, neural networks and K -nearest neighbours.

In Chapter 5, two novel methods for *multi camera* crowd counting were proposed. These algorithms makes use of a new concept called the *overlap map* which is constructed using camera calibration parameters, and provides an estimate for the amount of overlap at each region of an image.

The combination of scene invariance and multi camera crowd counting was demonstrated using a three-camera environment. In this experiment, the system was trained on a completely different environment (QUT database, Cameras 3, 5 and 8) and then deployed for testing on the PETS 2009 database [149] (Views 1, 2 and 3). A mean relative error of 6-8% was observed in this environment, demonstrating highly accurate crowd counting which was both scene invariant and spanned multiple cameras.

In Chapter 6, a novel crowd flow monitoring algorithm called the *virtual gate* was proposed. The proposed algorithm combines feature detection and regression of optical flow techniques to estimate the number of pedestrians passing through a region of interest. Optical flow histograms were shown to improve performance by separating the effects of small motion (potentially noise) and large motion. Local temporal windows (analogous to local crowd counting features) were proposed. This approach breaks a video sequence into a set of subsequences, so that counting can be performed locally in time.

A comprehensive evaluation was performed on five datasets collected from the Brisbane International Airport, and the following conclusions were reached:

- The proposed algorithm can successfully count pedestrians passing through a virtual gate.
- The use of edges as well as all pixels within the ROI provide the best performance (compared to various combinations of all pixels, edges and corners).
- The use of linear regression and GPR provide the best performance (compared to K -nearest neighbours and neural networks).
- The use of optical flow histograms improves performance.
- The use of local temporal windows improves performance.

Chapter 6 also presented a novel queue monitoring algorithm which combined crowd counting and virtual gate technologies within the same framework. This novel combination enables queue statistics such as arrival rate, service rate, queue size and growth rate to be monitored. The framework also enables these parameters to be estimated in the absence of adequate video coverage: for example, if the entry or exit to the queue is not monitored, the arrival or service rate can be estimated using the other statistics.

An evaluation on real world queue data demonstrated the efficacy of the proposed algorithm. Unlike existing crowd datasets which are captured in unordered public spaces such as shopping malls and outdoor walkways, the dataset used in this analysis was captured at the Brisbane International Airport.

Chapter 7 proposed a novel anomaly detection algorithm based on local image features extracted from spatio-temporal patches in a video sequence. While exist-

ing methods have used reduction techniques such as quantisation, histogram bins or dimensionality reduction, the proposed approach is based on a novel feature called *textures of optical flow* which is formulated to detect anomalous motion patterns in video sequences. The proposed feature measures the uniformity of an optical flow field in order to detect anomalous objects such as bicycles and vehicles in human crowds.

An evaluation across two large, publicly available anomaly detection datasets demonstrated the efficacy of the proposed algorithm compared to existing methods. The following conclusions were reached:

- The proposed algorithm based on textures of optical flow outperformed existing methods based on traditional visual representations.
- The optimal uniformity offset for the datasets used in this analysis was $(\frac{w}{4}, 0), (0, \frac{h}{4})$.
- The HMM classifier did not confer any additional benefit compared to the GMM (when classifying the patch-based features used in this thesis). Additionally, the GMM based approach operated at a higher frame rate of 9.4 fps (approximately real time). This is significantly faster than other methods; for example, Mahadevan [125] reported a frame rate of 0.04 fps.

A number of optimisations were proposed to speed up computation of the proposed algorithm. These included the use of integral images to rapidly calculate patch based features; the use of the K-Means++ algorithm to improve pre-clustering; and foreground masking to reduce the number of data samples without sacrificing relevant motion information.

8.3 Future Research

This thesis has provided a number of novel contributions to the field of crowd monitoring using computer vision. Avenues for additional future research are listed below.

- Exploration of auto-calibration methods could potentially simplify the construction of the pedestrian template and the overlap map, which underpin the scene invariant and multi camera crowd counting algorithms respectively.
- The use of bidirectional segmentation methods, such as the mixture of dynamic textures [37], could be used to extend the current virtual gate into a bidirectional system.
- In addition to anomaly detection, textures of optical flow may be considered for event modelling and event recognition purposes. Exploration of additional textural features, which may be applied to optical flow fields, is also warranted.

Appendix A

Related to Chapter 3

A.1 Vanishing Point Calculation

This section refers to Figure 3.2 (p. 107). The first reference line P_1P_2 is defined by the linear equation:

$$y_A = m_A x + c_A \quad (\text{A.1})$$

where

$$m_A = \frac{y_2 - y_1}{x_2 - x_1} \quad (\text{A.2})$$

$$c_A = y_2 - m_A x_2 \quad (\text{A.3})$$

Similarly, the line P_3P_4 is described by:

$$y_B = m_B x + c_B \quad (\text{A.4})$$

where

$$m_B = \frac{y_4 - y_3}{x_4 - x_3} \quad (\text{A.5})$$

$$c_B = y_4 - m_B x_4 \quad (\text{A.6})$$

The vanishing point therefore occurs at:

$$y_A = y_B \quad (\text{A.7})$$

$$m_A x_v + c_A = m_B x_v + c_B \quad (\text{A.8})$$

$$x_v = \frac{c_B - c_A}{m_A - m_B} \quad (\text{A.9})$$

$$y_v = m_A x_v + c_A \quad (\text{A.10})$$

This value of y_v is used in the computation of $S(y)$ in Equation 3.8 (p. 108).

A.2 Object Size Interpolation

This section refers to Figure 3.3 (p. 109). The interpolated person height h is defined by the linear equation:

$$h = m_h y + c_h \quad (\text{A.11})$$

where

$$m_h = \frac{h_2 - h_1}{y_2 - y_1} \quad (\text{A.12})$$

$$c_h = h_1 - m_h y_1 \quad (\text{A.13})$$

and y_2 and y_1 are the y -coordinates of line ab and cd in the image plane, respectively. Similarly, the interpolated width of the walkway is described by:

$$w = m_w y + c_w \quad (\text{A.14})$$

where

$$m_w = \frac{w_2 - w_1}{y_2 - y_1} \quad (\text{A.15})$$

$$c_w = w_1 - m_w y_1 \quad (\text{A.16})$$

These interpolated values of h and w are used in the computation of $S(y)$ in Equation 3.12 (p. 110).

A.3 Robust Regression using IRLS

This section describes robust regression using iteratively reweighted least squares (IRLS). Consider a linear model described by,

$$y_i = \mathbf{x}_i^T \mathbf{b} + \epsilon_i \quad (\text{A.17})$$

where $\{\mathbf{x}_i\}_{i=1}^m$ denotes the independent variables, \mathbf{b} denotes the unknown vector of parameters (linear weights) applied to these variables, $\{y_i\}_{i=1}^m$ denotes the observed values and ϵ_i denotes the observation error or residual. M -Estimation seeks to minimise the sum of a function of residuals:

$$\hat{\mathbf{b}} = \underset{\mathbf{b}}{\operatorname{argmin}} \sum_{i=1}^m \rho(\epsilon_i) \quad (\text{A.18})$$

$$= \underset{\mathbf{b}}{\operatorname{argmin}} \sum_{i=1}^m \rho(y_i - \mathbf{x}_i^T \mathbf{b}) \quad (\text{A.19})$$

where ρ is an error function. For ordinary least squares regression a quadratic penalty is used,

$$\rho_{LS}(z) = z^2 \quad (\text{A.20})$$

while other functions are used to reduce the influence of outliers. For example, the Tukey bisquare function is:

$$\rho_B(z) = \begin{cases} \frac{k^2}{6} \left[1 - \left(1 - \left(\frac{z}{k} \right)^2 \right)^3 \right] & |z| \leq k \\ \frac{k^2}{6} & |z| > k \end{cases} \quad (\text{A.21})$$

where k is a tuning constant. These functions and their derivatives are plotted in Figure A.1. Let the first derivative of $\rho(z)$ be denoted:

$$\psi(z) = \rho'(z) \quad (\text{A.22})$$

The minimum value of Equation A.19 is found by setting the partial derivatives to zero:

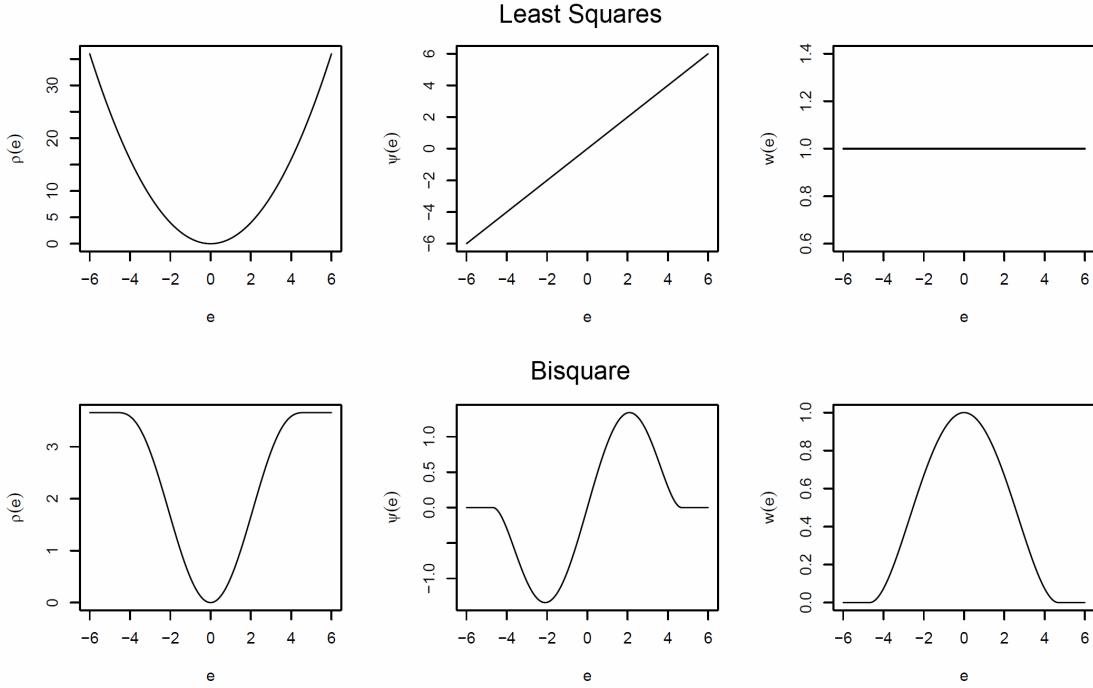


Figure A.1: Least squares and Tukey bisquare penalty functions, their derivatives and weight functions [70].

$$\mathbf{0} = \frac{\partial}{\partial \mathbf{b}} \sum_{i=1}^m \rho(y_i - \mathbf{x}_i^T \mathbf{b}) \quad (\text{A.23})$$

$$= \sum_{i=1}^m \psi(y_i - \mathbf{x}_i^T \mathbf{b}) \mathbf{x}_i^T \quad (\text{A.24})$$

A weight function is defined,

$$w_i = w(\epsilon_i) = \frac{\psi(\epsilon_i)}{\epsilon_i} \quad (\text{A.25})$$

and these are plotted in Figure A.1. Equation A.24 may then be rewritten:

$$\mathbf{0} = \sum_{i=1}^m w_i (y_i - \mathbf{x}_i^T \mathbf{b}) \mathbf{x}_i^T \quad (\text{A.26})$$

This is solved iteratively using the following procedure:

1. Select initial estimates for \mathbf{b}^0 such as the ordinary least squares regression estimate. Then iteratively repeat steps 2 and 3 as follows.
2. Calculate the residuals $\epsilon_i^{t+1} = y_i - \mathbf{x}^T \mathbf{b}^t$ and weights $w_i^{t+1} = w(\epsilon_i^{t+1})$.
3. Solve for the new weighted least squares estimates:

$$\mathbf{b}^{t+1} = [\mathbf{X}^T \mathbf{W}^t \mathbf{X}]^{-1} \mathbf{X}^T \mathbf{W}^t \mathbf{y} \quad (\text{A.27})$$

where \mathbf{X} is the model matrix whose i th row is \mathbf{x}^T and \mathbf{W}^t is a diagonal matrix containing the current weights.

The iteratively reweighted least squares (IRLS) procedure repeats steps 2 and 3 until convergence of $\hat{\mathbf{b}}$ is achieved. The algorithm is implemented by MATLAB's **robustfit** function [132], and the bisquare penalty function ρ_B is used to reduce the influence of outliers compared to the ordinary least squares function ρ_{LS} .

Appendix B

Related to Chapter 4

B.1 Regression Model Comparison Tables

The following tables provide supplementary information to Section 4.3.2.2 (p. 236). The error rates and corresponding rankings for each regression model across each of the seven datasets is listed.

Regression Model	MAE	MSE	MRE
GPR	1.321 (1)	4.250 (1)	10.32% (1)
Linear	1.474 (2)	5.167 (2)	12.38% (2)
KNN (K=1)	1.973 (5)	10.594 (6)	13.63% (5)
KNN (K=2)	1.840 (4)	9.128 (4)	13.21% (4)
KNN (K=4)	1.808 (3)	8.430 (3)	13.16% (3)
KNN (K=8)	2.059 (6)	10.429 (5)	13.89% (6)
KNN (K=16)	2.876 (7)	25.401 (8)	16.54% (7)
KNN (K=32)	3.827 (9)	51.536 (10)	20.20% (8)
NN (4)	8.810 (12)	103.788 (12)	96.58% (12)
NN (8)	3.124 (8)	16.585 (7)	33.64% (9)
NN (16)	7.867 (11)	100.878 (11)	82.52% (11)
NN (32)	5.453 (10)	40.214 (9)	74.45% (10)

Table B.1: Comparison of regression on **PETS 2009, View 1.**

Regression Model	MAE	MSE	MRE
GPR	3.365 (2)	17.514 (2)	9.55% (2)
Linear	2.911 (1)	12.897 (1)	8.98% (1)
KNN (K=1)	7.779 (3)	136.999 (5)	21.79% (3)
KNN (K=2)	9.872 (6)	153.337 (6)	25.70% (5)
KNN (K=4)	13.000 (7)	250.579 (8)	33.14% (6)
KNN (K=8)	15.592 (9)	347.145 (9)	39.73% (9)
KNN (K=16)	18.980 (10)	490.416 (10)	49.77% (10)
KNN (K=32)	21.800 (11)	622.059 (11)	58.40% (11)
NN (4)	13.604 (8)	242.907 (7)	37.64% (8)
NN (8)	8.659 (4)	127.045 (4)	23.71% (4)
NN (16)	9.534 (5)	125.898 (3)	35.34% (7)
NN (32)	32.485 (12)	1336.771 (12)	88.27% (12)

Table B.2: Comparison of regression on **PETS 2009, View 2**.

Regression Model	MAE	MSE	MRE
GPR	0.405 (6)	0.495 (6)	24.98% (6)
Linear	0.551 (10)	0.714 (9)	38.45% (11)
KNN (K=1)	0.459 (7)	0.850 (10)	25.63% (7)
KNN (K=2)	0.366 (5)	0.448 (5)	21.39% (3)
KNN (K=4)	0.338 (2)	0.379 (4)	21.34% (2)
KNN (K=8)	0.349 (4)	0.353 (3)	22.28% (4)
KNN (K=16)	0.341 (3)	0.319 (2)	22.41% (5)
KNN (K=32)	0.316 (1)	0.245 (1)	20.93% (1)
NN (4)	0.468 (9)	0.575 (8)	27.74% (9)
NN (8)	1.996 (12)	6.390 (12)	129.74% (12)
NN (16)	0.465 (8)	0.517 (7)	27.03% (8)
NN (32)	0.626 (11)	1.275 (11)	31.55% (10)

Table B.3: Comparison of regression on **PETS 2006, View 3.**

Regression Model	MAE	MSE	MRE
GPR	0.487 (2)	0.441 (4)	22.20% (2)
Linear	0.852 (8)	1.077 (8)	39.76% (8)
KNN (K=1)	0.562 (6)	0.648 (6)	23.38% (7)
KNN (K=2)	0.574 (7)	0.659 (7)	23.00% (6)
KNN (K=4)	0.519 (5)	0.483 (5)	22.50% (4)
KNN (K=8)	0.513 (4)	0.436 (3)	22.81% (5)
KNN (K=16)	0.495 (3)	0.421 (2)	22.49% (3)
KNN (K=32)	0.457 (1)	0.363 (1)	21.18% (1)
NN (4)	5.422 (11)	52.948 (11)	252.33% (11)
NN (8)	4.092 (10)	37.418 (10)	199.34% (10)
NN (16)	6.508 (12)	70.978 (12)	297.69% (12)
NN (32)	3.542 (9)	21.317 (9)	177.61% (9)

Table B.4: Comparison of regression on **PETS 2006, View 4.**

Regression Model	MAE	MSE	MRE
GPR	1.547 (3)	3.506 (3)	14.03% (3)
Linear	1.202 (1)	2.365 (1)	11.38% (1)
KNN (K=1)	3.541 (8)	18.097 (8)	30.12% (8)
KNN (K=2)	3.261 (7)	16.139 (7)	28.10% (7)
KNN (K=4)	2.900 (6)	10.783 (6)	25.08% (6)
KNN (K=8)	2.483 (5)	7.978 (5)	22.69% (5)
KNN (K=16)	1.744 (4)	4.132 (4)	16.56% (4)
KNN (K=32)	1.275 (2)	2.392 (2)	12.54% (2)
NN (4)	12.947 (11)	194.591 (10)	119.64% (11)
NN (8)	11.864 (10)	198.944 (11)	119.29% (10)
NN (16)	8.423 (9)	94.449 (9)	72.98% (9)
NN (32)	17.169 (12)	430.348 (12)	171.31% (12)

Table B.5: Comparison of regression on **QUT, Camera 3**.

Regression Model	MAE	MSE	MRE
GPR	0.886 (1)	1.524 (1)	12.17% (1)
Linear	1.097 (2)	2.193 (2)	14.90% (2)
KNN (K=1)	1.543 (7)	5.016 (7)	17.91% (7)
KNN (K=2)	1.271 (5)	3.002 (4)	16.06% (5)
KNN (K=4)	1.193 (3)	2.792 (3)	15.34% (4)
KNN (K=8)	1.265 (4)	3.238 (5)	15.23% (3)
KNN (K=16)	1.399 (6)	4.151 (6)	16.18% (6)
KNN (K=32)	1.638 (8)	5.808 (8)	18.04% (8)
NN (4)	7.333 (11)	66.930 (11)	112.04% (11)
NN (8)	11.483 (12)	180.752 (12)	189.94% (12)
NN (16)	3.858 (10)	24.775 (10)	60.69% (10)
NN (32)	2.403 (9)	11.269 (9)	29.22% (9)

Table B.6: Comparison of regression on **QUT, Camera 5**.

Regression Model	MAE	MSE	MRE
GPR	1.448 (2)	3.625 (2)	18.20% (2)
Linear	2.306 (8)	7.451 (8)	27.53% (8)
KNN (K=1)	1.860 (7)	6.365 (7)	26.34% (6)
KNN (K=2)	1.770 (4)	5.196 (5)	26.00% (5)
KNN (K=4)	1.854 (6)	5.553 (6)	26.82% (7)
KNN (K=8)	1.818 (5)	4.997 (4)	24.29% (4)
KNN (K=16)	1.579 (3)	3.985 (3)	20.98% (3)
KNN (K=32)	1.160 (1)	1.941 (1)	16.78% (1)
NN (4)	9.671 (10)	159.586 (10)	124.49% (10)
NN (8)	35.432 (12)	1669.462 (12)	443.49% (12)
NN (16)	6.840 (9)	60.679 (9)	87.43% (9)
NN (32)	16.948 (11)	462.636 (11)	197.25% (11)

Table B.7: Comparison of regression on **QUT, Camera 8**.

Appendix C

Hidden Markov Models

The information in this appendix is supplementary to Section 2.7.2 (p. 90) regarding HMMs.

C.1 Implementation

In practice the computation of the forward and reverse variables α, β tend exponentially toward zero, and for long sequences this can exceed the machine's precision. Thus the iterative procedures can be updated using a scaling factor $s(t)$ at each time step:

1. Initialisation:

$$\hat{\alpha}_i(1) = \frac{\pi_i b_i(1)}{s(1)} \quad (\text{C.1})$$

where

$$s(1) = \sum_{i=1}^N \pi_i b_i(1) \quad (\text{C.2})$$

2. Induction:

$$\hat{\alpha}_j(t+1) = \frac{\left[\sum_{i=1}^N \hat{\alpha}_i(t) a_{ij} \right] b_j(t+1)}{s(t+1)} \quad (\text{C.3})$$

where

$$s(t+1) = \sum_{j=1}^N \left(\left[\sum_{i=1}^N \hat{\alpha}_i(t) a_{ij} \right] b_j(t+1) \right) \quad (\text{C.4})$$

Note that this means:

$$\hat{\alpha}_i(t) = \frac{\alpha_i(t)}{\prod_{\tau=1}^t s(\tau)} \quad (\text{C.5})$$

3. Termination:

$$p(O|\lambda) = \sum_{i=1}^N \alpha_i(T) \quad (\text{C.6})$$

$$= \left(\prod_{t=1}^T s(t) \right) \left(\sum_{i=1}^N \hat{\alpha}_i(T) \right) \quad (\text{C.7})$$

$$= \prod_{t=1}^T s(t) \quad (\text{C.8})$$

$$\log p(O|\lambda) = \sum_{t=1}^T \log s(t) \quad (\text{C.9})$$

The reverse procedure is also modified using the *same* scaling factors:

1. Initialisation:

$$\hat{\beta}_i(T) = \frac{1}{s(T)} \quad (\text{C.10})$$

2. Induction:

$$\hat{\beta}_i(t) = \frac{\sum_{j=1}^N a_{ij} b_j(t+1) \hat{\beta}_j(t+1)}{s(t)} \quad (\text{C.11})$$

Note that this means:

$$\hat{\beta}_i(t) = \frac{\beta_i(t)}{\prod_{\tau=t}^T s(\tau)} \quad (\text{C.12})$$

From Equations C.5 and C.12 it follows:

$$\hat{\alpha}_i(t)\hat{\beta}_i(t) = \alpha_i(t)\beta_i(t) \left[\left(\prod_{\tau=1}^t s(\tau) \right) \left(\prod_{\tau=t}^T s(\tau) \right) \right]^{-1} \quad (\text{C.13})$$

$$= \alpha_i(t)\beta_i(t)\kappa(t) \quad (\text{C.14})$$

Thus the product $\hat{\alpha}_i(t)\hat{\beta}_i(t)$ can be used in place of $\alpha_i(t)\beta_i(t)$ in Equation 2.92 (p. 94), because the κ term will cancel in the numerator and denominator. Similarly,

$$\hat{\alpha}_i(t)\hat{\beta}_j(t+1) = \alpha_i(t)\beta_i(t+1) \left[\left(\prod_{\tau=1}^t s(\tau) \right) \left(\prod_{\tau=t+1}^T s(\tau) \right) \right]^{-1} \quad (\text{C.15})$$

$$= \alpha_i(t)\beta_j(t+1)\psi \quad (\text{C.16})$$

Thus the product $\hat{\alpha}_i(t)\hat{\beta}_j(t+1)$ can be used in place of $\alpha_i(t)\beta_j(t+1)$ in Equation 2.96 (p. 94) as the ψ term will cancel out.

C.2 Multiple Observation Sequences

When there are multiple observation sequences, $O = \{O_e\}_{e=1}^E$ with $O_e = \{o_t^e\}_{t=1}^{T_e}$, and hidden state sequences, $Q = \{Q_e\}_{e=1}^E$ with $Q_e = \{q_t^e\}_{t=1}^{T_e}$, the likelihood becomes:

$$p(O|\lambda) = \prod_{e=1}^E p(O_e|\lambda) \quad (\text{C.17})$$

$$\log p(O|\lambda) = \sum_{e=1}^E \log p(O_e|\lambda) \quad (\text{C.18})$$

Forward procedure:

$$s^e(1) = \sum_{i=1}^N \pi_i b_i^e(1) \quad (\text{C.19})$$

$$\hat{\alpha}_i^e(1) = \frac{\pi_i b_i^e(1)}{s^e(1)} \quad (\text{C.20})$$

$$s^e(t+1) = \sum_{j=1}^N \left(\left[\sum_{i=1}^N \hat{\alpha}_i^e(t) a_{ij} \right] b_j^e(t+1) \right) \quad (\text{C.21})$$

$$\hat{\alpha}_j^e(t+1) = \frac{\left[\sum_{i=1}^N \hat{\alpha}_i^e(t) a_{ij} \right] b_j^e(t+1)}{s^e(t+1)} \quad (\text{C.22})$$

$$\log p(O_e | \lambda) = \sum_{t=1}^{T_e} \log s^e(t) \quad (\text{C.23})$$

$$(\text{C.24})$$

Backward procedure:

$$\hat{\beta}_i^e(T) = \frac{1}{s^e(T)} \quad (\text{C.25})$$

$$\hat{\beta}_i^e(t) = \frac{\sum_{j=1}^N a_{ij} b_j^e(t+1) \hat{\beta}_j^e(t+1)}{s^e(t)} \quad (\text{C.26})$$

The expectation equations are:

$$\gamma_i^e(t) = \frac{\hat{\alpha}_i^e(t) \hat{\beta}_i^e(t)}{\sum_{j=1}^N \hat{\alpha}_j^e(t) \hat{\beta}_j^e(t)} \quad (\text{C.27})$$

$$\gamma_{i\ell}^e(t) = \gamma_i^e(t) \frac{c_{i\ell} b_{i\ell}^e(t)}{b_i^e(t)} \quad (\text{C.28})$$

$$\xi_{i,j}^e(t) = \frac{\hat{\alpha}_i^e(t) a_{ij} b_j^e(t+1) \hat{\beta}_j^e(t+1)}{\sum_{i=1}^N \sum_{j=1}^N \hat{\alpha}_i^e(t) a_{ij} b_j^e(t+1) \hat{\beta}_j^e(t+1)} \quad (\text{C.29})$$

The re-estimation equations are:

$$\pi_i = \frac{\sum_{e=1}^E \gamma_i^e(1)}{E} \quad (\text{C.30})$$

$$a_{ij} = \frac{\sum_{e=1}^E \sum_{t=1}^{T_e-1} \xi_{i,j}^e(t)}{\sum_{e=1}^E \sum_{t=1}^{T_e-1} \gamma_i^e(t)} \quad (\text{C.31})$$

$$c_{i\ell} = \frac{\sum_{e=1}^E \sum_{t=1}^{T_e} \gamma_{i\ell}^e(t)}{\sum_{e=1}^E \sum_{t=1}^{T_e} \gamma_i^e(t)} \quad (\text{C.32})$$

$$\mu_{i\ell} = \frac{\sum_{e=1}^E \sum_{t=1}^{T_e} \gamma_{i\ell}^e(t) o_t^e}{\sum_{e=1}^E \sum_{t=1}^{T_e} \gamma_{i\ell}^e(t)} \quad (\text{C.33})$$

$$\Sigma_{i\ell} = \frac{\sum_{e=1}^E \sum_{t=1}^{T_e} \gamma_{i\ell}^e(t) (o_t^e - \mu_{i\ell})(o_t^e - \mu_{i\ell})^T}{\sum_{e=1}^E \sum_{t=1}^{T_e} \gamma_{i\ell}^e(t)} \quad (\text{C.34})$$

Bibliography

- [1] Y.I. Abdel-Aziz and H.M. Karara. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In *Proceedings of the Symposium on Close-Range Photogrammetry*, pages 1–18, 1971. [207](#)
- [2] G. Acampora, V. Loia, G. Percannella, and M. Vento. Trainable estimators for indirect people counting: A comparative study. In *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, pages 139–145, June 2011. [doi:10.1109/FUZZY.2011.6007637](#). [65](#)
- [3] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. Robust real-time unusual event detection using multiple fixed-location monitors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(3):555–560, March 2008. [doi:10.1109/TPAMI.2007.70825](#). [20](#), [87](#), [88](#), [332](#), [348](#), [350](#)
- [4] Anubhav Agarwal, C. V. Jawahar, and P. J. Narayanan. A survey of planar homography estimation techniques. Technical report, Centre for Visual Information Technology, International Institute of Information Technology, 2005. URL: <http://cvit.iiit.ac.in/papers/anubhav05ASurvey.pdf>.
- [5] AgilityVideo. AgilityVideo - automated video surveillance protecting criti-

- cal infrastructure | secure your critical infrastructure with intelligent video analysis, 2013. URL: <http://www.agilityvideo.com/>. 3
- [6] A. Albiol and J. Silla. Statistical video analysis for crowds counting. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 2569–2572, November 2009. [doi:10.1109/ICIP.2009.5414002](https://doi.org/10.1109/ICIP.2009.5414002). 65, 69, 131, 190, 204, 358
 - [7] Saad Ali and Mubarak Shah. Floor fields for tracking in high density crowd scenes. In *Computer Vision – ECCV 2008*, volume 5303 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin / Heidelberg, 2008. [doi:10.1007/978-3-540-88688-4_1](https://doi.org/10.1007/978-3-540-88688-4_1). 70, 275
 - [8] E.L. Andrade, S. Blunsden, and R.B. Fisher. Hidden Markov models for optical flow analysis in crowds. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 460–463, 2006. [doi:10.1109/ICPR.2006.621](https://doi.org/10.1109/ICPR.2006.621). 82, 332
 - [9] E.L. Andrade, S. Blunsden, and R.B. Fisher. Modelling crowd scenes for event detection. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 175–178, 2006. [doi:10.1109/ICPR.2006.806](https://doi.org/10.1109/ICPR.2006.806). 82, 88
 - [10] E.L. Andrade, S.J. Blunsden, and R.B. Fisher. Performance analysis of event detection models in crowded scenes. *IET Conference Publications*, 2006(CP522):427–432, 2006. [doi:10.1049/cp:20060569](https://doi.org/10.1049/cp:20060569). 82, 88
 - [11] E.L. Andrade, R.B. Fisher, and S. Blunsden. Detection of emergency events in crowded scenes. In *Crime and Security, 2006. The Institution of Engineering and Technology Conference on*, pages 528–533, June 2006. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4123814. 79, 82, 85, 88, 320, 327, 332, 351

- [12] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. URL: <http://theory.stanford.edu/~sergei/papers/kMeansPP-soda.pdf>. 341
- [13] D. Aubert. Passengers queue length measurement. In *Image Analysis and Processing, 1999. Proceedings. International Conference on*, pages 1132–1135, 1999. doi:[10.1109/ICIAP.1999.797754](https://doi.org/10.1109/ICIAP.1999.797754). 72, 73
- [14] Douglas Ayers and Mubarak Shah. Monitoring human behavior from video taken in an office environment. *Image and Vision Computing*, 19(12):833–846, 2001. doi:[10.1016/S0262-8856\(01\)00047-6](https://doi.org/10.1016/S0262-8856(01)00047-6). 76
- [15] Kheir-Eddine Aziz, Djamel Merad, Bernard Fertil, and Nicolas Thome. Pedestrian head detection and tracking using skeleton graph for people counting in crowded environments. In *Machine Vision Applications. MVA 2011. IAPR Conference on*, pages 516–519, 2011. URL: <http://www.mva-org.jp/Proceedings/2011CD/papers/14-25.pdf>. 63
- [16] Simon Baker, Daniel Scharstein, J. Lewis, Stefan Roth, Michael Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. URL: <http://vision.middlebury.edu/flow/>, doi:[10.1007/s11263-010-0390-2](https://doi.org/10.1007/s11263-010-0390-2). 25, 154
- [17] J. Barandiaran, B. Murguia, and F. Boto. Real-time people counting using multiple lines. In *Image Analysis for Multimedia Interactive Services, 2008. WIAMIS '08. Ninth International Workshop on*, pages 159–162, May 2008. doi:[10.1109/WIAMIS.2008.27](https://doi.org/10.1109/WIAMIS.2008.27). 69
- [18] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*,

- 110(3):346–359, June 2008. [doi:10.1016/j.cviu.2007.09.014](https://doi.org/10.1016/j.cviu.2007.09.014). xxvi, xxxviii, 65, 131, 132
- [19] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, number 3951 in Lecture Notes in Computer Science, pages 404–417. Springer Berlin Heidelberg, January 2006. [doi:10.1007/11744023_32](https://doi.org/10.1007/11744023_32). 131
- [20] Yassine Benabbas, N. Ihaddadene, T. Yahiaoui, T. Urruty, and C. Djeraba. Spatio-temporal optical flow analysis for people counting. In *Advanced Video and Signal Based Surveillance (AVSS). 2010 Seventh IEEE International Conference on*, pages 212–217, 2010. [doi:10.1109/AVSS.2010.29](https://doi.org/10.1109/AVSS.2010.29). 70, 275
- [21] Jeff Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical report, International Computer Science Institute, 1998. URL: <ftp://ftp.icsi.berkeley.edu/pub/techreports/1997/tr-97-021.pdf>. 90, 91, 93, 339, 340
- [22] Michael J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, January 1996. [doi:10.1006/cviu.1996.0006](https://doi.org/10.1006/cviu.1996.0006). 24, 26, 327
- [23] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003. URL: <http://jmlr.org/papers/volume3/blei03a/blei03a.pdf>. 79
- [24] B. Boghossian and J. Black. The challenges of robust 24/7 video surveillance systems. In *Imaging for Crime Detection and Prevention, 2005. ICDP*

2005. *The IEEE International Symposium on*, pages 33–38, June 2005.
[doi:10.1109/IC.2005.0066](https://doi.org/10.1109/IC.2005.0066). 3
- [25] Gustave Le Bon. *The Crowd: A Study of the Popular Mind*. 1895. URL:
<http://www.gutenberg.org/ebooks/445>. 1
- [26] Biswajit Bose and Eric Grimson. Ground plane rectification by tracking moving objects. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS), Joint IEEE International Workshop on*, 2003. 210
- [27] M. Bozzoli, L. Cinque, and E. Sanginetto. A statistical method for people counting in crowded environments. In *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*, pages 506–511, September 2007. [doi:10.1109/ICIAP.2007.4362828](https://doi.org/10.1109/ICIAP.2007.4362828). 70, 275
- [28] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/Kanade meets Horn/Schunck: combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, February 2005.
[doi:10.1023/B:VISI.0000045324.43199.43](https://doi.org/10.1023/B:VISI.0000045324.43199.43). 22
- [29] Hung H. Bui, Svetha Venkatesh, and Geoff West. Tracking and surveillance in wide-area spatial environments using the abstract hidden Markov model. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):177–196, February 2001. [doi:10.1142/S0218001401000782](https://doi.org/10.1142/S0218001401000782).
76
- [30] D. Butler, S. Sridharan, and Jr. Bove, V.M. Real-time adaptive background segmentation. In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, volume 3, pages 341–344, 2003. [doi:10.1109/ICME.2003.1221318](https://doi.org/10.1109/ICME.2003.1221318). 28, 103

- [31] Darren E. Butler, V. Michael Bove, and Sridha Sridharan. Real-time adaptive Foreground/Background segmentation. *EURASIP Journal on Advances in Signal Processing*, 2005(14):2292–2304, August 2005. [doi:10.1155/ASP.2005.2292](https://doi.org/10.1155/ASP.2005.2292). 28, 103
- [32] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 8(6):679–698, November 1986. [doi:10.1109/TPAMI.1986.4767851](https://doi.org/10.1109/TPAMI.1986.4767851). 54, 130, 280
- [33] H. Celik, A. Hanjalic, and E.A. Hendriks. Towards a robust solution to people counting. In *Image Processing, 2006 IEEE International Conference on*, pages 2401–2404, October 2006. [doi:10.1109/ICIP.2006.312946](https://doi.org/10.1109/ICIP.2006.312946). 34, 50, 51, 65, 105
- [34] A. B. Chan, M. Morrow, and N. Vasconcelos. Analysis of crowded scenes using holistic properties. In *Performance Evaluation of Tracking and Surveillance (PETS 2009), Eleventh IEEE International Workshop on*, pages 101–108, Miami, Florida, 2009. URL: http://www.cvg.rdg.ac.uk/PETS2009/PETS09_PROCEEDINGS.pdf. 36, 55, 56, 112
- [35] A. B. Chan and N. Vasconcelos. Counting people with low-level features and Bayesian regression. *Image Processing, IEEE Transactions on*, 21(4):2160–2177, April 2012. [doi:10.1109/TIP.2011.2172800](https://doi.org/10.1109/TIP.2011.2172800). 36, 55, 127, 153
- [36] 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*, pages 1–7, June 2008. URL: <http://www.svcl.ucsd.edu/projects/peoplecnt/>, [doi:10.1109/CVPR.2008.4587569](https://doi.org/10.1109/CVPR.2008.4587569). xv, xviii, xxv, xxvi, 19, 20, 34, 36, 48, 55, 68, 101, 104, 108, 109, 110, 114, 117, 118, 119, 127, 133, 136, 139, 144, 153, 157, 158, 162, 190, 193, 194, 199, 204, 213, 358

- [37] A.B. Chan and N. Vasconcelos. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(5):909–926, May 2008. [doi:10.1109/TPAMI.2007.70738](https://doi.org/10.1109/TPAMI.2007.70738). 19, 55, 56, 87, 130, 193, 362
- [38] A.B. Chan and N. Vasconcelos. Bayesian poisson regression for crowd counting. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 545–551, October 2009. [doi:10.1109/ICCV.2009.5459191](https://doi.org/10.1109/ICCV.2009.5459191). 36, 55, 162, 190, 194, 213
- [39] Chao-Ho Chen, Tsong-Yi Chen, Da-Jinn Wang, and Tsang-Jie Chen. A cost-effective people-counter for a crowd of moving people based on two-stage segmentation. *Journal of Information Hiding and Multimedia Signal Processing*, 3(1):12–23, January 2012. URL: <http://www.jihmsp.org/~jihmsp/2012/vol3/JIH-MSP-2012-01-002.pdf>. 69, 70, 190, 275
- [40] Duan-Yu Chen and Kuan-Yi Lin. A novel viewer counter for digital billboards. In *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP '09. Fifth International Conference on*, pages 653–656, September 2009. [doi:10.1109/IIH-MSP.2009.211](https://doi.org/10.1109/IIH-MSP.2009.211). 63
- [41] Duan-Yu Chen, Chih-Wen Su, Yi-Chong Zeng, Shih-Wei Sun, Wei-Ru Lai, and Hong-Yuan Mark Liao. An online people counting system for electronic advertising machines. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 1262–1265, 2009. [doi:10.1109/ICME.2009.5202731](https://doi.org/10.1109/ICME.2009.5202731). 63
- [42] Ke Chen, Chen Change Loy, Shaogang Gong, and Tony Xiang. Feature mining for localised crowd counting. In *Proceedings of the British Machine Vision Conference*, pages 21.1–21.11. British Machine Vision Association, 2012. URL: http://www.eecs.qmul.ac.uk/~ccloy/downloads_

- [mall_dataset.html](#), doi:10.5244/C.26.21. xviii, xxvii, 43, 66, 101, 104, 153, 157, 158, 162, 164, 197, 204, 358
- [43] Li Chen, Ji Tao, Yap-Peng Tan, and Kap-Luk Chan. People counting using iterative mean-shift fitting with symmetry measure. In *Information, Communications Signal Processing, 2007 6th International Conference on*, pages 1–4, December 2007. doi:10.1109/ICICS.2007.4449760. 62
- [44] Thou-Ho Chen. An automatic bi-directional passing-people counting method based on color image processing. In *Security Technology, 2003. Proceedings. IEEE 37th Annual 2003 International Carnahan Conference on*, pages 200–207, October 2003. doi:10.1109/CCST.2003.1297560. 69
- [45] Thou-Ho Chen, Tsong-Yi Chen, and Zhi-Xian Chen. An intelligent people-flow counting method for passing through a gate. In *Robotics, Automation and Mechatronics, 2006 IEEE Conference on*, pages 1–6, June 2006. doi:10.1109/RAMECH.2006.252623. 69
- [46] Tsong-Yi Chen, Chao-Ho Chen, Da-Jinn Wang, and Tsang-Jie Chen. Real-time counting method for a crowd of moving people. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*, pages 643–646, October 2010. doi:10.1109/IIHMSP.2010.163. 69
- [47] Tsong-Yi Chen, Chao-Ho Chen, Da-Jinn Wang, and Yi-Li Kuo. A people counting system based on face-detection. In *Genetic and Evolutionary Computing (ICGEC), 2010 Fourth International Conference on*, pages 699–702, December 2010. doi:10.1109/ICGEC.2010.178. 70, 275
- [48] Siu-Yeung Cho, T. W.S Chow, and Chi-Tat Leung. A neural-based crowd estimation by hybrid global learning algorithm. *Systems, Man and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 29(4):535–541, August 1999. doi:10.1109/3477.775269. 47, 48, 49

- [49] Siu-Yeung Cho and Tommy W. S. Chow. A fast neural learning vision system for crowd estimation at underground stations platform. *Neural Processing Letters*, 10(2):111–120, October 1999. [doi:10.1023/A:1018781301409](https://doi.org/10.1023/A:1018781301409). 34, 48, 49
- [50] T.W.S. Chow, J.Y.-F. Yam, and S.-Y Cho. Fast training algorithm for feed-forward neural networks: application to crowd estimation at underground stations. *Artificial Intelligence in Engineering*, 13(3):301–307, July 1999. [doi:10.1016/S0954-1810\(99\)00016-3](https://doi.org/10.1016/S0954-1810(99)00016-3). 48, 49
- [51] Robert T. Collins, Alan J. Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, Osamu Hasegawa, Peter Burt, and Lambert Wixson. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, The Robotics Institute, Carnegie Mellon University, Pittsburgh PA, 2000. URL: http://www.ri.cmu.edu/pub_files/pub2/collins_robert_2000_1/collins_robert_2000_1.pdf. 18, 75
- [52] D. Conte, P. Foggia, G. Percannella, F. Tufano, and M. Vento. Counting moving people in videos by salient points detection. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 1743–1746, August 2010. [doi:10.1109/ICPR.2010.431](https://doi.org/10.1109/ICPR.2010.431). 65, 104, 153
- [53] D. Conte, P. Foggia, G. Percannella, F. Tufano, and M. Vento. A method for counting people in crowded scenes. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pages 225–232, September 2010. [doi:10.1109/AVSS.2010.78](https://doi.org/10.1109/AVSS.2010.78). 64, 65
- [54] D. Conte, P. Foggia, G. Percannella, and M. Vento. A method based on the indirect approach for counting people in crowded scenes. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International*

- Conference on*, pages 111–118, September 2010. [doi:10.1109/AVSS.2010.86](https://doi.org/10.1109/AVSS.2010.86). 65
- [55] Donatello Conte, Pasquale Foggia, Gennaro Percannella, Francesco Tu-fano, and Mario Vento. A method for counting moving people in video surveillance videos. *EURASIP Journal on Advances in Signal Processing*, 2010(1):1–10, June 2010. [doi:10.1155/2010/231240](https://doi.org/10.1155/2010/231240). 65, 131, 190, 204, 358
- [56] R. Cutler and L.S. Davis. Robust real-time periodic motion detection, analysis, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):781–796, August 2000. [doi:10.1109/34.868681](https://doi.org/10.1109/34.868681). 18
- [57] 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*, volume 1, pages 886–893, June 2005. [doi:10.1109/CVPR.2005.177](https://doi.org/10.1109/CVPR.2005.177). 18, 54, 59, 110, 130
- [58] A. C Davies, Jia Hong Yin, and S. A Velastin. Crowd monitoring using image processing. *Electronics & Communication Engineering Journal*, 7(1):37–47, February 1995. [doi:10.1049/ecej:19950106](https://doi.org/10.1049/ecej:19950106). 2, 34, 35, 36, 37, 48, 49, 117, 124, 130, 144, 147, 162
- [59] S. Denman, C. Fookes, and S. Sridharan. Improved simultaneous computation of motion detection and optical flow for object tracking. In *Digital Image Computing: Techniques and Applications, 2009. DICTA '09.*, pages 175–182, December 2009. [doi:10.1109/DICTA.2009.35](https://doi.org/10.1109/DICTA.2009.35). 19, 30, 32, 33, 48, 49, 57, 103, 124, 219
- [60] Simon Denman, Vinod Chandran, and Sridha Sridharan. An adaptive optical flow technique for person tracking systems. *Pattern Recognition Letters*,

- 28(10):1232–1239, July 2007. [doi:10.1016/j.patrec.2007.02.008](https://doi.org/10.1016/j.patrec.2007.02.008). 19, 28, 32, 33, 48, 49, 103, 124, 219
- [61] Simon Paul Denman. *Improved Detection and Tracking of Objects in Surveillance Video*. PhD thesis, Queensland University of Technology (QUT), 2009. URL: <http://eprints.qut.edu.au/29328/>. xxiv, 29, 31, 32, 33, 49, 103, 124
- [62] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, April 2012. [doi:10.1109/TPAMI.2011.155](https://doi.org/10.1109/TPAMI.2011.155). 59
- [63] Lan Dong, V. Parameswaran, V. Ramesh, and I. Zoghalmi. Fast crowd segmentation using shape indexing. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, October 2007. [doi:10.1109/ICCV.2007.4409075](https://doi.org/10.1109/ICCV.2007.4409075). xxv, 61, 62, 73, 127
- [64] Nan Dong, Fuqiang Liu, and Zhipeng Li. Crowd density estimation using sparse texture features. *Journal of Convergence Information Technology*, 5(6):125–137, August 2010. [doi:10.4156/jcit.vol5.issue6.13](https://doi.org/10.4156/jcit.vol5.issue6.13). 66
- [65] M. Enzweiler and D.M. Gavrila. Monocular pedestrian detection: Survey and experiments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2179–2195, 2009. [doi:10.1109/TPAMI.2008.260](https://doi.org/10.1109/TPAMI.2008.260). 59
- [66] D. Fehr, R. Sivalingam, V. Morellas, N. Papanikolopoulos, O. Lotfallah, and Youngchoon Park. Counting people in groups. In *Advanced Video and Signal Based Surveillance, 2009. AVSS '09. Sixth IEEE International Conference on*, pages 152–157, September 2009. [doi:10.1109/AVSS.2009.55](https://doi.org/10.1109/AVSS.2009.55). 65, 105

-
- [67] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008. [doi:10.1109/CVPR.2008.4587597](https://doi.org/10.1109/CVPR.2008.4587597). 18, 59, 110
 - [68] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, September 2010. URL: <http://www.cs.brown.edu/~pff/latent-release4/>, [doi:10.1109/TPAMI.2009.167](https://doi.org/10.1109/TPAMI.2009.167). 18, 59, 110, 111, 115
 - [69] Clinton Fookes, Simon Denman, Ruan Lakemond, David Ryan, Sridha Sridharan, and Massimo Piccardi. Semi-supervised intelligent surveillance system for secure environments. In *Industrial Electronics (ISIE), 2010 IEEE International Symposium on*, pages 2815–2820, July 2010. [doi:10.1109/ISIE.2010.5636922](https://doi.org/10.1109/ISIE.2010.5636922). 14
 - [70] John Fox and Sanford Weisberg. Robust regression: Appendix to an r and s-PLUS companion to applied regression, January 2002. URL: <http://cran.r-project.org/doc/contrib/Fox-Companion/appendix-robust-regression.pdf>. xxxii, 367
 - [71] W. Ge and R.T. Collins. Marked point processes for crowd counting. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2913–2920, June 2009. [doi:10.1109/CVPR.2009.5206621](https://doi.org/10.1109/CVPR.2009.5206621). 61
 - [72] Weina Ge and Robert T. Collins. Crowd detection with a multiview sampler. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, volume 6315, pages 324–337.

- Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. [doi:10.1007/978-3-642-15555-0_24](https://doi.org/10.1007/978-3-642-15555-0_24). 61
- [73] Weina Ge, Robert T. Collins, and R. Barry Ruback. Vision-based analysis of small groups in pedestrian crowds. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(5):1003–1016, May 2012. [doi:10.1109/TPAMI.2011.176](https://doi.org/10.1109/TPAMI.2011.176). 61
- [74] Weina Ge and R.T. Collins. Evaluation of sampling-based pedestrian detection for crowd counting. In *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*, pages 1–7, December 2009. [doi:10.1109/PETS-WINTER.2009.5399553](https://doi.org/10.1109/PETS-WINTER.2009.5399553). 61
- [75] Weina Ge and R.T. Collins. Crowd density analysis with marked point processes [applications corner]. *Signal Processing Magazine, IEEE*, 27(5):107–123, September 2010. [doi:10.1109/MSP.2010.937495](https://doi.org/10.1109/MSP.2010.937495). 61
- [76] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361, June 2012. [doi:10.1109/CVPR.2012.6248074](https://doi.org/10.1109/CVPR.2012.6248074). 25, 154
- [77] Graeme Gerrard and Richard Thompson. Two million cameras in the UK. *CCTVImage*, 42:10–12, 2011. 3
- [78] Donald Gross, John F. Shortle, James M. Thompson, and Carl M. Harris. *Fundamentals of Queueing Theory*. Wiley, Hoboken, 4 edition, 2013. 71, 72
- [79] R.M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, May 1979. [doi:10.1109/PROC.1979.11328](https://doi.org/10.1109/PROC.1979.11328). 38, 39, 66, 323

- [80] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: real-time surveillance of people and their activities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):809–830, August 2000. [doi:10.1109/34.868683](https://doi.org/10.1109/34.868683). 75
- [81] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference, Proceedings of the*, pages 23.1–23.6. Alvey Vision Club, 1988. [doi:10.5244/C.2.23](https://doi.org/10.5244/C.2.23). 65, 131
- [82] W. K. Hastings. Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970. [doi:10.2307/2334940](https://doi.org/10.2307/2334940). 60
- [83] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, May 1995. [doi:10.1103/PhysRevE.51.4282](https://doi.org/10.1103/PhysRevE.51.4282). 82
- [84] Dirk Helbing and Pratik Mukerji. Crowd disasters as systemic failures: analysis of the love parade disaster. *EPJ Data Science*, 1(1):1–40, December 2012. [doi:10.1140/epjds7](https://doi.org/10.1140/epjds7). 99
- [85] Honeywell. Honeywell video systems, 2012. URL: <http://www.honeywellvideo.com>. 3
- [86] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, August 1981. [doi:10.1016/0004-3702\(81\)90024-2](https://doi.org/10.1016/0004-3702(81)90024-2). 21, 22, 23, 24, 26, 33, 327
- [87] Ya-li Hou and G.K.H. Pang. Automated people counting at a mass site. In *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, pages 464–469, September 2008. [doi:10.1109/ICAL.2008.4636196](https://doi.org/10.1109/ICAL.2008.4636196). 36, 52

- [88] Ya-Li Hou and G.K.H. Pang. People counting and human detection in a challenging situation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(1):24–33, January 2011. [doi:10.1109/TSMCA.2010.2064299](https://doi.org/10.1109/TSMCA.2010.2064299). 36, 52
- [89] Wei-Lieh Hsu, Kun-Fong Lin, and Chang-Lung Tsai. Crowd density estimation based on frequency analysis. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2011 Seventh International Conference on*, pages 348–351, October 2011. [doi:10.1109/IIHMSP.2011.49](https://doi.org/10.1109/IIHMSP.2011.49). 57
- [90] Weiming Hu, Tieniu Tan, Liang Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352, August 2004. [doi:10.1109/TSMCC.2004.829274](https://doi.org/10.1109/TSMCC.2004.829274). 3, 19, 20
- [91] Weiming Hu, Xuejuan Xiao, Zhouyu Fu, D. Xie, Tieniu Tan, and S. Maybank. A system for learning statistical motion patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(9):1450–1464, September 2006. [doi:10.1109/TPAMI.2006.176](https://doi.org/10.1109/TPAMI.2006.176). 320
- [92] Weiming Hu, Dan Xie, and Tieniu Tan. A hierarchical self-organizing approach for learning the patterns of motion trajectories. *Neural Networks, IEEE Transactions on*, 15(1):135–144, January 2004. [doi:10.1109/TNN.2003.820668](https://doi.org/10.1109/TNN.2003.820668). 19, 76
- [93] D. Huang, T.W.S. Chow, and W.N. Chau. Neural network based system for counting people. In *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, volume 3, pages 2197–2201, November 2002. [doi:10.1109/IECON.2002.1185313](https://doi.org/10.1109/IECON.2002.1185313). 36, 48, 49
- [94] Norhaida Hussain, Halimatul Saadiah Md. Yatim, Nor Liza Hussain, Jasy Liew Suet Yan, and Fazilah Haron. CDES: a pixel-based crowd density

- estimation system for masjid al-haram. *Safety Science*, 49(6):824–833, July 2011. [doi:10.1016/j.ssci.2011.01.005](https://doi.org/10.1016/j.ssci.2011.01.005). 52
- [95] C.R. Jung, J.C.S. Jacques, J. Soldera, and S.R. Musse. Detection of unusual motion using computer vision. In *Computer Graphics and Image Processing, 2006. SIBGRAPI '06. 19th Brazilian Symposium on*, pages 349–356, October 2006. [doi:10.1109/SIBGRAPI.2006.11](https://doi.org/10.1109/SIBGRAPI.2006.11). 76
- [96] P. Kilambi, O. Masoud, and N. Papanikolopoulos. Crowd analysis at mass transit sites. In *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pages 753–758, September 2006. [doi:10.1109/ITSC.2006.1706832](https://doi.org/10.1109/ITSC.2006.1706832). 65, 105
- [97] Prahlad Kilambi, Evan Ribnick, Ajay J. Joshi, Osama Masoud, and Niko-laos Papanikolopoulos. Estimating pedestrian counts in groups. *Computer Vision and Image Understanding*, 110(1):43–59, April 2008. [doi:10.1016/j.cviu.2007.02.003](https://doi.org/10.1016/j.cviu.2007.02.003). 34, 64, 65, 105, 149
- [98] Byeoung-su Kim, Gwang-Gook Lee, Ja-Young Yoon, Jae-Jun Kim, and Whoi-Yul Kim. A method of counting pedestrians in crowded scenes. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, volume 5227 of *Lecture Notes in Computer Science*, pages 1117–1126. Springer Berlin / Heidelberg, 2008. [doi:10.1007/978-3-540-85984-0_134](https://doi.org/10.1007/978-3-540-85984-0_134). 20, 69, 70, 275, 280
- [99] Jae-won Kim, Kang-sun Choi, Byeong-doo Choi, and Sung-jea Ko. Real-time vision-based people counting system for the security door. *Circuits Systems Computers and Communications, International Technical Conference on*, 3672:1416–1419, 2002. 69
- [100] Jaechul Kim and K. Grauman. Observe locally, infer globally: A space-time MRF for detecting abnormal activities with incremental updates. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference*

- on*, pages 2921–2928, June 2009. [doi:10.1109/CVPR.2009.5206569](https://doi.org/10.1109/CVPR.2009.5206569). xxv, 79, 84, 87, 88, 348, 350
- [101] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Artificial Intelligence, Proceedings of the 14th International Joint Conference on*, volume 2, pages 1137–1143, 1995. 160
- [102] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990. [doi:10.1109/5.58325](https://doi.org/10.1109/5.58325). 40
- [103] D. Kong, D. Gray, and Hai Tao. A viewpoint invariant approach for crowd counting. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 1187–1190, 2006. [doi:10.1109/ICPR.2006.197](https://doi.org/10.1109/ICPR.2006.197). 20, 34, 36, 37, 52, 54, 117, 130, 131, 144, 162, 190, 191, 204, 358
- [104] Dan Kong, Doug Gray, and Hai Tao. Counting pedestrians in crowds using viewpoint invariant training. In *British Machine Vision Conference, Proceedings of the*, pages 63.1–63.10. BMVA Press, 2005. [doi:10.5244/C.19.63](https://doi.org/10.5244/C.19.63). 36, 52, 53, 100, 190, 191
- [105] N. Krahnstoever and P.R.S. Mendonca. Bayesian autocalibration for surveillance. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1858–1865, 2005. [doi:10.1109/ICCV.2005.44](https://doi.org/10.1109/ICCV.2005.44). 210
- [106] L. Kratz and K. Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1446–1453, June 2009. [doi:10.1109/CVPR.2009.5206771](https://doi.org/10.1109/CVPR.2009.5206771). 18, 79, 82, 83, 84, 320, 332

- [107] Louis Kratz and Ko Nishino. Spatio-temporal motion pattern modeling of extremely crowded scenes. In *Machine Learning for Vision-based Motion Analysis - MLVMA '08, The 1st International Workshop on*, 2008. URL: <http://hal.inria.fr/inria-00326717>. 18, 82, 83
- [108] Louis Kratz and Ko Nishino. Spatio-temporal motion pattern models of extremely crowded scenes. In Liang Wang, Guoying Zhao, Li Cheng, and Matti Pietikäinen, editors, *Machine Learning for Vision-Based Motion Analysis, Advances in Pattern Recognition*, pages 263–274. Springer London, January 2011. doi:[10.1007/978-0-85729-057-1_10](https://doi.org/10.1007/978-0-85729-057-1_10). 18, 82
- [109] Gwang-Gook Lee, Byeoung-su Kim, and Whoi-Yul Kim. Automatic estimation of pedestrian flow. In *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, pages 291–296, September 2007. doi:[10.1109/ICDSC.2007.4357536](https://doi.org/10.1109/ICDSC.2007.4357536). 20, 69, 70
- [110] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In *Advances in Neural Information Processing Systems (NIPS)*, volume 23, pages 1324–1332, December 2010. URL: http://books.nips.cc/papers/files/nips23/NIPS2010_0330.pdf. 66, 104, 137, 190, 193, 204, 358
- [111] V. Leung, A. Colombo, J. Orwell, and S.A. Velastin. Modelling periodic scene elements for visual surveillance. *IET Computer Vision*, 2(2):88–98, 2008. doi:[10.1049/iet-cvi:20070070](https://doi.org/10.1049/iet-cvi:20070070). 32
- [112] Daw-Tung Lin and Li-Wei Liu. Real-time detection of passing objects using virtual gate and motion vector analysis. In Frode Eika Sandnes, Yan Zhang, Chunming Rong, Laurence T. Yang, and Jianhua Ma, editors, *Ubiquitous Intelligence and Computing*, number 5061 in Lecture Notes in Computer Science, pages 710–719. Springer Berlin Heidelberg, January 2008. doi:[10.1007/978-3-540-69293-5_56](https://doi.org/10.1007/978-3-540-69293-5_56). 70

- [113] Sheng-Fuu Lin, Jaw-Yeh Chen, and Hung-Xin Chao. Estimation of number of people in crowded scenes using perspective transformation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 31(6):645–654, November 2001. [doi:10.1109/3468.983420](https://doi.org/10.1109/3468.983420). 35, 63
- [114] Alan J. Lipton. Local application of optic flow to analyse rigid versus non-rigid motion. In *Frame-Rate Vision, IEEE Workshop on*, pages 1–9, 1999. 18
- [115] X. Liu, P.H. Tu, J. Rittscher, A. Perera, and N. Krahnstoever. Detecting and counting people in surveillance applications. In *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*, pages 306–311, September 2005. [doi:10.1109/AVSS.2005.1577286](https://doi.org/10.1109/AVSS.2005.1577286). 62
- [116] Jiangung Lou, Qifeng Liu, Tieniu Tan, and Weiming Hu. Semantic interpretation of object activities in a surveillance system. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 777–780, 2002. [doi:10.1109/ICPR.2002.1048115](https://doi.org/10.1109/ICPR.2002.1048115). 76
- [117] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Artificial Intelligence, Proceedings of the 7th International Joint Conference on*, volume 2 of *IJCAI'81*, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. 22, 33, 81, 281, 327
- [118] P Luff, C Heath, and M Jirotka. Surveying the scene: technologies for everyday awareness and monitoring in control rooms. *Interacting with Computers*, 13(2):193–228, December 2000. [doi:10.1016/S0953-5438\(00\)00038-2](https://doi.org/10.1016/S0953-5438(00)00038-2). 2
- [119] Fengjun Lv, Tao Zhao, and R. Nevatia. Self-calibration of a camera from video of a walking human. In *Pattern Recognition, 2002. Proceedings. 16th*

- International Conference on*, volume 1, pages 562–567, 2002. [doi:10.1109/ICPR.2002.1044793](https://doi.org/10.1109/ICPR.2002.1044793). 210
- [120] Fengjun Lv, Tao Zhao, and R. Nevatia. Camera calibration from video of a walking human. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(9):1513–1518, September 2006. [doi:10.1109/TPAMI.2006.178](https://doi.org/10.1109/TPAMI.2006.178). 210
- [121] R. Ma, L. Li, W. Huang, and Q. Tian. On pixel count based crowd density estimation for visual surveillance. In *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, volume 1, pages 170–173, December 2004. [doi:10.1109/ICCIS.2004.1460406](https://doi.org/10.1109/ICCIS.2004.1460406). xxv, 34, 36, 50, 51, 52, 66, 106, 107, 110, 114, 117, 124, 213
- [122] Wenhua Ma, Lei Huang, and Changping Liu. Advanced local binary pattern descriptors for crowd estimation. In *Computational Intelligence and Industrial Application, 2008. PACIIA '08. Pacific-Asia Workshop on*, volume 2, pages 958–962, December 2008. [doi:10.1109/PACIIA.2008.258](https://doi.org/10.1109/PACIIA.2008.258). 66
- [123] Wenhua Ma, Lei Huang, and Changping Liu. Crowd estimation using multi-scale local texture analysis and confidence-based soft classification. In *Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on*, volume 1, pages 142–146, December 2008. [doi:10.1109/IITA.2008.303](https://doi.org/10.1109/IITA.2008.303). 64, 66
- [124] Wenhua Ma, Lei Huang, and Changping Liu. Crowd density analysis using co-occurrence texture features. In *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*, pages 170–175, December 2010. [doi:10.1109/ICCIT.2010.5711051](https://doi.org/10.1109/ICCIT.2010.5711051). 66
- [125] V. Mahadevan, Weixin Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In *Computer Vision and Pattern Recognition*

- (CVPR), 2010 IEEE Conference on, pages 1975–1981, June 2010. [doi:10.1109/CVPR.2010.5539872](https://doi.org/10.1109/CVPR.2010.5539872). xxii, xxxii, 20, 87, 88, 320, 343, 347, 348, 349, 350, 351, 352, 361
- [126] A. Marana, M. Cavenaghi, R. Ulson, and F. Drumond. Real-time crowd density estimation using images. In *Advances in Visual Computing*, volume 3804 of *Lecture Notes in Computer Science*, pages 355–362. Springer Berlin / Heidelberg, 2005. [doi:10.1007/11595755_43](https://doi.org/10.1007/11595755_43). 36, 37
- [127] A. N Marana, L. F Costa, R. A Lotufo, and S. A Velastin. On the efficacy of texture analysis for crowd monitoring. In *International Symposium on Computer Graphics, Image Processing, and Vision, 1998. Proceedings. SIBGRAPI '98*, pages 354–361. IEEE, October 1998. [doi:10.1109/SIBGRA.1998.722773](https://doi.org/10.1109/SIBGRA.1998.722773). 36
- [128] A. N Marana, S. A Velastin, L. F Costa, and R. A Lotufo. Estimation of crowd density using image processing. In *Image Processing for Security Applications (Digest No.: 1997/074), IEEE Colloquium on*, pages 11/1–11/8. IET, March 1997. [doi:10.1049/ic:19970387](https://doi.org/10.1049/ic:19970387). 18, 34, 36, 37, 40, 45, 49, 66, 117, 144
- [129] A.N. Marana, L. Da Fontoura Costa, R.A. Lotufo, and S.A. Velastin. Estimating crowd density with minkowski fractal dimension. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings. 1999 IEEE International Conference on*, volume 6, pages 3521–3524, March 1999. [doi:10.1109/ICASSP.1999.757602](https://doi.org/10.1109/ICASSP.1999.757602). 34, 36, 40, 42, 45, 49, 117
- [130] A.N. Marana, S.A. Velastin, L.F. Costa, and R.A. Lotufo. Automatic estimation of crowd density using texture. *Safety Science*, 28(3):165–175, April 1998. [doi:10.1016/S0925-7535\(97\)00081-7](https://doi.org/10.1016/S0925-7535(97)00081-7). 36, 326
- [131] O. Masoud and N. P Papanikolopoulos. A novel method for tracking

- and counting pedestrians in real-time using a single camera. *Vehicular Technology, IEEE Transactions on*, 50(5):1267–1278, September 2001. [doi:10.1109/25.950328](https://doi.org/10.1109/25.950328). 34, 62, 147
- [132] MATLAB. *version 7.14.0.739 (R2012a)*. The MathWorks Inc., Natick, Massachusetts, 2012. xxxiii, 368
- [133] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 935–942, June 2009. [doi:10.1109/CVPR.2009.5206641](https://doi.org/10.1109/CVPR.2009.5206641). 79, 82, 88, 320, 348, 350
- [134] Djamel Merad, Kheir-Eddine Aziz, and Nicolas Thome. Fast people counting using head detection from skeleton graph. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pages 151–156, September 2010. [doi:10.1109/AVSS.2010.91](https://doi.org/10.1109/AVSS.2010.91). 63
- [135] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. [doi:10.1063/1.1699114](https://doi.org/10.1063/1.1699114). 60
- [136] Anton Milan. Data, 2012. URL: <http://www.gris.informatik.tu-darmstadt.de/~aandriye/data.html>. 156
- [137] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2–3):90–126, November 2006. [doi:10.1016/j.cviu.2006.08.002](https://doi.org/10.1016/j.cviu.2006.08.002). 3
- [138] Vinod Nair and James J. Clark. Automated visual surveillance using hidden Markov models. In *Vision Interface, 15th International Conference on*, pages 88–93, 2002. 76, 320

- [139] Hajananth Nallaivarothayan, David Ryan, Simon Denman, Sridha Sridharan, and Clinton Fookes. Anomalous event detection using a semi-two dimensional hidden Markov model. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*, pages 1–7, 2012. [doi:10.1109/DICTA.2012.6411711](https://doi.org/10.1109/DICTA.2012.6411711). 14
- [140] Hajananth Nallaivarothayan, David Ryan, Simon Denman, Sridha Sridharan, Clinton Fookes, and Andry Rakotonirainy. Detecting anomalous events at railway level crossings. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 227(5):539–553, September 2013. [doi:10.1177/0954409713501296](https://doi.org/10.1177/0954409713501296). 13
- [141] N.T. Nguyen, D.Q. Phung, S. Venkatesh, and H. Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 955–960, June 2005. [doi:10.1109/CVPR.2005.203](https://doi.org/10.1109/CVPR.2005.203). 76
- [142] Clive Norris, Mike McCahill, and David Wood. The growth of CCTV: a global perspective on the international diffusion of video surveillance in publicly accessible space. *Surveillance & Society*, 2(2/3), September 2002. 3
- [143] N.M. Oliver, B. Rosario, and A.P. Pentland. A Bayesian computer vision system for modeling human interactions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):831–843, August 2000. [doi:10.1109/34.868684](https://doi.org/10.1109/34.868684). 78
- [144] N. Paragios and V. Ramesh. A MRF-based approach for real-time subway monitoring. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. IEEE Conference on*, volume 1, pages 1034–1040. IEEE, 2001. [doi:10.1109/CVPR.2001.990644](https://doi.org/10.1109/CVPR.2001.990644). 34, 36, 50, 66, 73, 213

- [145] Vasu Parameswaran, Vinay Shet, and Visvanathan Ramesh. Design and validation of a system for people queue statistics estimation. In Caifeng Shan, Fatih Porikli, Tao Xiang, and Shaogang Gong, editors, *Video Analytics for Business Intelligence*, number 409 in Studies in Computational Intelligence, pages 355–373. Springer Berlin Heidelberg, January 2012. [doi:10.1007/978-3-642-28598-1_11](https://doi.org/10.1007/978-3-642-28598-1_11). 73
- [146] Park, Sangho, Aggarwal, and J. A hierarchical Bayesian network for event recognition of human actions and interactions. *Multimedia Systems*, 10(2):164–179, August 2004. [doi:10.1007/s00530-004-0148-1](https://doi.org/10.1007/s00530-004-0148-1). 78
- [147] M. Patzold, R.H. Evangelio, and T. Sikora. Counting people in crowded environments by fusion of shape and motion information. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pages 157–164, September 2010. [doi:10.1109/AVSS.2010.92](https://doi.org/10.1109/AVSS.2010.92). 63
- [148] PETS. Ninth IEEE international workshop on performance evaluation of tracking and surveillance, 2006. URL: <http://www.cvg.rdg.ac.uk/PETS2006/>. xix, xxxviii, 39, 206, 207, 216, 218, 220, 242
- [149] PETS. Eleventh IEEE international workshop on performance evaluation of tracking and surveillance, 2009. URL: <http://www.cvg.rdg.ac.uk/PETS2009/>. xix, xxvi, xxxviii, 41, 68, 70, 101, 110, 111, 113, 139, 153, 156, 158, 206, 207, 216, 220, 242, 284, 359
- [150] PETS. Fourteenth IEEE international workshop on performance evaluation of tracking and surveillance, 2012. URL: <http://www.cvg.rdg.ac.uk/PETS2012/>. xxxviii
- [151] V. Rabaud and S. Belongie. Counting crowded moving objects. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference*

- on*, volume 1, pages 705–711, June 2006. [doi:10.1109/CVPR.2006.92](https://doi.org/10.1109/CVPR.2006.92). 19, 62, 104
- [152] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989. [doi:10.1109/5.18626](https://doi.org/10.1109/5.18626). 91, 93, 340
- [153] H. Rahmalan, M.S. Nixon, and J.N. Carter. On crowd density estimation for surveillance. In *Crime and Security, 2006. The Institution of Engineering and Technology Conference on*, pages 540–545, June 2006. URL: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4123816>. 18, 34, 36, 40, 42, 44, 49, 117
- [154] Hidayah Rahmalan. *Application of invariant moments for crowd analysis*. PhD thesis, University of Southampton, January 2010. URL: <http://eprints.soton.ac.uk/156895/>. 42
- [155] Carl Edward Rasmussen. Gaussian processes in machine learning. In Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch, editors, *Advanced Lectures on Machine Learning*, number 3176 in Lecture Notes in Computer Science, pages 63–71. Springer Berlin Heidelberg, January 2004. [doi:10.1007/978-3-540-28650-9_4](https://doi.org/10.1007/978-3-540-28650-9_4). 145
- [156] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006. URL: <http://www.gaussianprocess.org/gpml/>. 97, 145
- [157] C. S Regazzoni, A. Tesei, and V. Murino. A real-time vision system for crowding monitoring. In *Industrial Electronics, Control, and Instrumentation, 1993. Proceedings of the IECON '93., International Conference on*, volume 3, pages 1860–1864. IEEE, November 1993. [doi:10.1109/IECON.1993.339357](https://doi.org/10.1109/IECON.1993.339357). 34, 36, 46, 47, 152, 166, 191, 227

- [158] C.S. Regazzoni, A. Tesei, and G. Vernazza. A Bayesian network for automatic visual crowding estimation in underground stations. In *Image Technology: Advances in Image Processing, Multimedia and Machine Vision*, pages 203–230. Springer, 1996. [doi:10.1007/978-3-642-58288-2_7](https://doi.org/10.1007/978-3-642-58288-2_7). 46
- [159] J. Rittscher, P.H. Tu, and N. Krahnstoever. Simultaneous estimation of segmentation and shape. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 486–493, June 2005. [doi:10.1109/CVPR.2005.323](https://doi.org/10.1109/CVPR.2005.323). 62
- [160] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):105–119, January 2010. [doi:10.1109/TPAMI.2008.275](https://doi.org/10.1109/TPAMI.2008.275). xxxvii, 132, 281
- [161] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 430–443. Springer Berlin / Heidelberg, 2006. [doi:10.1007/11744023_34](https://doi.org/10.1007/11744023_34). 281
- [162] David Ryan, Simon Denman, Clinton Fookes, and Sridha Sridharan. Scene invariant crowd counting for real-time surveillance. In *Signal Processing and Communication Systems, 2008. ICSPCS 2008. 2nd International Conference on*, pages 1–7, December 2008. [doi:10.1109/ICSPCS.2008.4813759](https://doi.org/10.1109/ICSPCS.2008.4813759). 14
- [163] David Ryan, Simon Denman, Clinton Fookes, and Sridha Sridharan. Crowd counting using multiple local features. In *Digital Image Computing: Techniques and Applications, 2009. DICTA '09.*, pages 81–88, December 2009. [doi:10.1109/DICTA.2009.22](https://doi.org/10.1109/DICTA.2009.22). 13, 144, 147

- [164] David Ryan, Simon Denman, Clinton Fookes, and Sridha Sridharan. Crowd counting using group tracking and local features. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pages 218–224, September 2010. [doi:10.1109/AVSS.2010.30](https://doi.org/10.1109/AVSS.2010.30). 13, 34, 144, 149
- [165] David Ryan, Simon Denman, Clinton Fookes, and Sridha Sridharan. Textures of optical flow for real-time anomaly detection in crowds. In *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, pages 230–235, September 2011. [doi:10.1109/AVSS.2011.6027327](https://doi.org/10.1109/AVSS.2011.6027327). 14
- [166] David Ryan, Simon Denman, Clinton Fookes, and Sridha Sridharan. Scene invariant multi camera crowd counting. *Pattern Recognition Letters*, 2013. In press. [doi:10.1016/j.patrec.2013.10.002](https://doi.org/10.1016/j.patrec.2013.10.002). 12
- [167] David Ryan, Simon Denman, Sridha Sridharan, and Clinton Fookes. Scene invariant crowd counting. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*, pages 237–242, December 2011. [doi:10.1109/DICTA.2011.46](https://doi.org/10.1109/DICTA.2011.46). 14
- [168] David Ryan, Simon Denman, Sridha Sridharan, and Clinton Fookes. Scene invariant crowd counting and crowd occupancy analysis. In Caifeng Shan, Fatih Porikli, Tao Xiang, and Shaogang Gong, editors, *Video Analytics for Business Intelligence*, number 409 in Studies in Computational Intelligence, pages 161–198. Springer Berlin Heidelberg, January 2012. [doi:10.1007/978-3-642-28598-1_6](https://doi.org/10.1007/978-3-642-28598-1_6). 13
- [169] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, April 2002. [doi:10.1023/A:1014573219977](https://doi.org/10.1023/A:1014573219977). 154

- [170] A.J. Schofield, P.A. Mehta, and T.J. Stonham. A system for counting people in video images using neural networks to identify the background scene. *Pattern Recognition*, 29(8):1421–1428, August 1996. [doi:10.1016/0031-3203\(95\)00163-8](https://doi.org/10.1016/0031-3203(95)00163-8). 63
- [171] A.J. Schofield, T.J. Stonham, and P.A. Mehta. Automated people counting to aid lift control. *Automation in Construction*, 6(5–6):437–445, September 1997. [doi:10.1016/S0926-5805\(97\)00022-8](https://doi.org/10.1016/S0926-5805(97)00022-8). 63
- [172] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, March 1978. [doi:10.1214/aos/1176344136](https://doi.org/10.1214/aos/1176344136). 342
- [173] H. Septian, Ji Tao, and Yap-Peng Tan. People counting by video segmentation and tracking. In *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on*, pages 1–4, December 2006. [doi:10.1109/ICARCV.2006.345396](https://doi.org/10.1109/ICARCV.2006.345396). 70, 275
- [174] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994. [doi:10.1109/CVPR.1994.323794](https://doi.org/10.1109/CVPR.1994.323794). xxxvii, 18, 62
- [175] O. Sidla, Y. Lypetskyy, N. Brandle, and S. Seer. Pedestrian detection and tracking for counting applications in crowded situations. In *Advanced Video and Signal Based Surveillance, 2006. AVSS '06. IEEE International Conference on*, pages 70–75, November 2006. [doi:10.1109/AVSS.2006.91](https://doi.org/10.1109/AVSS.2006.91). 70, 275
- [176] Kevin Smith, Pedro Quelhas, and Daniel Gatica-perez. Detecting abandoned luggage items in a public space. In *Performance Evaluation of Tracking and Surveillance (PETS 2006), Ninth IEEE International Workshop on*, pages 75–82, 2006. URL: http://www.cvg.rdg.ac.uk/PETS2006/PETS2006_PROCEEDINGS.pdf. 317

- [177] S. Soatto, G. Doretto, and Ying Nian Wu. Dynamic textures. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 439–446, 2001. [doi:10.1109/ICCV.2001.937658](https://doi.org/10.1109/ICCV.2001.937658).
55
- [178] C. 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*, volume 2, pages 246–252, 1999. [doi:10.1109/CVPR.1999.784637](https://doi.org/10.1109/CVPR.1999.784637). 18, 26, 33, 49, 103, 219
- [179] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):747–757, August 2000. [doi:10.1109/34.868677](https://doi.org/10.1109/34.868677). 19, 26, 75, 103, 320
- [180] Deqing Sun, S. Roth, and M.J. Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439, June 2010. [doi:10.1109/CVPR.2010.5539939](https://doi.org/10.1109/CVPR.2010.5539939). 25, 327
- [181] Ben Tan, Junping Zhang, and Liang Wang. Semi-supervised elastic net for pedestrian counting. *Pattern Recognition*, 44(10–11):2297–2304, October 2011. URL: http://www.iipl.fudan.edu.cn/~zhangjp/Dataset/fd_pede_dataset_intro.htm, [doi:10.1016/j.patcog.2010.10.002](https://doi.org/10.1016/j.patcog.2010.10.002). 57, 101, 153, 157, 158, 162
- [182] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, December 2006. [doi:10.1198/016214506000000302](https://doi.org/10.1198/016214506000000302). 79
- [183] T. Teixeira and A. Savvides. Lightweight people counting and localizing in indoor spaces using camera sensor nodes. In *Distributed Smart Cameras*,

2007. *ICDSC '07. First ACM/IEEE International Conference on*, pages 36–43, September 2007. [doi:10.1109/ICDSC.2007.4357503](https://doi.org/10.1109/ICDSC.2007.4357503). 63
- [184] K. Terada, D. Yoshida, S. Oe, and J. Yamaguchi. A method of counting the passing people by using the stereo images. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, volume 2, pages 338–342, 1999. [doi:10.1109/ICIP.1999.822913](https://doi.org/10.1109/ICIP.1999.822913). 69
- [185] M.E. Tipping and C.M. Bishop. Mixtures of principal component analyzers. In *Artificial Neural Networks, Fifth International Conference on (Conf. Publ. No. 440)*, pages 13–18, 1997. [doi:10.1049/cp:19970694](https://doi.org/10.1049/cp:19970694). 85, 86
- [186] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, School of Computer Science, Carnegie Mellon University, 1991. URL: http://www.ri.cmu.edu/pub_files/pub2/tomasi_c_1991_1/tomasi_c_1991_1.pdf. xxxvii, 18, 62, 281
- [187] Fuan Tsai, Chun-Kai Chang, Jian-Yeo Rau, Tang-Huang Lin, and Gin-Ron Liu. 3D computation of gray level co-occurrence in hyperspectral image cubes. In Alan L. Yuille, Song-Chun Zhu, Daniel Cremers, and Yongtian Wang, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, number 4679 in Lecture Notes in Computer Science, pages 429–440. Springer Berlin Heidelberg, January 2007. [doi:10.1007/978-3-540-74198-5_33](https://doi.org/10.1007/978-3-540-74198-5_33). 323
- [188] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *Robotics and Automation, IEEE Journal of*, 3(4):323–344, August 1987. [doi:10.1109/JRA.1987.1087109](https://doi.org/10.1109/JRA.1987.1087109). 17, 207
- [189] R.Y. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. *Computer Vision and Pattern Recognition (CVPR'86)*,

- IEEE Conference on*, pages 364–374, 1986. [xxviii](#), [17](#), [207](#), [208](#), [210](#), [211](#), [244](#)
- [190] S. Velipasalar, Ying-Li Tian, and A. Hampapur. Automatic counting of interacting people by using a single uncalibrated camera. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 1265–1268, July 2006. [doi:10.1109/ICME.2006.262768](#). 69
- [191] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. IEEE Conference on*, volume 1, pages 511–518, 2001. [doi:10.1109/CVPR.2001.990517](#). 18, 336
- [192] L. Wang and N.H.C. Yung. Crowd counting and segmentation in visual surveillance. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 2573–2576, November 2009. [doi:10.1109/ICIP.2009.5413919](#). 320
- [193] Xiaogang Wang, Xiaoxu Ma, and W.E.L. Grimson. Unsupervised activity perception in crowded and complicated scenes using hierarchical Bayesian models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(3):539–555, March 2009. [doi:10.1109/TPAMI.2008.87](#). 79, 80, 88
- [194] Xinyu Wu, Guoyuan Liang, Ka Keung Lee, and Yangsheng Xu. Crowd density estimation using texture analysis and learning. In *Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on*, pages 214–219, December 2006. [doi:10.1109/ROBIO.2006.340379](#). 34, 66
- [195] Xinyu Wu, Yongsheng Ou, Huihuan Qian, and Yangsheng Xu. A detection system for human abnormal behavior. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1204–1208. IEEE, August 2005. [doi:10.1109/IROS.2005.1545205](#). 78

-
- [196] Tao Xiang and Shaogang Gong. Incremental and adaptive abnormal behaviour detection. *Computer Vision and Image Understanding*, 111(1):59–73, July 2008. [doi:10.1016/j.cviu.2007.06.004](https://doi.org/10.1016/j.cviu.2007.06.004). 77
 - [197] Li Xiaohua, Shen Lansun, and Li Huanqin. Estimation of crowd density based on wavelet and support vector machine. *Transactions of the Institute of Measurement and Control*, 28(3):299–308, 2006. [doi:10.1191/0142331206tim178oa](https://doi.org/10.1191/0142331206tim178oa). xxiv, xxv, 36, 37, 40, 42, 44, 46, 49, 117
 - [198] Tao Yang, Yanning Zhang, Dapei Shao, and Ying Li. Clustering method for counting passengers getting in a bus with single camera. *Optical Engineering*, 49(3):037203–1 – 037203–10, March 2010. [doi:doi:10.1117/1.3374439](https://doi.org/10.1117/1.3374439). 70, 275
 - [199] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):1–45, December 2006. [doi:10.1145/1177352.1177355](https://doi.org/10.1145/1177352.1177355). 18
 - [200] Junping Zhang, Ben Tan, Fei Sha, and Li He. Predicting pedestrian counts in crowded scenes with rich and high-dimensional features. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4):1037–1046, December 2011. [doi:10.1109/TITS.2011.2132759](https://doi.org/10.1109/TITS.2011.2132759). 57
 - [201] Xiaowei Zhang and G. Sexton. Automatic human head location for pedestrian counting. In *Image Processing and Its Applications, 1997. Sixth International Conference on*, volume 2, pages 535–540, July 1997. [doi:10.1049/cp:19970951](https://doi.org/10.1049/cp:19970951). 63
 - [202] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, November 2000. [doi:10.1109/34.888718](https://doi.org/10.1109/34.888718). 207

- [203] Tao Zhao and R. Nevatia. Bayesian human segmentation in crowded situations. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages 459–466, 2003. [doi:10.1109/CVPR.2003.1211503](https://doi.org/10.1109/CVPR.2003.1211503). xxv, 59, 60
- [204] Tao Zhao, R. Nevatia, and Bo Wu. Segmentation and tracking of multiple humans in crowded environments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(7):1198–1211, July 2008. [doi:10.1109/TPAMI.2007.70770](https://doi.org/10.1109/TPAMI.2007.70770). 73
- [205] Xi Zhao, E. Delleandrea, and Liming Chen. A people counting system based on face detection and tracking in a video. In *Advanced Video and Signal Based Surveillance, 2009. AVSS '09. Sixth IEEE International Conference on*, pages 67–72, September 2009. [doi:10.1109/AVSS.2009.45](https://doi.org/10.1109/AVSS.2009.45). 70, 275
- [206] Bolei Zhou, Xiaogang Wang, and Xiaoou Tang. Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2871–2878, June 2012. URL: <http://www.ee.cuhk.edu.hk/~xgwang/grandcentral.html>, [doi:10.1109/CVPR.2012.6248013](https://doi.org/10.1109/CVPR.2012.6248013). 101, 153, 157, 158, 162
- [207] Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31, 2004. [doi:10.1109/ICPR.2004.1333992](https://doi.org/10.1109/ICPR.2004.1333992). 28, 33, 49, 219
- [208] Zoran Zivkovic. Zoran zivkovic - SOFTWARE DOWNLOAD, 2007. URL: <http://staff.science.uva.nl/~zivkovic/DOWNLOAD.html>. 28
- [209] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction.

Pattern Recognition Letters, 27(7):773–780, 2006. doi:DOI:10.1016/j.patrec.2005.11.005. 28, 33, 49, 219, 342

- [210] Keju Zu, Fuqiang Liu, and Zhipeng Li. Counting pedestrian in crowded subway scene. In *Image and Signal Processing, 2009. CISIP '09. 2nd International Congress on*, pages 1–4, October 2009. doi:10.1109/CISP.2009.5303594. 63