Video segmentation and multiple object tracking

Ratnesh Kumar

UNIVERSITY OF NICE - SOPHIA ANTIPOLIS

DOCTORAL SCHOOL STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H E S I S

to obtain the title of

PhD of Science

of the University of Nice - Sophia Antipolis

Specialty: COMPUTER SCIENCE

by

Ratnesh KUMAR

Video Segmentation and Multiple Object Tracking

Thesis Advisors:

Monique THONNAT

Guillaume CHARPIAT

Prepared at INRIA Sophia Antipolis, STARS Team

defense on December 2014

Jury :

Reviewers :	Jean-Marc ODOBEZ	-	IDIAP (Martigny, Switzerland)
	Patrick BOUTHEMY	-	INRIA (Rennes)
President :	Frederic PRECIOSO	-	University of Nice (Sophia Antipolis)
Invited :	Francois BREMOND	-	INRIA (Sophia Antipolis)

Acknowledgments

TBD

Contents

1 Introduction **3**
- 1.1 Motivation . . . 3
  - 1.1.1 Sources of video data . . . 3
  - 1.1.2 Video Understanding . . . 5
- 1.2 Optical Flow . . . 6
- 1.3 Challenges for a video understanding system . . . 7
- 1.4 Problems Addressed . . . 8
  - 1.4.1 Video Segmentation . . . 9
  - 1.4.2 Multiple Object Tracking . . . 12
- 1.5 Contributions . . . 13
  - 1.5.1 Video volume segmentation with Fibers . . . 13
  - 1.5.2 Multiple object tracking using efficient graph partitioning . . . 14
  - 1.5.3 Distance covariance based descriptor for appearance matching . . . 14
- 1.6 Thesis Structure . . . 14

I Video Segmentation **17**

2 Video Segmentation: Related Works **19**
- 2.1 Image Segmentation . . . 20
  - 2.1.1 Seeded Region Growing . . . 20
  - 2.1.2 Graph Based Image Segmentation . . . 21
  - 2.1.3 Superpixels . . . 23
- 2.2 Video Segmentation without correspondences . . . 23
  - 2.2.1 Segmentation with Object Proposals . . . 24
  - 2.2.2 Hierarchical graph based video segmentation . . . 28
- 2.3 Video Segmentation with correspondences . . . 31
  - 2.3.1 Clustering of Point Tracks . . . 31
  - 2.3.2 Point Tracks and Hierarchical Video Segmentation . . . 34
  - 2.3.3 Tube or mesh based video representation . . . 35
  - 2.3.4 Motion layer segmentation . . . 35
- 2.4 Conclusion . . . 37

3 Video Segmentation: Fibers **39**
- 3.1 Introduction . . . 39
- 3.2 Fibers : Definition and Approach . . . 40
  - 3.2.1 Formalization . . . 40
  - 3.2.2 Criteria for a good representation . . . 41
  - 3.2.3 Approach outline . . . 42
- 3.3 Sparse reliable fibers . . . 42

3.3.1 Straight fibers . . . 43
3.3.2 Initiating fibers at corners . . . 47
3.4 Full video coverage . . . 51
3.4.1 Geodesics between the sparse fibers and rest of the video . . . 52
3.4.2 Enforcing trajectory coherency . . . 53
3.5 Hierarchical representation . . . 57
3.6 Computational complexity of the fiber extension stage . . . 59
3.7 Summary . . . 59

4 Video Segmentation: Implementation and Experiments 61
4.1 Implementation . . . 61
4.2 Experiments . . . 62
4.2.1 Datasets . . . 62
4.2.2 Qualitative results . . . 64
4.3 Computation times . . . 70
4.4 Quantitative evaluation w.r.t. point tracks . . . 70
4.5 Optical flow based evaluation . . . 72
4.6 Video Inpainting . . . 76
4.7 Conclusion . . . 78

II Multiple Object Tracking 79

5 Multiple Object Tracking: Related Works 81
5.1 Maximal cliques and independent sets for tracking . . . 82
5.2 Network flow based tracking models . . . 84
5.3 Stochastic sampling for tracking . . . 85
5.4 Multiple hypothesis tracking and Probabilistic data association techniques . . . 87
5.5 Discrete-Continuous optimization for tracking . . . 88
5.6 CRF based approaches . . . 88
5.7 Probabilistic occupancy maps for tracking . . . 89
5.8 Practical Considerations . . . 90
5.9 Summary . . . 91

6 Multiple Object Tracking using Graph Partitioning 93
6.1 Introduction . . . 93
6.2 Model . . . 94
6.2.1 Objective for a multi-object tracker . . . 94
6.2.2 Criteria . . . 95
6.2.3 Approach and Formalization . . . 95
6.2.4 Repulsive constraints . . . 96
6.2.5 Temporal neighborhoods and Point tracks . . . 96
6.2.6 Appearance connections . . . 98
6.2.7 Favoring smooth trajectories . . . 99

6.3 Optimization . . . 101
6.3.1 Unifying cues . . . 101
6.3.2 Graph reduction . . . 102
6.3.3 Optimizer selection . . . 103
6.3.4 Parameter setting . . . 104
6.3.5 Graph cleaning . . . 105
6.3.6 Streaming trajectories computation . . . 105
6.4 Summary . . . 106

7 Multiple Object Tracking: Implementation and Experiments 107
7.1 Implementation . . . 107
7.2 Datasets . . . 111
7.3 Inconsistency in evaluations by different proposed approaches . . . 112
7.4 Evaluating the proposed tracker . . . 114
7.5 Experiments . . . 115
7.5.1 PETS S2L1 . . . 115
7.5.2 Towncenter . . . 117
7.5.3 Parking Lot . . . 118
7.6 Optimizer Convergence . . . 119
7.7 Processing Time . . . 120
7.7.1 Usefulness of Graph Reduction . . . 120
7.7.2 Comparing processing times w.r.t. the state-of-the-art . . . 121
7.8 Conclusion . . . 126

8 Distance Correlation for Appearance Matching 127
8.1 Appearance Matching and People Re-Identification . . . 128
8.2 Classical Covariance . . . 128
8.3 Distance Correlation $\mathcal{V}^2$ . . . 130
8.4 Qualitative Results in Visual Tracking . . . 132
8.5 Quantitative Results on Person Re-identification datasets . . . 133
8.5.1 i-LIDS-MA . . . 136
8.5.2 i-LIDS-AA . . . 136
8.5.3 i-LIDS . . . 136
8.5.4 Descriptor Efficiency . . . 138
8.6 Perspectives on Single Camera Multi-Object Tracking . . . 138
8.7 Conclusion . . . 141

9 Conclusions and Future Work 145
9.1 Contributions . . . 145
9.1.1 Video segmentation . . . 145
9.1.2 Multiple object tracking . . . 146
9.1.3 Distance Correlation for appearance matching . . . 147
9.2 Future Perspectives . . . 147
9.2.1 Short term future work . . . 147

9.2.2 Long term future work 148

A Computing geodesics **151**
A.1 Dijkstra and Fast Marching Method 151

B Markov Random Fields **153**
B.1 Markov Random Fields: Connect locally and optimize globally 153
B.2 Optimization for MRF model 154

C Notes on Structure Tensor **155**
C.1 Structure Tensor 155
C.2 Properties of $\mathcal{T}$ 155
C.3 Geometric interpretation of eigenvalues of $\mathcal{T}$ 156

Bibliography **159**

List of Figures

1.1 Examples of Videos. **Top Left**: A scene from a subway station. **Top Right**: A smartphone captured video. **Bottom Left**: Medical imaging for recording heart movements. **Bottom Right**: A professional movie data. . . 4

1.2 Low and mid-level vision components for an activity recognition system. First and third rows display input videos; while the second and fourth rows indicate the outputs by employing basic mid-level vision tools, namely*background subtraction* and *multiple object tracking*. *Colored blobs* in the second and the fourth rows indicate different objects. Image source: [Ryoo 2007]. 6

1.3 Optical Flow vectors for successive frames. The flow vectors are incorrect in the circled red zone due to lack of intensity structure. 7

1.4 Sample detection output. Green boxes correspond to detector outputs. Notice missed detections and impreciseness of the detector in localization of persons. 8

1.5 **First Row**: Sample input image frames from marple13 sequence by [Brox 2010]. **Middle Row**: A dense video segmentation. The colors represent different groups. **Bottom Row**: A grouping of pixel trajectories (by [Brox 2010]) . White zones in images are not labeled and hence do not belong to any segments. 10

1.6 **Slices of a video volume. Top** row shows a few sample frames from a video. **Bottom** row displays the three orthogonal slices from the volume. The blue line indicates the separation of the slices. Bottom right inset displays the location of three slices inside the volume. 11

1.7 Deriche filter is applied on a video volume to extract the foreground (White pixels constitute the foreground). 12

1.8 Examples and visualization of fibers: **Top Left** images show a few frames from a Marple video [Brox 2010]. **Top Right** shows sample fibers in the first frame of the video. **Bottom Row** images display two snapshots from a 3D visualizer, considering the video as a 3D volume (2D+T). 13

2.1 An image grid with an eight-neighborhood system 20

2.2 Different automatic seed selection sources. **Left :** Input Image. **Center :** Seeds obtained from homogeneity criteria. **Right :** Seeds obtained from edges in Input image (Seeds are shown as white regions). 21

2.3 Image showing the importance of non local criterion which can segment above into three regions. There are three segments in an image with a rectangular-shaped intensity ramp in the left half, a constant intensity region with a hole of a high variability rectangular region. 21

2.4 Segmentation result obtained by [Felzenszwalb 2004]. 22

2.5 Sample superpixels from [Achanta 2012] of sizes 64, 256 and 1024. 23

2.6 **Left:** Braided Pattern, **Right:** T-Junctions (marked in red/green). 24
2.7 Pipeline: Object Proposal Generation by [Endres 2010]. **Top Right - Bottom Left, Bottom Right:** Compute a hierarchical segmentation, generate proposals and rank proposed regions. At each stage, classifiers are trained to focus on likely object regions and encourage diversity among the proposals, enabling the system to localize many types of objects. (**Image source: [Endres 2013]**). 26
2.8 Object proposals on a sample SegTrack video dataset (source [Tianyang 2012]). 26
2.9 Object proposal with *objectness* score. 26
2.10 Segmentation results from [Tsai 2011] on the SegTrack dataset. 27
2.11 Motion Boundaries. (a) Two Input Frames. (b) Optical flow. The hue of a pixel indicates its motion direction and the color saturation its velocity. (c) Motion boundaries based on the gradient of the optical flow. (d) Motion boundaries based on the difference in the motion direction between a pixel and its neighbors. (e) Combined motion boundaries. (f) Final motion boundaries after thresholding. (**Image source : [Papazoglou 2013]**). 29
2.12 A Video grid. 30
2.13 Image frames from different videos and corresponding outputs at a particular hierarchy. (**Image source : [Grundmann 2010]**) 30
2.14 **First Row** : Sample input image frames from marple13 sequence by [Brox 2010]. **Bottom Row**: A sparse segmentation obtained by grouping of pixel trajectories (by [Brox 2010]). White zones in images do not belong to any segments. 32
2.15 Output from [Ochs 2011]. This approach uses the sparse labeling information from [Brox 2010] (**Bottom Left**) and utilizes information from superpixels and gradients to obtain the final labeling (**Bottom Right**). 32
2.16 Background Subtraction in freely moving cameras from [Sheikh 2009]. **First Row**: Input image frames. **Second Row**: Clustered point tracks with blue representing foreground trajectories. **Third Row**: Estimated foreground labeling. **Fourth Row**: Groundtruth for foreground separation. . . . 33
2.17 Output from [Lezama 2011]. **First Row**: Input image frames. **Second Row**: Clustered point tracks with colors denoting separate clusters. **Third Row**: Dense segmentation output. **Fourth Row**: The ground truth of object regions (left) which is automatically propagated by [Lezama 2011] to other frames (middle and right). **Image Source**: [Lezama 2011]. 34
2.18 Motion tubes based representation from [Urvoy 2009]. **Image Source**: [Urvoy 2009]. 35
2.19 Sample motion layer segmentation and relative depth ordering from [Sun 2010]. **First** image shows an image frame. **Second** image shows flow field, **Third** image displays the segmentation output. The colors indicate depth ordering with green for front, blue for middle and red for back. Detected occlusions are shown in the **Fourth** image. **Image Source**: [Sun 2010]. 36

3.1 2D+T video volume represented as a 3D data, as in medical imaging: frontal, sagittal and horizontal slices correspond to cuts along planes (x, y), (y, t) and (x, t) respectively. The Colored boxes around frames are in sync with the colored boxes in right bottom images. The red lines indicate the values of x and y chosen for the cut. This video shows people exiting a train in a train station. Note in particular the lines in the (x, t) slice, formed by people trajectories, by the train or the background. We name these sets of lines *fibers*. 41
3.2 A mesh of trajectories induces a mesh of points in each frame. 42
3.3 Sample fibers. **Left:** Red colored patches show sample fibers on one frame. **Right:** Sample fibers are displayed by a 3D visualizer (ImageJ). The Green box encloses two *straight fibers* which are formed by the patches on the background. Patches forming straight fibers are marked by the blue arrows on the left image. 43
3.4 Straight fibers search. **Left**: A set of successive frames are shown, overlaid with a 2D rectangular grid. **Right**: A finer resolution grid is employed if the color correlation in coarser resolution grid is insufficient for building straight fibers. 44
3.5 Displays two straight fibers F1 and F2. Straight fibers sharing the same 2D+T cuboid and appearing at different time frames, are merged together if they are sufficiently color-correlated. 45
3.6 Background fibers and the background distance map for a static camera scene. **First Row**: Input frames for background fiber computation. **Second Row**: Straight fibers. Colors indicate the length of straight fibers. *Green* indicates fibers of same temporal length as that of the video. *Red* colored fibers are of very short temporal length (less than 0.5 of the video temporal length). *Blue* colored fibers are of intermediate temporal length between the red and green fibers. **Third Row**: Distance map obtained after extending (*cf.* section 3.4) only green (*i.e.* background) fibers. The *red* color in this distance map indicates lower distance from the background fibers, while lighter colors indicate higher distance from the background fibers. Observe that the moving objects are at a higher distance from the background fibers, than the static regions. 46
3.7 Sample interest points in an image frame. We consider a high density of interest points as corner detectors in practice often miss some important corners. 47
3.8 **Left**: Corner and its neighborhood. **Center**: K-means in color space. **Right**: Spatial coherency using graph cuts. 48
3.9 **Flow improvement** by regularizing the mesh. Left: $\mathcal{M}_t$. Right: regularized mesh M_{t+1}. The black arrow indicates the direction of the motion of the moving object. The mesh shown is on the background and the elongation of this mesh is due to the inaccurate flow near the motion boundary of the black moving object. Green edges indicate the closing boundaries of this mesh. 50

3.10 **Flow improvement** by regularizing the mesh. Left: $\mathcal{M}_t$. Right: regularized mesh M_{t+1}. The black arrow indicates the direction of the motion of the moving object. The mesh shown is on the background and the elongation of this mesh is due to inaccurate flow near the motion boundaries. Green edges indicate the closing boundaries of this mesh. 51
3.11 **Splitting a fiber**. *Red* and *Green* colors indicate separate fibers. During the fiber build process, for a certain time interval, these two fibers were considered a single fiber. Thanks to our motion coherency criteria, we detect a motion in-homogeneity at a certain time (indicated by dashed *blue* lines). Owing to this motion in-homogeneity detection, we split (spatially and temporally) the fiber into two, and continue the build process for each of these fibers separately. 51
3.12 **Top Row**: Displays five images of a sequence from [Brox 2010]. **Bottom Row**: Left image displays the same video as a 3D volume as in Fig. 3.1, with an additional bottom right corner showing the 3D position of the xt and yt slices. Right image displays all fibers found, in different colors. Notice that the fibers are well stopped near occlusion vicinity, despite high similarity in color of the tractor and the lady. 52
3.13 Homogeneous and heterogeneous trajectories. The color histogram of a pixel trajectory should be as close to a Dirac peak as possible. In practice, due to some lighting variations or an imperfect camera sensor, the color histogram will not be a perfect Dirac peak. The vicinity of a histogram to a Dirac peak is quantified using the Earth Movers Distance (*cf.* expression 3.4). 54
3.14 The possible extension zone (in green) of a fiber (in brown) is projected on a reference frame (in red), following the trajectories T_p^F (in blue) associated coherently with the fiber F. 56
3.15 **Left** image displays a meshed region of a fiber footprint. **Right** shows the distance map w.r.t. fiber. The red regions correspond to low distances, while green and blue correspond to comparatively higher distances respectively. Note here that the leaks are at higher distance from the sources in Red color. 56
3.16 Sample distance maps for fiber footprints. The Red regions correspond to low distances, while green and blue correspond to comparatively higher distances respectively. Notice that the leaks are at higher distance from the sources in Red color. 57
3.17 Displays extended fibers merged at a higher hierarchy. For a simple illustrative viewing, we fuse fibers which have similar motion. The next section 3.5 will elaborate on merging fibers. Darker regions refer to a low reliability of segmentation. 58

4.1 Workflow for fiber based video segmentation. 61
4.2 Sample frames from the dataset provided by [Grundmann 2010]. This dataset does not provide groundtruth annotations. 62

4.3 Sample frames from `xiph.org` dataset. Each row indicates three regularly spaced frames from a separate video. This dataset does not represent current day video usage as the frame resolution is mere 240x160, along with a poor frame rate. Importantly in many videos, there is no motion and the first and the last frames are identical (*cf.* last row). 63
4.4 Sample video frames from the motion segmentation dataset (*moseg*) provided by [Brox 2010]. This dataset provides pixel wise groundtruth annotation for a few frames in each sequence. We consider many videos from this dataset and also use their evaluation metrics (and software) to provide quantitative results. 64
4.5 Example from the *marple3* sequence till the 50th frame. **Second-Third Rows**: Un-extended fibers at different hierarchy of merging. **Third-Sixth Rows**: Increasing hierarchy in merging of extended fibers. 65
4.6 **First Row**: Displays five images of a sequence from [Brox 2010]. **Second Row**: Left image displays the same video as a 3D volume as in Fig. 3.1, with an additional bottom right corner showing the 3D position of the xt and yt slices. Colored arrows on the bottom slice indicate the tractor (*green arrows*) and the lady (*blue arrows*). Right image displays all fibers found, in different colors. **Third Row**: Left image displays a high level of their hierarchical clustering. Right Image displays the highest level of the hierarchical clustering, with fiber extension. This result compares favorably to the state-of-the-art of video segmentation in Figure 4.10. 66
4.7 **First Row**: Sample input image frames (12, 24, 46, 53, 66) from marple13 sequence by [Brox 2010]. **Bottom Row**: After 5 steps of hierarchical merging. Parts of the foreground and background are undistinguishable during the first frames (*until the 12th frame,* i.e. *the leftmost image in the above figure*) of the video (same color). Yet, the two objects follow later significantly different trajectories, which enables us, when propagating this information back in time, to separate them as different fibers in all frames (*cf.* section 3.3.2). 67
4.8 Output from [Sand 2008] dataset (frame numbers 1, 10, 20, 40). Fast moving cars are appropriately segmented at a particular hierarchy. 67
4.9 **First Row**: Sample input image frames at spacing of 40 frames till 110th frame. **Second Row**: Fiber merging at a lower hierarchy. Notice that the background belongs to one cluster now. **Third Row**: Fiber merging at higher hierarchy. Notice that different parts of the moving persons (*e.g.* head, torso, elbow) belong to different clusters. **Fourth Row**: Dense fiber flow after trajectory based association (3.4.2). **Fifth Row** : Extended Merged Fibers at a lower hierarchy. In this and following images, *zones segmented with low reliability are darker in color*. **Sixth Row**: Extended merged fibers at a higher hierarchy level. Some part of leg now belongs to the background. This due to the hierarchical merging cost function we employ. The un-extended fibers, however, have well respected boundaries as can be seen in other images. 68

4.10 Output from [Grundmann 2010]. **Left**: Input image and its spatio-temporal slices. Colored arrows on the bottom slice indicate the tractor (*green arrows*) and the lady (*blue arrows*). **Right**: Result obtained by the state-of-the-art segmentation of videos [Grundmann 2010], using also optical flow, for comparison. The main foreground object and the background are already merged at a relatively low hierarchical level (third). It may appear that increasing the number of hierarchy levels could solve this leak. However the two objects in the circled zone is of similar color and a local color & motion assessment is incapable of indicating the presence of two different objects. Our result for this video can be seen in Figure 4.6. 69
4.11 Sample image frames from the MPI-Sintel optical flow evaluation dataset [Butler 2012]. Each column displays consecutive image frames from one sequence. Large motion and frequent occlusions prohibit the usage of this dataset for our fiber based video representation. 73
4.12 Sample frames from the MIT optical flow dataset [Liu 2008]. This dataset includes groundtruth annotations. 73
4.13 Optical flow computed by [Werlberger 2009] for the two consecutive frames on the left. Colors on the *right* image is the standard *Middlebury dataset* encoding of the optical flow. We use this optical flow algorithm as input flow for our fiber based video representation. Notice that the flow is wrongly estimated for areas between the fingers of the hand. 74
4.14 Interpolation of trajectories between articulated object parts. **Left**: A rotating object part. t_1 and t_2 mark the time instants and show poses of object. **Middle**: Expected flow is shown by blue arrows. Notice that a linear interpolation is sufficient, while a piecewise-constant interpolation cannot work in the case of rotations. **Right**: Interpolating trajectories from two trajectories (**A**, **B**) located at articulated object parts. 75
4.15 Input frames for the inpainting demo (Frame numbers: Top Row: 0, 22. Bottom Row: 44, 61). 76
4.16 Inpainting task. **Left**: original video (top) and xt slice (bottom) showing trajectories. **Right**: Our result. Clusters of fibers were computed and selected with only 7 mouse clicks to distinguish the disturbing girl from the reporter and background. The girl was removed and the hole was filled by extending the background fibers in time. 77
4.17 xt slices for the input (*top* row), and dense fibers (*bottom* row) at the finest hierarchy. The spatial statistics help in getting good boundary contours of the foreground object (*i.e.* reporter) by only a few user mouse-clicks. The straight fibers parallel to the time axis help in completing the hole formed by removing the girl. Fibers belonging to the girl are identified by finding fibers which are not parallel to the time axis. 77

5.1 Multiple Object (Person) Tracking. The labels for each bounding box is the unique identity of a person. Dotted points show the trajectories of bounding boxes in a few previous and successive frames. 82

5.2 A maximal cliques-search based tracker. Gray and colored edges represent input graph and optimized subgraph respectively. **Image source : [Zamir 2012]**. 83

5.3 Network flow model graph for [Zhang 2008]. Nine detections (including false positives) from three successive frames are shown. Trajectory computation is performed by maximizing the flow from the source s to sink t. Nodes which are not connected (in the above Figure) to either of s, or t are deemed as false positives (post optimization) **Image Source:**[Zhang 2008]. 84

5.4 Higher order connections in network flow model by [Butt 2013]. **Top**: Graph depicting a three frame sequence with three observations in the first frame, two in the second and four in the third frame. **Bottom**: Graph proposed by [Butt 2013]. Candidate match pairs from top graph form nodes in this graph and thin black edges are added between match pairs that share an observation in frame 2, and thick colored hyperedges represent additional constraints that mush be enforced so that each observation is used only once in the matching solution. **Image source**: [Butt 2013]. 86

5.5 Probabilistic Occupancy Maps. **a:** Original Image from three cameras. **b:** Probabilistic Occupancy Maps for images in above row. **c:** Figure represents the corresponding occupancy probabilities on the grid. **Image Source:**[Berclaz 2011] 89

6.1 High repulsive exclusion constraints, shown for the central green bounding box in frame t, with **red** edges. Intra-frame exclusions prevent this detection from having the same label as any other detection in the same frame t, while inter-frame exclusion constraints prevent it from having the same label as detections in previous and next frames $t-1, t+1$ that are too far. The maximum radius **r** is detection-dependent, in that it corresponds to the maximum physically-plausible displacement within a duration of one frame, and that speed estimation depends on object depth. 97

6.2 Point tracks and detections. 98

6.3 Sample Appearance Clusters for PETS09 S2L1: Each row displays a few samples belonging to the same cluster. Different sizes of cropped detections indicate the distance of persons from the camera. It can be seen that the appearance features are robust to scale changes. 100

6.4 Computing the trajectory straightness (equation (6.1)) for a triplet of detections j, i, k at the temporal scale δt. 101

6.5 Edge weights between the fused nodes 102

6.6 Graph reduction and flipper speed gain. Leftmost drawing shows the labeling of 12 nodes. Node colors indicate a labeling (stuck at a local minima) and the colored lines (*green* and *yellow*) correspond to the groundtruth trajectories. Middle inset drawing shows the reduced graph by fusing nodes inside the *dotted* ellipse in the Leftmost drawing (*cf.* sec. 6.3.2). Since the graph is reduced, we only need to search for flips of subgraphs of size = 2 (*i.e.* $O(|V|^2)$), instead of enumerating over all possible subgraphs of size = 6 (*i.e.* $O(|V|^6)$). The subgraph-flip required to correct the initial labeling is shown by *magenta* colored rectangular box in the middle inset. The Rightmost drawing correspond to the final correct output in the original graph. 103
6.7 Illustration of the *graph cleaning* (*cf.* section 6.3.5) step. The colors of the nodes represent their associations, and same colored nodes belong to the same output trajectory. Nodes inside *green* boxes indicate interpolated detections for false negatives. *Magenta* colored nodes are false positives. . . 105
6.8 Streaming trajectory computation. Green and black colored lines represents two trajectories computed in batch 1. *Magenta* colored nodes in the overlap time span, which belong to the same trajectory (from batch 1) are **fused** together for batch 2. 106

7.1 Workflow of the proposed approach. 107
7.2 **Model Summary**. **Left**: Point Tracks and detections. **Middle**: High repulsive exclusion constraints, shown for the central green bounding box in frame t, with **red** edges. Intra-frame repulsive constraints prevent this detection from having the same label as any other detection in the same frame t, while inter-frame repulsive constraints prevent it from having the same label as detections in previous and next frames $t-1$, $t+1$ that are too far. The maximum radius **r** is detection-dependent, in that it corresponds to the maximum physically-plausible displacement within a duration of one frame, and that speed estimation depends on object depth. **Right**: Computing the trajectory straightness (*cf.* equation (7.2)) for a triplet of detections j, i, k at the temporal scale δt. 109
7.3 Sample frames from the PETS dataset. 111
7.4 Sample frames from the TownCenter dataset. 111
7.5 Sample frames from the Parking Lot dataset. 112
7.6 A sample frame from the PNNL ParkingLot sequence showing major deficits in the provided annotation. Besides mixing up the identities, several people are not marked in this ground truth. **Image source**: [Milan 2013a]. 113
7.7 **Missing persons in annotation** : Frame numbers 509 and 526 from PETS S2L1 video. The person below the red arrow is not annotated for a length of 17 frames. 113
7.8 Video slices along the (x, t) plane, i.e. for fixed y, for PETS S2L1, showing that labels for trajectories before and after occlusions are maintained. 115

7.9 PETS S2L1 Output: The numbers in *red* on each image frames show the corresponding frame numbers from this video batch. The dotted points shows the trajectories of bounding boxes in previous and successive frames. To reduce clutter in display, we show 3 trajectories and their *few* past and future tracks. Notice that due to the global appearance incorporation, ID 4 is kept intact for the person as he leaves and comes back in the view. Also trajectories before and after crossing are fully consistent. This immaculate consistency despite False Negatives near occlusion vicinity is obtained due to the triplet factors. 116
7.10 **Consistent people crossing.** IDs 0, 11, 13, 14 (in the left part of image frames) are of interest in **top rows**. IDs 17 and 19's (right part of the image frames) consistency is shown in the **bottom row**. 122
7.11 **Consistent people crossing in dense scenarios.** IDs 8, 9, 11, 2,3,7 are intact even after occlusion and false negatives. Images in the first row and last rows are 121 frames apart. 123
7.12 Iterations vs. Energy, on a problem size of 4000 variables. TRW-S is able to find usable solutions from first iteration (*cf.* section 7.6). 124
7.13 Iterations vs. Energy. The RED line corresponds to TRW-S, while the BLUE line corresponds to MPLP [Sontag 2012]. We observe that TRW-S is fast to minimize energy, and MPLP is unable to find one usable solution. Similar is the issue with another polyhedral method AD3 (*cf.* section 7.6). . 124
7.14 Processing time improvement by using graph reduction (similar MOTA in both cases), for the first 15 batches. High processing times from batch 8 to 12 are due to high detection rate/frame. In the above cases we let the optimizer run until convergence. 125
7.15 Average optimization processing times for the state-of-the-art tracking approaches. Note that this comparison is approximate due to the reasons mentioned in section 7.7. Nevertheless there is a significant gap indicating good computational efficiency of our approach. Note that we have not added the time required by the external tracker for [Milan 2013c]. 125

8.1 Top and Bottom Rows correspond to images from different cameras. 128
8.2 Correlation between a Parabola and a Line. We show that the classical covariance fails to compute any correlation, while the recently proposed *distance correlation* by [Szekely 2009] computes good correlation. 129
8.3 Template Tracking using Covariance. **Top** image shows tracking with the covariance based descriptor. **Bottom** image shows better drift immunity by using the proposed Brownian descriptor. 134
8.4 Template Tracking using Covariance. **Top** image shows tracking with the covariance based descriptor. **Bottom** image shows better drift immunity by using the proposed Brownian descriptor. 135
8.5 Performance comparison on i-LIDS-MA [Bak 2011b].Top figure corresponds to $N = 1, 3$, while bottom figure has $N = 5, 10$ (*cf.* section 8.5 for details). 137

8.6 Performance comparison on i-LIDS-AA [Bak 2011b] using different metrics: $L1$ corresponds to the L_1 norm metric between vectorized Brownian descriptors, $L1T$ refers to the L_1 norm on a tangent plane, R - geodesic distance [Forstner 2003] 138
8.7 Performance comparison on i-LIDS [Zheng 2009]. 139

A.1 Difference between Dijkstra and Fast Marching Method. **Image Source** : `http://tosca.cs.technion.ac.il/book/course_siam10.html` 151
A.2 Dijkstra Algorithm. x_0 is the source and hence $d(x_0)$ is zero; Set X represents all nodes on a grid. **Image Source** : `http://tosca.cs.technion.ac.il/book/course_siam10.html`. 152
A.3 Algorithm Fast Marching Method. x_0 is the source and hence $d(x_0)$ is zero; Set X represents all nodes on a grid. Notice the difference in the update step w.r.t. Dijkstra Algorithm in Figure A.2. **Image Source** : `http://tosca.cs.technion.ac.il/book/course_siam10.html`. . 152

C.1 Top: A noisy image with no structure such as edges and corners. Bottom: The distribution of I_x and I_y. Both eigenvalues of $\mathcal{T}$ are of very small magnitude, and hence the ellipse is of small size. 156
C.2 Top: An image with an **edge**. Bottom: The distribution of I_x and I_y. One eigenvalue is of very large magnitude stretching one axis of ellipsoid along the corresponding eigenvector. 157
C.3 Top: An image with a **corner**. Bottom: The distribution of I_x and I_y. Both eigenvalues are of very large value and hence the fitted ellipsoid is of larger size than in Figures C.1 and C.2 157

List of Tables

2.1 State-of-the-art approaches and the information provided by them. **Correspondences** column indicate long term information about a pixel's motion across the video. **Dense Coverage** column indicates if the segmentation approach partitions the video fully *i.e.* each pixel is associated to a particular segment. **Reliability Factor** column shows if the segmentation approach provides a per-pixel factor expressing the long term quality of segmentation. 38

4.1 Computational time: Typical times observed for one second of a standard video, worst case times (fast background motions), and time expected if using the last GPU card and parallelization of successive chunks in a video stream on a standard 8 processor 2.4 GHz machine. 70

4.2 Quantitative results on the *moseg* dataset [Brox 2010]. Above metrics are the averaged over 15 videos (of 50 frames each). 71

4.3 Optical flow evaluation on the MIT dataset [Liu 2008]. Each row shows quantitative optical flow evaluation for a particular video, in terms of **per pixel error**, averaged over the whole sequence. 74

5.1 State-of-the-art approaches and their incorporation of local and global clues. TF stands for the curvature factor (triplet). GAF denotes global appearance factor. 91

7.1 Comparison with recent proposed approaches on PETS S2L1 Video. The metrics on the first and last row are obtained using the same code, while the middle ones are taken from the respective papers as their result files are unavailable. 117

7.2 Quantitative Results on Towncenter Video batch for frames 1-1000. The authors of [Benfold 2011] use information from head detector along with other information such as point-tracks to refine the locations of bounding boxes, and hence the quantity MOTP is higher. MOTA is the preferred metric for us as we do not make any changes in the locations of input bounding boxes, apart from adding bounding boxes for dealing with false negatives from the detector. **Note** that the work by [Heili 2014b] was published during this thesis completion period. 118

7.3 Quantitative Results on a crowded scene batch of 400 frames, starting from the 450th frame. 118

7.4 Quantitative Results on Parking Lot Video. Note that the groundtruth annotations exclude some non-occluded pedestrians and there is ID ambiguity at some frames. 119

7.5 **Optimizer scaling w.r.t. problem size**. Computation times (in **seconds**) for different batch sizes for PETS S2L1 video. Times in column *App*+k-means measure both appearance feature computation and clustering processing. For a more practical streaming trajectory computation on 50 frame batches, our optimizer takes 0.016 s/frame (on an average). Refer to the Figure 7.15 for comparative details on processing times. 120
7.6 **Usefulness of graph reduction**: PETS S2L1 computation times (in **seconds**) for the cases when the graph reduction step is either chosen (*Row 1*) or discarded (*Row 2*). The column ***App*+k-means**, shows cumulative times for both appearance feature computation and clustering. The quantitative results for both rows are similar. 121
7.7 State-of-the-art approaches and their incorporation of local and global clues. TF stands for the curvature penalization factor (triplet). GAF denotes global appearance factor. 126

8.1 Quantitative Results for the Towncenter Video for a 390 frame batch. The first column identifies the appearance feature descriptors. 139

Introduction

CHAPTER 1

Introduction

Contents

1.1 Motivation 3
1.1.1 Sources of video data 3
1.1.2 Video Understanding 5
1.2 Optical Flow 6
1.3 Challenges for a video understanding system 7
1.4 Problems Addressed 8
1.4.1 Video Segmentation 9
1.4.2 Multiple Object Tracking 12
1.5 Contributions 13
1.5.1 Video volume segmentation with Fibers 13
1.5.2 Multiple object tracking using efficient graph partitioning 14
1.5.3 Distance covariance based descriptor for appearance matching . . . 14
1.6 Thesis Structure 14

1.1 Motivation

Video understanding is the automatic analysis and interpretation of video data. Owing to the huge amount of video data being generated everyday, an automatic video interpreter has become one of the most sought-after computer vision application.

Despite recent developments in computer vision, the maturity of video understanding systems is far from being perfect. In this thesis we address the problem of *segmentation* and *multiple object tracking* to aid video understanding. These two key problems form the core of almost all approaches for video understanding.

The following subsection presents the motivation for an automatic video understanding. Subsequently we elaborate on the building blocks that constitute most video understanding systems.

1.1.1 Sources of video data

The proliferation of cameras has driven an explosion in the amount of video data generated these days. Major contributing sources for this enormous video data are :

- Installation of security cameras at public places like streets, airports, subways. There is one camera for every 32 persons in the United Kingdom with 1.85 million cameras in function (both indoors and outdoors).

- Healthcare: During the last decade there has been a significant move towards evidence-based medicine, which involves systematically analyzing clinical data acquired from different sensors and making treatment decisions based on the best available information. Typical application examples include videos recording movement of organs such as the heart, video data corresponding to monitoring daily livings of patients for detection of crucial diseases like Alzheimer.

- Structured and Unstructured movies: *Structured* video data refers to the videos created by professionals *e.g.* cinemas, TV shows; while unstructured corresponds to data created by general public. With the decrease in prices of smartphones & consumer cameras, coupled with a steep increase in social media interaction, there has been a burst in the creation of unstructured data. One of internet's major video data provider Youtube gets uploaded with 48 hours of video everyday. Images and videos altogether comprise 80% of all corporate and public unstructured big data.

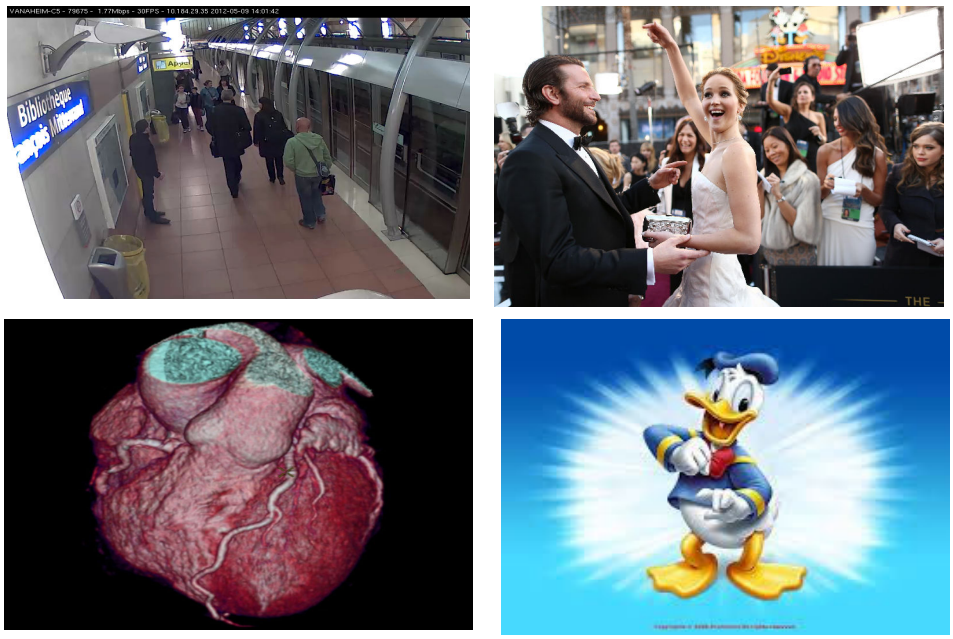

Figure 1.1: Examples of Videos. **Top Left**: A scene from a subway station. **Top Right**: A smartphone captured video. **Bottom Left**: Medical imaging for recording heart movements. **Bottom Right**: A professional movie data.

Figure 1.1 shows typical sources for video data. Owing to this huge amount of video data, there is a pervasive need in any sector (industry, government) to analyze these data **automatically**. The enormousness of this data also presents challenges to create efficient storage hardware and software, which is a driving force behind research in data compres-

sion and related hardware. Automatic analysis of video (*i.e.* video analytics) data can help prescribe actions which in-turn will improve health care, reduce crime, and so on.

1.1.2 Video Understanding

Video understanding is the automatic and logical analysis of information found in the video data. An example of an understanding system is a **people counter** at supermarkets which could help managing customer services at tills efficiently. On a computer, images are represented as *vector images* or *raster images*. Raster images are sequences of pixels with discrete numerical values for color while vector images are a set of color annotated polygons. A video is a sequence of images. In order to extract logical information from videos, the encoding must be transformed into constructs depicting physical structures, objects and motion. These constructs will then be analyzed by the computer.

In order to understand the typical building blocks of a video understanding system, let us consider the workflow of an *activity recognition* system. The aim of an activity recognition system is to *automatically* label objects, persons and events in a given video. Activity recognition is an important research domain in computer vision and its application areas include surveillance systems, healthcare monitoring, human-computer interactions. Automated surveillance systems in public places such as streets, airports, subways aim at detecting abnormal and suspicious activities. Another important usage of activity recognition is in automatic monitoring of patients and elderly human beings.

Owing to the importance of the task of activity recognition, numerous approaches are being proposed every year at major computer vision conferences. However the accuracy of these systems are far from being perfect. The major limiting factor for good performance of an activity recognition system is the unreliability of *low and mid level* computer vision components. Low-level vision refers to processing of an image or a video for extracting primitive features such as color edges or corners. Mid-level vision tasks use inputs from low-level vision algorithms to accomplish tasks such as estimation of *3D scene properties* from 2D images, *determining camera motion* from videos and *segmentation* of an image (or video) into coherent subsets. The automatic interpretation of the information provided by low and mid level vision for conceptual description of the scene (*e.g.* activity recognition) is referred to as *high level* vision.

The low and mid level vision components form building blocks for high level vision tasks such as activity recognition. Figure 1.2 displays sample outputs after a low and mid level processing of the input video. This processing is required for the task of human-object interaction (picking a trash-can or moving hand-luggage). Given an input video, object hypotheses are obtained on a per-frame basis. This task is commonly referred to as *object detection*. Often algorithms such as *background removal* is employed in order to boost the accuracy of object detection algorithms. A background removal aids an object detection algorithm in removing the noisy outputs which are on high probable background regions.

Subsequently, in order to detect activities performed by objects, the requirement is to infer the motion of the objects in the scene. This task of computing motion of objects in a scene by associating detection hypotheses in image frames is known as *Multiple Object*

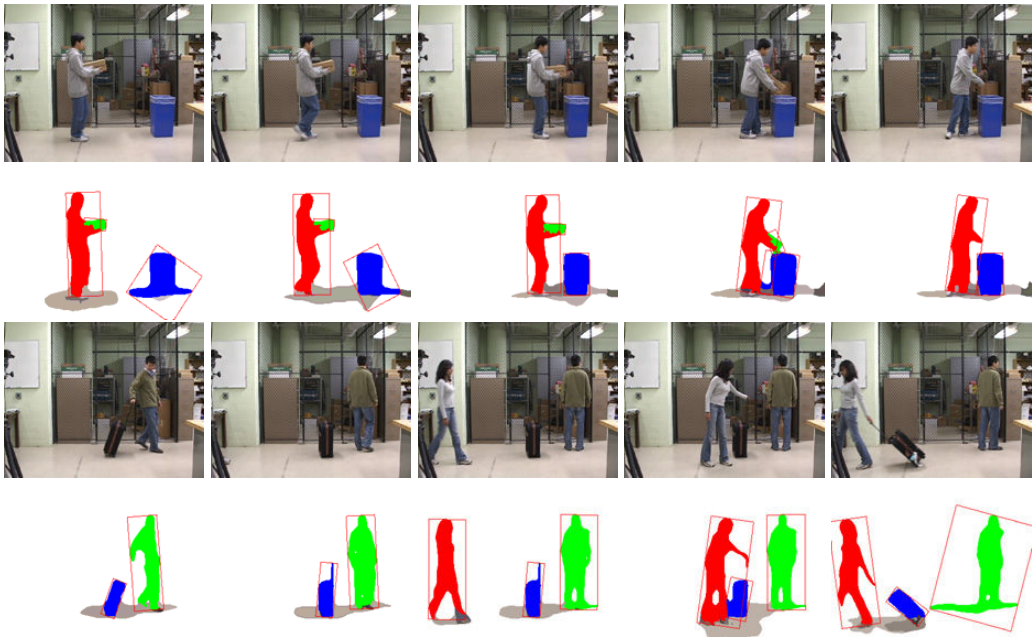

Figure 1.2: Low and mid-level vision components for an activity recognition system. First and third rows display input videos; while the second and fourth rows indicate the outputs by employing basic mid-level vision tools, namely*background subtraction* and *multiple object tracking*. *Colored blobs* in the second and the fourth rows indicate different objects. Image source: [Ryoo 2007].

Tracking. The second and fourth row in Figure 1.2 show outputs after performing background removal and object tracking on the input video.

Recognition of tasks such as picking a hand-luggage or placing a box in trash-can (*cf.* Figure 1.2) require details about movement of the hand and pixels constituting the hand. This is accomplished by computing trajectories of pixels belonging to the hand. Trajectory of a pixel refers to a time ordered set of points which determine its temporal movement across the video.

An activity recognition approach exploit the above low and mid-level vision information to recognize activities in a video. Typically this step involves statistical pattern recognition algorithms to model the relationship between activities and the basic information obtained from low and mid-level vision tasks.

1.2 Optical Flow

Before we delve any further, it is important to mention one of the important concepts, namely *optical flow*, which forms the basis for motion computation of pixels in videos, and is used in almost all video understanding systems. Optical flow is the distribution of apparent velocities of movement of brightness patterns in an image [Horn 1981]. Optical flow computing algorithms input a pair of consecutive image frames (one as source and another as target), and provides apparent motion estimates for each pixel from the source

Figure 1.3: Optical Flow vectors for successive frames. The flow vectors are incorrect in the circled red zone due to lack of intensity structure.

image to the target image. Several techniques for computing optical flow exist in the literature. However the accuracy is very limited due to practical difficulties in estimating large displacements for pixels. Moreover it is difficult to compute reliable flow vectors in homogeneous color areas. The red zone in Figure 1.3 shows inaccuracy in optical flow vectors inside the red colored elliptical zone. Notice that the flow vectors are of much higher accuracy near corners and edges. The flow at corners are more reliable due to lesser ambiguity in matching corners in successive frames. The flow ambiguity at edges is also referred as the *aperture problem* in the literature.

1.3 Challenges for a video understanding system

In the previous sections we have seen some basic vision components used for recognizing activities. The information obtained from the low and mid level vision component is also referred to as *features*, in the computer vision literature. The features are encoded into suitable descriptors. The features and their descriptions need to be carefully chosen so that they encode definitive information regarding the presence of activities.

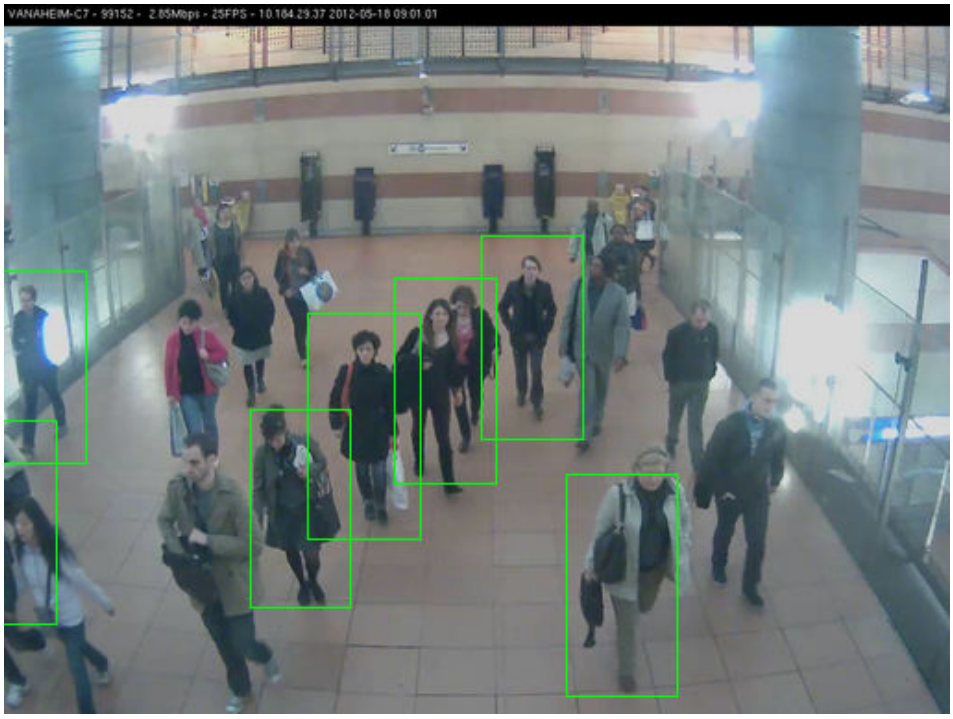

Figure 1.4: Sample detection output. Green boxes correspond to detector outputs. Notice missed detections and impreciseness of the detector in localization of persons.

Video understanding is challenging due to unreliable outputs provided by basic low and mid-level vision algorithms such as optical flow, object detection. Another challenge for a video understanding system lies in building tractable models for relating features and objectives.

In an earlier section we have seen the inaccuracy of motion estimates provided by optical flow, which impairs the computation of dense and accurate trajectories for pixels in the video. This in-turn limits the amount of reliable temporal information available for video understanding.

Figure 1.4 shows the output from another important lower level vision component, namely an object detector, for a video understanding system. A person detector is employed to obtain detection hypotheses on the image frame shown in Figure 1.4. The detections obtained by the person detector are shown by the rectangular boxes. Notice that many persons are not detected by the detector, and the localization of some persons are poor. This impreciseness of an object detector pose serious challenges to the *tracking of objects*, and further limits the accuracy of an understanding system requiring an object tracker.

1.4 Problems Addressed

In this thesis we aim to improve the key mid-level vision components, namely *video segmentation* and *multiple object tracking*, to aid video understanding.

A segmentation algorithm provides detailed spatio-temporal relationships of pixels in a video, which are necessary to analyze the geometry and shape changes of object/object-parts across time and space. Another important component of a video understanding system is a *multiple object tracker* which provides spatio-temporal movement details about objects in a scene. This output provided by a multi-object tracker is at an object level and hence detailed movements of pixels comprising objects is not available. On the other hand segmentation algorithms, generally do not provide a notion of objects in the outputs. Segmentation algorithms which provide *semantic label* for each pixel of a video are referred to as *semantic* segmentation methods.

A good segmentation and multi-object tracking for a video understanding system aids in the extraction of reliable features. This subsequently helps in modeling robust relationships between features and output variables. In this thesis we address both these problems, namely video segmentation and multiple object tracking.

1.4.1 Video Segmentation

A segmentation algorithm partitions an input video into several components. Each components is homogeneous w.r.t. one or more properties *i.e.* the variation of measurements within the regions should be considerably less than variation at the object borders.

Numerous segmentation algorithms exist in the literature. Also any particular segmentation approach cannot be suitable for all video understanding tasks. About 30 novel segmentation approaches get published at a major computer vision conference, underlining the hunch for a reliable segmentation approach. However a reliable and robust segmentation still eludes the vision community.

The requirements for segmentation depends on the video understanding task. Moreover there can be different segmentation algorithms catering to the same video understanding task. A people counting system may use a motion based segmentation or a segmentation based on filters whose response can localize the persons in a video.

Video segmentation approaches can be broadly classified into two categories, based on the impetus on color on motion information to obtain a segmentation :

- Color based : Approaches belonging to this category primarily rely on color of the pixels to perform segmentation.

- Motion based : Motion provides vital clues for grouping and algorithms under this category rely mainly on motion information to achieve segmentation. Owing to the difficulty in extraction of motion at color homogeneous regions, many algorithms belonging to this category can provide grouping of only a subset of pixels in a video. The last row in Figure 1.5 shows an example for sparse motion based grouping. This grouping has little or no spatial coherency in the output. Typically these sparse representations provide grouping for a mere subset of 3 to 5% of the total pixels in the video.

Most segmentation approaches use a combination of both motion and color information. However the segments obtained are 2D+time (3D) blobs, termed as *supervoxels*,

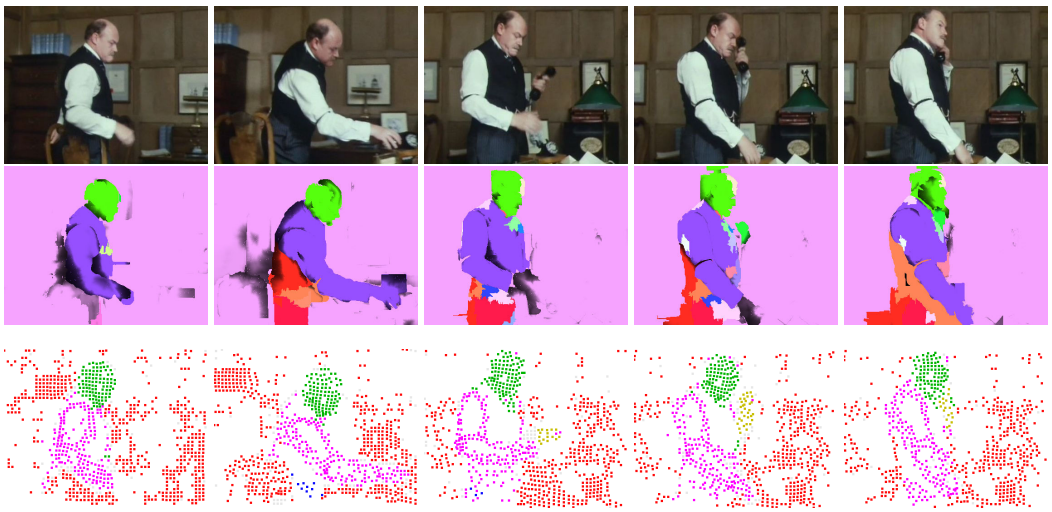

Figure 1.5: **First Row**: Sample input image frames from marple13 sequence by [Brox 2010]. **Middle Row**: A dense video segmentation. The colors represent different groups. **Bottom Row**: A grouping of pixel trajectories (by [Brox 2010]) . White zones in images are not labeled and hence do not belong to any segments.

which provide little or no detail about spatio temporal motion about pixels constituting the blobs.

The proposed approach in this thesis uses both color and motion information to segment a video. In contrast to the existing state-of-the-art segmentation approaches, we aim at building long term spatio-temporal movement details for all pixels in the video.

Most of the segmentation approaches proposed until the early 2000s are based on frame-by-frame processing. Recently, with the increase in computational efficiency of computers, there is an emerging trend to compute representations based on stacking multiple successive frames (in the order of hundreds) of the video. We term this stack **video volumes**. Volume-based approaches have gained importance, as jointly processing all frames of a video brings more information and helps maintaining a coherent segmentation over time. Figure 1.6 shows slices of a video volume formed by stacking 100 successive frames. In this figure, the first row indicates sample frames of a video. The lower image (inset) displays the video volume by it's three orthogonal slices (*cf.* caption for more details).

On this video volume, one of the simplest segmentation task could be to extract foreground by computing edges in the temporal (T) dimension. The intuition is that for a fixed camera video, if a person is walking perpendicularly to the camera's optical axis with sufficient speed, then the pixels corresponding to the edges will belong to the person. To this end we use the Deriche filter [Deriche 1987] for computing edges along the temporal axis. This filtering is followed by a thresholding step wherein sufficiently strong-valued edges are kept and the rest are suppressed (dark zones in Figure 1.7). Thus we now obtain a binary labeling of the video wherein the foreground corresponds to the white zones in Figure 1.7. The high coherency in background labeling is due to jointly processing the 2D+T data in the form of a video volume. This joint processing helps an edge detector to localize

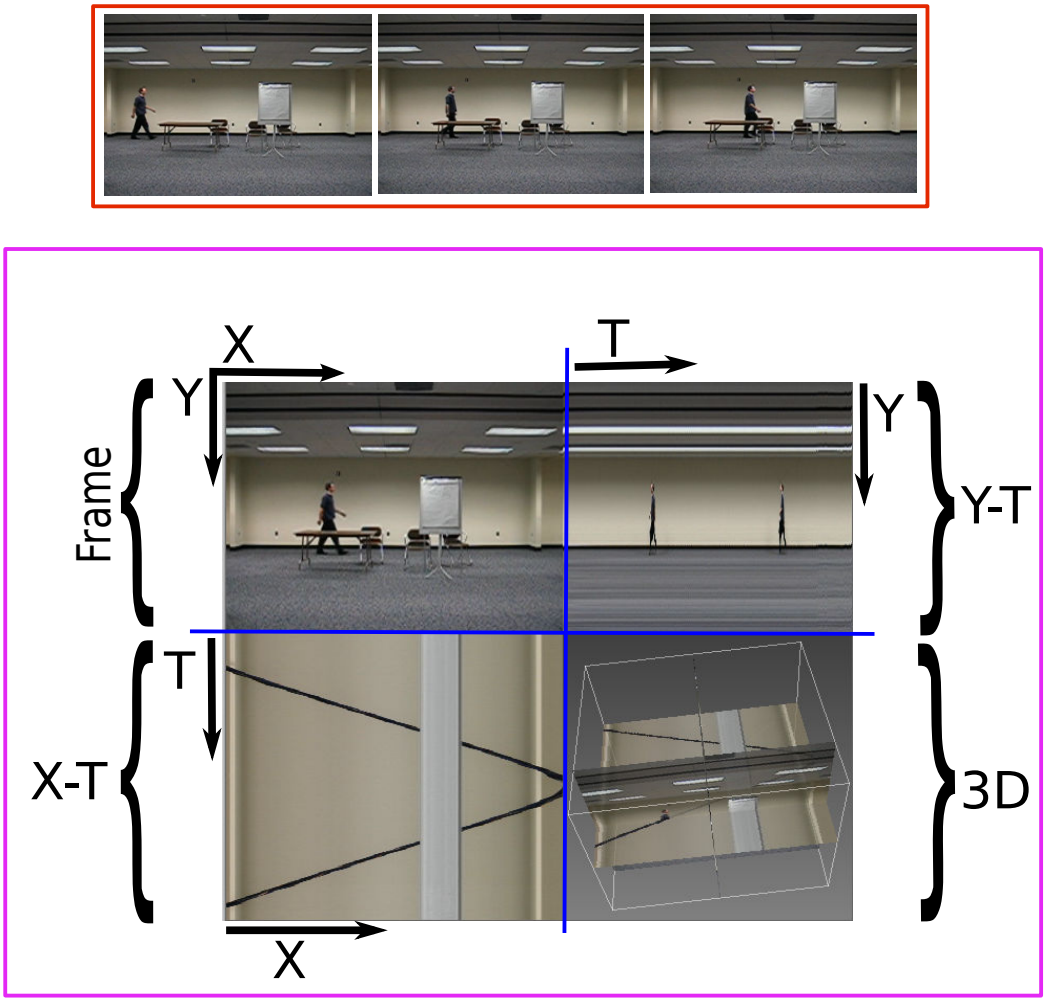

Figure 1.6: **Slices of a video volume. Top** row shows a few sample frames from a video. **Bottom** row displays the three orthogonal slices from the volume. The blue line indicates the separation of the slices. Bottom right inset displays the location of three slices inside the volume.

edges in a better way than considering only two successive frames.

Another advantage brought by this volume is long temporal range information. Local motion information among spatially neighboring pixels can be inaccurate as shown in Figure 1.3. However modeling long term affinity (with suitable criteria) between pixels could help in getting around this problem in color homogeneous regions. With reference to the example in Figure 1.3, a viable approach to build accurate flow in red zones (lacking discriminative intensity structure) is to propagate the high accuracy flow from the edges and corners on the moving person.
In this thesis we pursue video volumes to obtain a robust, reliable and dense video segmentation along with long range motion estimate for each pixel in the video.

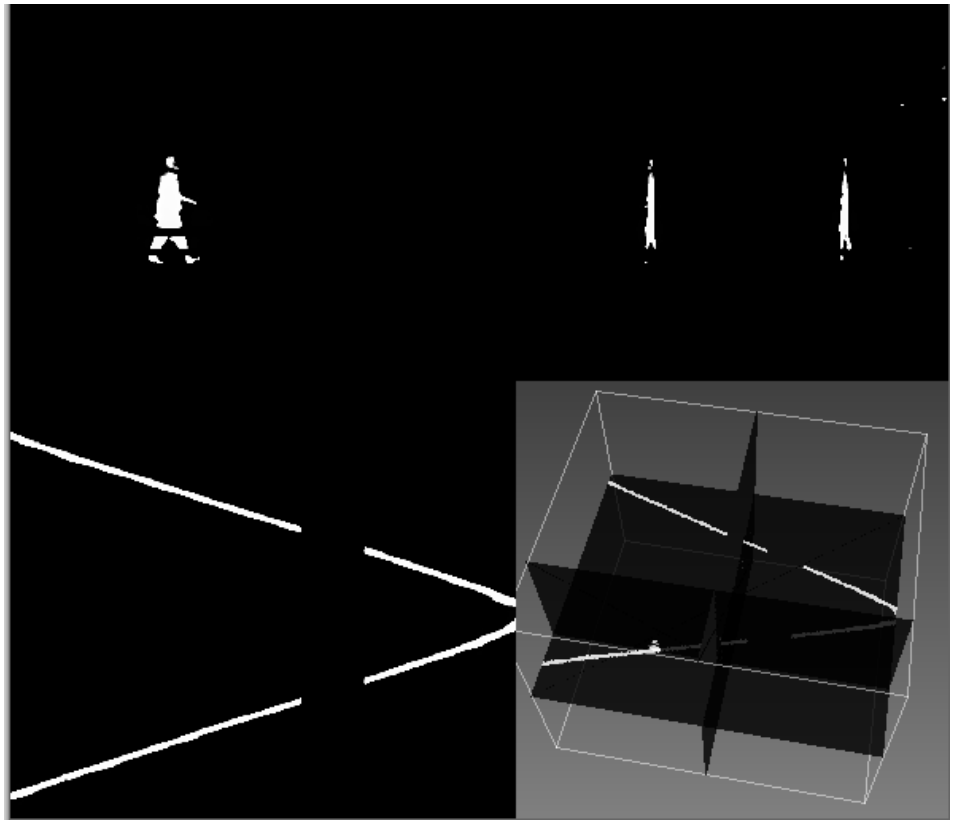

Figure 1.7: Deriche filter is applied on a video volume to extract the foreground (White pixels constitute the foreground).

1.4.2 Multiple Object Tracking

The task of multiple object tracking refers to the detection and tracking of moving objects (of a particular category) in a scene. This task is usually tackled in two steps :

- **Detecting Objects** : An object specific detector is applied to obtain detection hypotheses in a video.
- **Data Association** : Detection hypotheses are associated to compute trajectories for objects. An object trajectory refers to a time ordered set of object detections.

In this thesis, we focus on the *data association* aspect of a multiple object tracker.

Video volume opens up new perspectives in object tracking. Recent tracking approaches in the state-of-the-art report the usefulness of video volumes in managing occlusions and maintaining precise trajectories for objects.

Figure 1.4 shows an object detector's response (*a.k.a.* detection hypotheses) on an image frame. Spurious and missed detections along with imprecise localizations are the main challenges for multiple object tracking. Detection hypotheses form interesting spatio-temporal patterns (*e.g.* cliques in [Zamir 2012]) and in the recent past there has been an increase in interest in finding these patterns to compute object trajectories.

We address the problem of object tracking by utilizing the spatio-temporal structure around the detection hypotheses.

1.5 Contributions

This thesis brings out the following contributions :

1.5.1 Video volume segmentation with Fibers

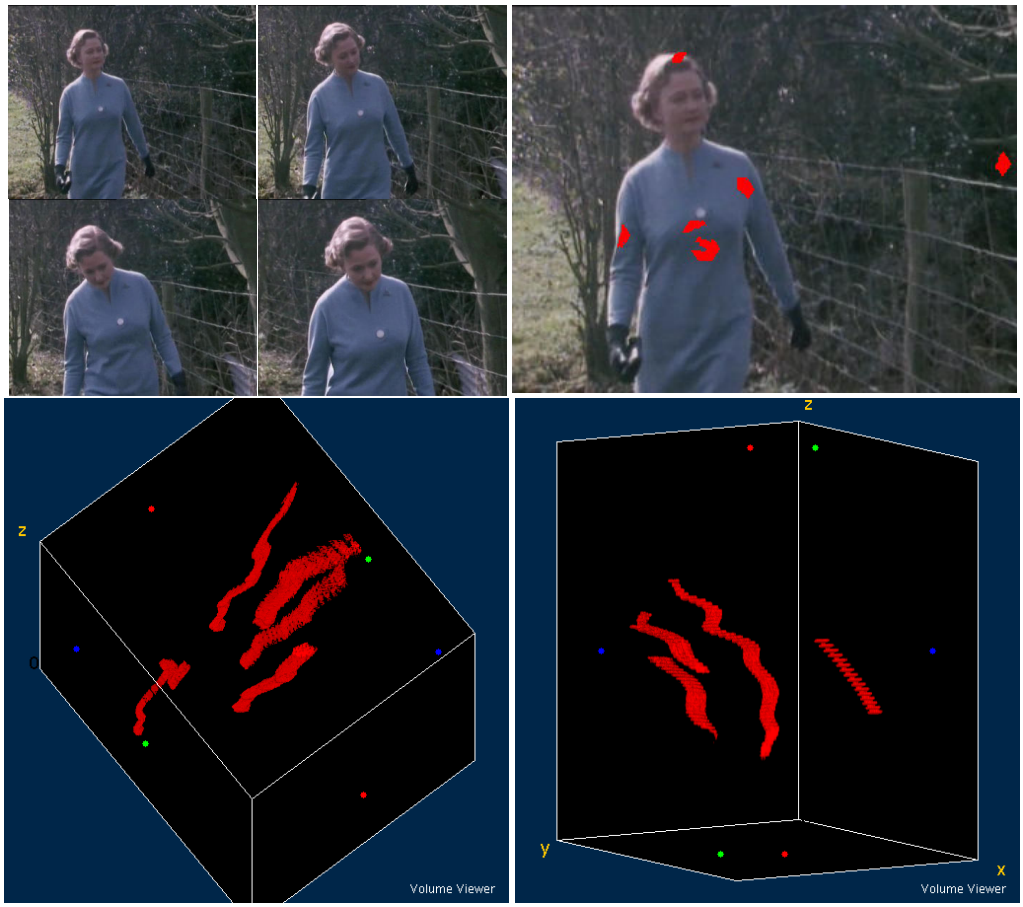

Figure 1.8: Examples and visualization of fibers: **Top Left** images show a few frames from a Marple video [Brox 2010]. **Top Right** shows sample fibers in the first frame of the video. **Bottom Row** images display two snapshots from a 3D visualizer, considering the video as a 3D volume (2D+T).

With respect to the state-of-the-art in video segmentation, the two contributions are as follows :

- We combine both spatial and temporal aspect of video into a single notion, **Fiber**. A fiber is a set of trajectories spatially connected by a triangular mesh. Some examples

of sparse fibers can be seen in Figure 1.8. Fibers are a mid level entity and serve the purpose of bridging the gap between low-level pixels and high-level activity recognition.

- Another contribution is the incorporation of hierarchical clustering into segmentation, with meshes. For domains like action recognition, it is often desired to keep the finest representation in term of fibers (long term dense optical flow) while for domains like background segmentation or foreground estimation (in freely moving cameras) a much coarser representation can be selected.

1.5.2 Multiple object tracking using efficient graph partitioning

Another contribution of this thesis is in the domain of multiple object tracking. We conduct trajectory computation on a video volume. An off-the-shelf person detector is used to obtain probable locations of persons for all frames of this volume. A graph is formed with detections hypotheses as the nodes, and suitable edges are incorporated to encode suitable affinity functions using spatio-temporal cues. The multi-object tracking is then modeled as a graph partitioning problem, such that a part corresponds to trajectory of a person.

This thesis brings out following contributions in this domain :

- Formulating tracking as an energy minimization problem, wherein the natural constraints of object tracking are dealt in a principled manner.

- A computationally efficient combination of optimizers to achieve a near real time tracking performance, along with maintaining competitive results with the state-of-the-art.

1.5.3 Distance covariance based descriptor for appearance matching

A minor contribution of this thesis is in developing a novel descriptor for appearance matching. We consider the task of identifying persons across multiple cameras. This task is popularly known as re-identification. Appearance matching for the person re-identification is often based on the *covariance* of several features such as intensity, gradients and wavelets. However the covariance matrix C(X,Y) cannot express reliable correlations if the random variables X,Y are non-linearly dependent. Recently [Szekely 2009] proposed a novel coavriance measure termed *Distance Correlation*, which can compute all degrees of possible relationships between random variables. We proposed a descriptor based on distance correlation which outperforms previous state-of-the-art covariance descriptors on re-identification datasets.

1.6 Thesis Structure

This complete thesis is divided into three parts. The first part provide details about *Video Segmentation*, and subsequently the second part elaborates on *multiple object tracking*.

The final part concludes this thesis by summarizing the contributions and indicating future research directions.

The chapters are organized as follows:

Chapter 1 introduces the motivation and objective of this thesis.

Part 1: Video Segmentation

Chapter 2 provides an overview of the state-of-the-art in video segmentation.

Chapter 3 describes the first major contribution of this thesis *i.e.* video segmentation using *Fibers*.

Chapter 4 presents quantitative and qualitative evaluations, and comparisons with the state-of-the-art for video segmentation using fibers.

Part 2: Multiple Object Tracking

Chapter 5 outlines state-of-the-art approaches in multiple object tracking.

Chapter 6 describes the second major contribution in the domain of multi-object tracking.

Chapter 7 presents qualitative and quantitative results along with comparisons from state-of-the-art for multiple object tracking.

Chapter 8 provides detail on appearance matching using distance correlation. This chapter will conclude with quantitative results and discussions on the appearance descriptor applied to tracking and Re-Identification.

Part 3: Concluding the thesis

Chapter 9 concludes this thesis by outlining future research directions and perspectives on video segmentation & multiple object tracking.

Part I

Video Segmentation

CHAPTER 2

Video Segmentation: Related Works

Contents

2.1 Image Segmentation 20
2.1.1 Seeded Region Growing 20
2.1.2 Graph Based Image Segmentation 21
2.1.3 Superpixels 23
2.2 Video Segmentation without correspondences 23
2.2.1 Segmentation with Object Proposals 24
2.2.2 Hierarchical graph based video segmentation 28
2.3 Video Segmentation with correspondences 31
2.3.1 Clustering of Point Tracks 31
2.3.2 Point Tracks and Hierarchical Video Segmentation 34
2.3.3 Tube or mesh based video representation 35
2.3.4 Motion layer segmentation 35
2.4 Conclusion 37

Video segmentation has been an active area of research since 1980s. Current work on video segmentation can be broadly classified into two categories: *frame-by-frame* based approaches and *volume-based* approaches. Frame-by-frame approaches take as input one or two successive frames of a video, while volume-based approaches consider many successive frames of a video at once. Frame-by-frame approaches have the advantages of low memory requirement while volume-based approaches in recent years have demonstrated good coherency in object labeling over time. Frame-by-frame approaches are also termed as *streaming algorithms* as they can be directly applied to a video stream. Recently with the increase in computers' memory, volume-based approaches have gained importance, as jointly processing all frames of a video brings more information and helps maintaining segmentation or trajectory coherency over time. However the accuracy of current video segmentation algorithms still needs to be improved.

Video segmentation approaches combine various cues from motion, color, to provide spatially and temporally coherent output. An object or part of an object in a video can be identified based on the similarity of color. However the presence of texture, or inhomogeneous appearance of objects *e.g.* persons wearing different shaded clothing, adds to the complexity in analyzing the extent of objects. In these cases temporal information from various frames of a video can be exploited to identify the object or its parts. For example

an object in a scene with similar color as the background can be identified based on the difference in motion w.r.t. background.

Motion provides vital cues regarding the grouping of pixels, and the existence of similar motion for pixels implies their similar fate *i.e.* similar grouping (Gestalt principle).

A key component for many vision algorithms is the estimate of long term correspondences for pixels in a video. This long term correspondency for a pixel is also referred to as *point trajectory* in the computer vision literature. Dense point trajectories are required for finely analyzing the videos. For example, a gesture recognition system characterizing different hand movements will require temporal details about the movement of the hand, *i.e.* long term correspondences for pixels constituting the hand. A video segmentation algorithm can be broadly categorized into two classes depending on the motion estimates provided:

- Segmentation **without** correspondences: Approaches belonging to this category provide grouping of pixels as objects or object-parts across the video. This grouping provides no detail about long term motion of components (pixels) constituting the group. In plus of color information, most of these approaches use local motion information computed from optical flow. These algorithms are sought out for application like video tooning, object extraction.

- Segmentation **with** correspondences: These approaches provide trajectories for most pixels of the video. Application areas include activity/gesture recognition, video compression, object extraction.

2.1 Image Segmentation

Before we delve into video segmentation approaches, it is important to mention some reliable image segmentation techniques which form the basis for several state-of-the-art methods in video segmentation.

2.1.1 Seeded Region Growing

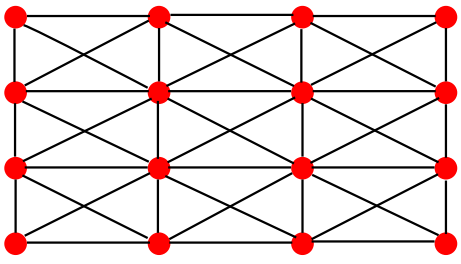

Figure 2.1: An image grid with an eight-neighborhood system

A region growing approach starts with an incomplete segmentation (*i.e.* set of regions which does not cover the full image) and attempts to aggregate the unlabeled pixels to one of the regions. The initial set of regions are also known as seeds. A pixel is merged

Figure 2.2: Different automatic seed selection sources. **Left :** Input Image. **Center :** Seeds obtained from homogeneity criteria. **Right :** Seeds obtained from edges in Input image (Seeds are shown as white regions).

to a seed if it fits a predefined criterion based on color similarity. Seeds can either be found manually or in an automatic manner. In manual seed selection, a user annotates the object or its parts by scribbles. On the other hand, an automatic selection of seeds can be performed by information from image gradients, edges as performed by [Köthe 1995]. Some automatic seed selection techniques are shown in Figure 2.2. Once these seeds are located, optimization approaches like Dijkstra [Köthe 1995] or graph cuts [Boykov 2006] can be employed to obtain the final segmentation.

2.1.2 Graph Based Image Segmentation

The core motive of this work is in finding criteria which define non local characteristics of an image.

Figure 2.3: Image showing the importance of non local criterion which can segment above into three regions. There are three segments in an image with a rectangular-shaped intensity ramp in the left half, a constant intensity region with a hole of a high variability rectangular region.

Unlike the classical methods in image segmentation (such as split and merge), this method adaptively adjusts the segmentation criterion based on the degree of variability in neighboring regions of the image. This results in a method that, while making greedy decisions, can be shown to obey certain non-obvious global properties. Figure 2.3 from

[Felzenszwalb 2004] shows that a purely local criteria is insufficient to distinguish the three regions. Specifically, in order to obtain a segmentation of the image in Figure 2.3, breaking edges with large color variations will never be able to find a segmentation of the high variation region on the right.

Another important aspect of this work is its computational efficiency. The complexity of the approach is quasilinear *i.e.* O($ElogE$), where E is the number of edges. Hence this approach is practically more amenable than its counterparts in image segmentation *e.g.* Normalized cuts by [Shi 2000] whose complexity is quadratic in terms of number of edges of the graph.. A sample output from this segmentation algorithm is shown in Figure 2.4.

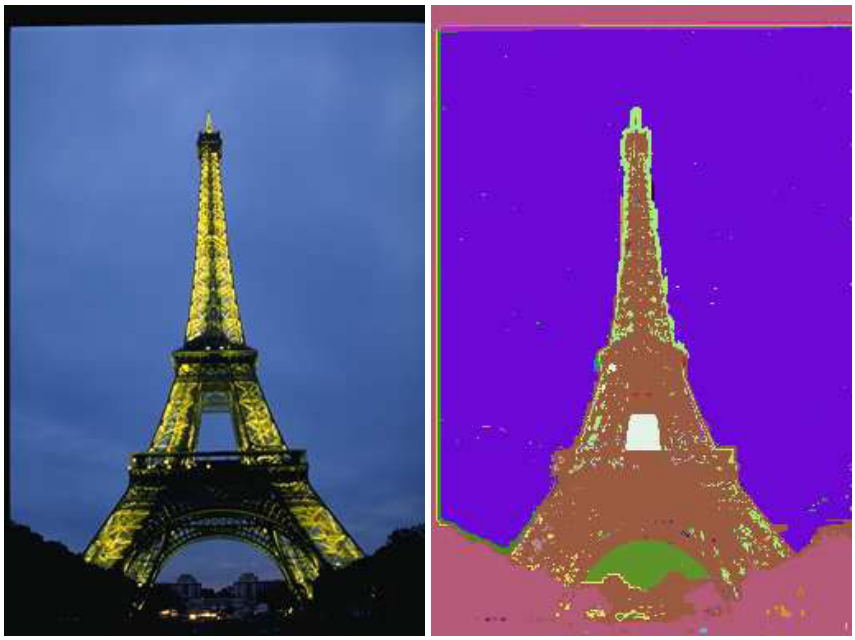

Figure 2.4: Segmentation result obtained by [Felzenszwalb 2004].

Implementation

An image is viewed as a graph (V, E), where vertices (or nodes) V correspond to the pixels of an image and edges E are defined by four or eight neighborhood connectivity, as shown in the Figure 2.1. Edges are weighted according to the color difference between the nodes they connect. The segmentation process begins with the completely disconnected graph, edges are added one at a time in increasing order of their weights. Prior to adding an edge, there are as many connected components as the number of pixels in the image. An edge addition decreases the number of connected components. The process maintains a forest of Minimum Spanning Trees (MST) for the current components. During an edge addition, each MST C_i is associated with a threshold $T(C_i) = w(C_i) + k/|C_i|$, where $w(C_i)$ is the maximum weight in the spanning tree of the component C_i and $k > 0$ is a constant.

An edge e, whose endpoints belong to the different components C_i and C_j is added if and only if

$w(e) <= min(T(C_i), T(C_j))$, where $w(e)$ is the weight of the edge.

This process is a reminiscent of the famous Kruskal's Minimum Spanning Tree algorithm. The parameter k for fusing nodes (or adding edges) can be varied to obtain hierarchical segmentation outputs ([Grundmann 2010]). At the lowest level (small value for k) in the hierarchy, an image is segmented to many regions and the number of regions decreases by increasing the value of k.

2.1.3 Superpixels

Superpixels are atomic clusters of pixels which are perceptually meaningful. Superpixel computation has become an important pre-processing step for several image processing and vision tasks. Instead of working on a large graph of pixels, many high level inference tasks extract superpixels in an image to reduce the size of the problem. This provides a significant gain in speed as the number of superpixels for an image is typically 1000 times lesser than the number of pixels. Notice that the lowest level hierarchy (if appropriately chosen) of the approach by [Felzenszwalb 2004] can also provide superpixels.

One of the most popular superpixel algorithm is by [Achanta 2012] and is referred to as SLIC superpixels. Sample images from SLIC are shown in Figure 2.5.

Figure 2.5: Sample superpixels from [Achanta 2012] of sizes 64, 256 and 1024.

2.2 Video Segmentation without correspondences

Each moving part of an object carves a distinct sub-volume in the 2D+T cube formed by the video. A viable approach for segmentation would be to directly extract or model these sub-volumes. [Niyogi, S.A. and Adelson 1994] is one of the first works on analyzing the

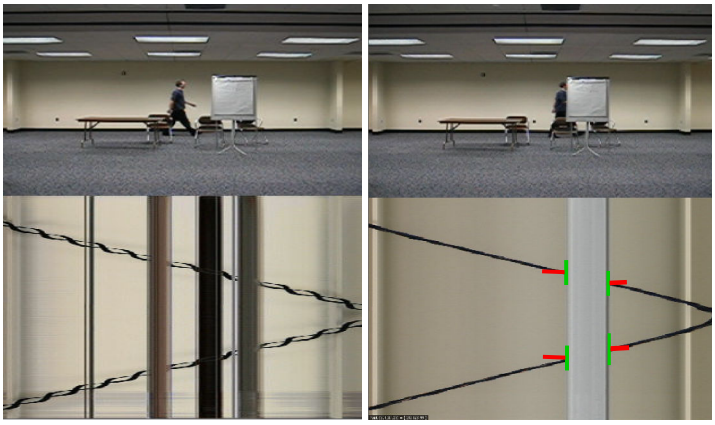

Figure 2.6: **Left:** Braided Pattern, **Right:** T-Junctions (marked in red/green).

patterns formed in video volumes. The observation here is that the human gait forms a typical braided pattern in the space time slice. Left image in Figure 2.6 shows this pattern when the person is walking at a constant speed and the direction of walk is lateral relative to the camera. The authors [Niyogi, S.A. and Adelson 1994] model this gait pattern using four snakes.

Occlusions can be detected in videos by using spatio-temporal slices. T-junctions (*cf.* Figure 2.6) formed on spatio-temporal slices indicate the presence of an occlusion [Apostoloff 2006]. [Apostoloff 2006] learns to detect T-junctions using a support vector machine. This sparse occlusion edge information is used to build a prior about objects and subsequently the video is segmented using graph-cuts.

Both of the above approaches use information from the spatio-temporal slices instead of the full volume. Authors in [Ristivojević 2006] search for the volume carved out by a moving object in the video volume. The problem of segmenting is viewed as a volume competition. This is a generalization of many approaches developed for region segmentation in images. A level set framework is used to parameterize the surfaces that need to be estimated.

A recent work on extracting human volumes is proposed by [Niebles 2010]. For each image frame candidate regions are obtained by a detection step and the refinements of these regions are based on shape priors. Temporal propagation of identified regions is performed in a level set framework and subsequently the complete optimization for the extraction of human volume is performed by belief propagation.

The limiting factor to all these approaches lie in their assumptions such as linear camera motion, or in strict requirements for the direction of movements. Another limitation of the above approaches is their computational speed, as most of these require time in order of minutes per frame to obtain a segmentation.

2.2.1 Segmentation with Object Proposals

Recently there has been an increase in interest in developing approaches for extracting objects in videos starting with *initial proposals* for objects. The proposals delineate the ob-

ject of interest for a frame in a pixel-wise manner. Usually proposal generation algorithms provide multiple proposals for a single frame (*cf.* Figures 2.9, 2.8).

Most of the video segmentation approaches under this category employ the object proposal algorithm by [Endres 2010]. Hence we review this approach for generating object proposals in the following section before proceeding to video segmentation approaches utilizing these proposals. Other diverse objects proposal generation infrastructures can be obtained form [Alexe 2012], [Ziming 2014].

Category Independent Object Proposals

Given an input image, the algorithm from [Endres 2010] provides a ranked list of diverse **category independent** candidate object proposals. The ranking is done based on *objectness* such that high ranked proposals are likely to be a good segmentation.

The first step for generating category independent proposals (*cf.* Figure 2.7) is to compute a hierarchical segmentation of the image. The authors here use occlusion boundary extraction from [Hoiem 2007b] and various geometric contexts for non-planar vertical surface of [Hoiem 2007a]. Long range spatial interactions between regions of images are captured by considering affinity for pairs of regions to lie on the same object. An object proposal is then generated by choosing one of the regions to seed the segmentation, and thereafter computing the probability that each other regions belong to the same object as this seed. In order to make reliable object predictions from larger regions while maintaining the flexibility of a superpixel-based segmentation, the authors construct a graph of superpixels and transfer the affinities obtained previously onto this graph. This process is then repeated to obtain a bag of proposals for an image.

The suppression of undesired proposals is obtained by ranking these proposals (using Structured SVM [Tsochantaridis 2004]). The ranker incrementally adds proposals from best to worst, based on the combination of an object appearance score and a penalty for overlapping with previously added proposals. Sample outputs from [Endres 2013] are shown in Figures 2.9 and 2.8.

Motion Coherent Tracking using MRF by [Tsai 2011]

[Tsai 2011] proposed a semi-supervised approach to object extraction in videos wherein a user annotates the contour of the object which needs to be tracked for the whole sequence. Another contribution of this work is to provide the *SegTrack* dataset which includes pixel-wise groundtruth for all frames of the video.

The authors adopt a volumetric MRF formulation, in which a video volume is represented as a multi-label MRF with hidden nodes corresponding to the unknown labels. The resulting optimization problem is to find a joint label assignment L for all pixel sites in the volume. The energy term quantifies the spatial and temporal coherency. The label assignment l_p to a pixel p encodes the following :

- An attribute which gives a segmentation label indicating foreground or background for p.

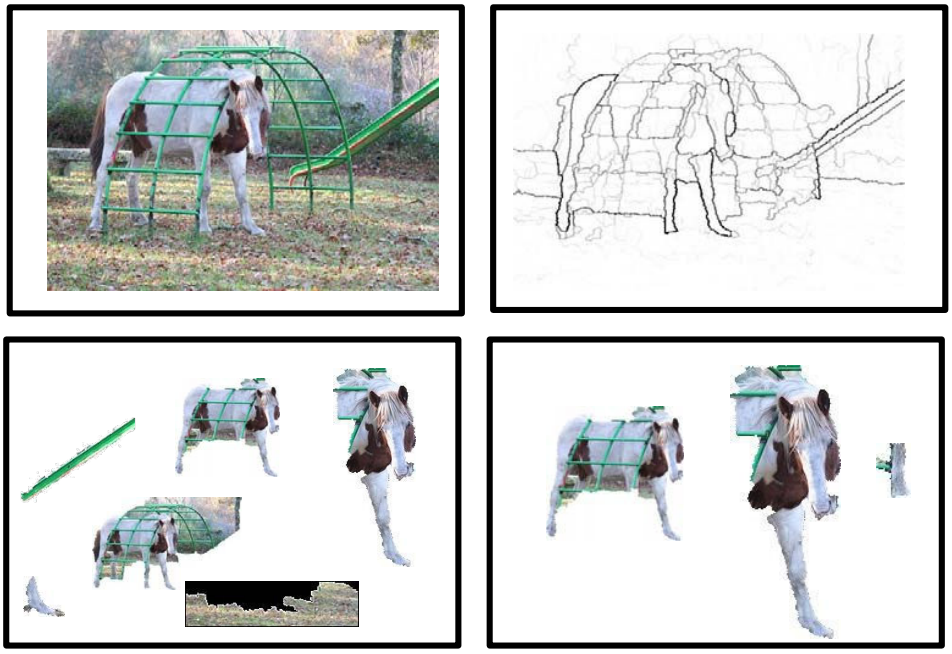

Figure 2.7: Pipeline: Object Proposal Generation by [Endres 2010]. **Top Right - Bottom Left, Bottom Right:** Compute a hierarchical segmentation, generate proposals and rank proposed regions. At each stage, classifiers are trained to focus on likely object regions and encourage diversity among the proposals, enabling the system to localize many types of objects. (**Image source: [Endres 2013]**).

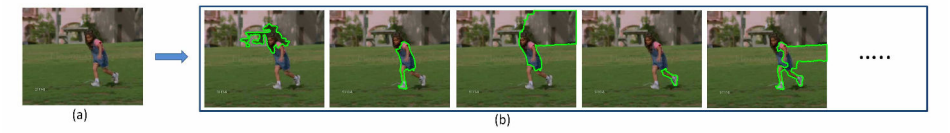

Figure 2.8: Object proposals on a sample SegTrack video dataset (source [Tianyang 2012]).

Figure 2.9: Object proposal with *objectness* score.

- A displacement which gives the offset from p to the corresponding pixel in the next frame.

The MRF optimization is performed by the Fast-PD library from [Komodakis 2007]. Sample outputs on SegTrack dataset by [Tsai 2011] are shown in Figure 2.10.

Figure 2.10: Segmentation results from [Tsai 2011] on the SegTrack dataset.

Segmentation using category independent proposals

Approaches such as [Lee 2011], [Tianyang 2012] use object proposals (generated from [Endres 2010]) to find a set of diverse segmentation for each image in the video. Once the proposals are generated for each frame of a video, then the problem of video-object segmentation is posed as an optimization problem wherein the objective is to find a segmentation that defines a video in the best manner, and subsequently refine the foreground contour to fit the object properly.

As opposed to [Tsai 2011], thanks to the object proposals from [Endres 2010], the works by [Lee 2011], [Tianyang 2012] can work in an unsupervised manner.

The approach by [Lee 2011] first identifies object-like regions (key segments) in any frame according to both static and dynamic cues. Subsequently a series of binary partitions among the candidate "key segments" are computed to discover hypothesis groups with persistent motion and appearance. Finally a pixel level object labeling across the full video is obtained using a *gaussian mixture model* based foreground extraction. One of the core ideas of this approach is to quantify a motion based foreground region confidence for a region r, and pixels $\bar{r}$ around it in a loosely fit bounding box. This is given by the following equation 2.1, where $\chi^2_{flow}(r, \bar{r})$ is the χ^2-distance between L_1 normalized optical flow histograms:

$$M(r) = 1 - exp(-\chi^2_{flow}(r, \bar{r})) \tag{2.1}$$

This cue describes how the motion of the proposal region differs from its closest surround-

ing regions and allows to forgo assumptions about camera motion, while also providing allowance to different magnitudes of motion. The authors [Lee 2011] have shown in experiments that this is indeed a vital cue for computing key segments in videos, and appearance alone is insufficient.

Another approach to video segmentation using object proposals is by [Tianyang 2012] which differs from [Lee 2011] mainly in the usage of the optimization model to find object. The object selection problem is expressed as finding maximum weighted cliques in a weighted graph G, where each region proposal corresponds to a node. A clique in an undirected graph is a subset of its vertices such that every two vertices in the subset is connected by an edge. Given the affinity matrix for the nodes in G, the aim of [Tianyang 2012] is to find the heaviest subset of nodes which form a clique. The task of finding cliques in a graph is NP hard, and even approximating the clique number within a factor of O($n^{1-\varepsilon}$) is an NP hard for any $\varepsilon > 0$ [Zuckerman 2007]. The authors in [Tianyang 2012] relax the problem from a discrete to a continuous domain and compute the solution in a iterative manner. The approach is indeed a reminiscent of the Lagrange relaxation technique used in many optimization problems.

As with many other relaxation techniques, one obvious drawback of this approach, from a vision system perspective, is that it needs multiple initializations, and hence multiple runs to arrive at a good solution. On a PC with 3.4 GhZ and 8 GB RAM it takes 2 minutes to select regions on a typical 100 frame SegTrack dataset video, with 50 different initializations. However the result obtained by this optimization model outperforms the earlier approaches by [Lee 2011] and [Tsai 2011] on the SegTrack dataset.

Fast video segmentation

While the diverse object proposals are necessary to extract foreground and background, the low computational speed of algorithms providing object proposals limits the applicability to low resolution videos. The object proposal generation from [Endres 2010] computes proposals at a speed of 120 seconds per frame, *i.e.* 3000 times slower than real time (frame resolution : 400x230).

In order to gain speed, the authors of [Papazoglou 2013] proposed a rapid technique to produce a rough estimate of which pixels are inside the object based on motion boundaries in pairs of successive frames. This initial estimate is then refined by integrating information over the whole video with a spatio-temporal extension of GrabCut [Rother 2004]. This stage automatically bootstraps an appearance model based on the initial foreground estimate, and uses it to refine the spatial accuracy of the segmentation and to also segment objects in frames where it does not move. Figure 2.11 shows the workflow of this technique.

2.2.2 Hierarchical graph based video segmentation

This work [Grundmann 2010] is inspired from the graph based image segmentation by [Felzenszwalb 2004]. The graph here corresponds to the complete video volume (*cf.* Figure 2.12). The neighborhood size of a pixel is 26, 9 in each of the previous and next frames,

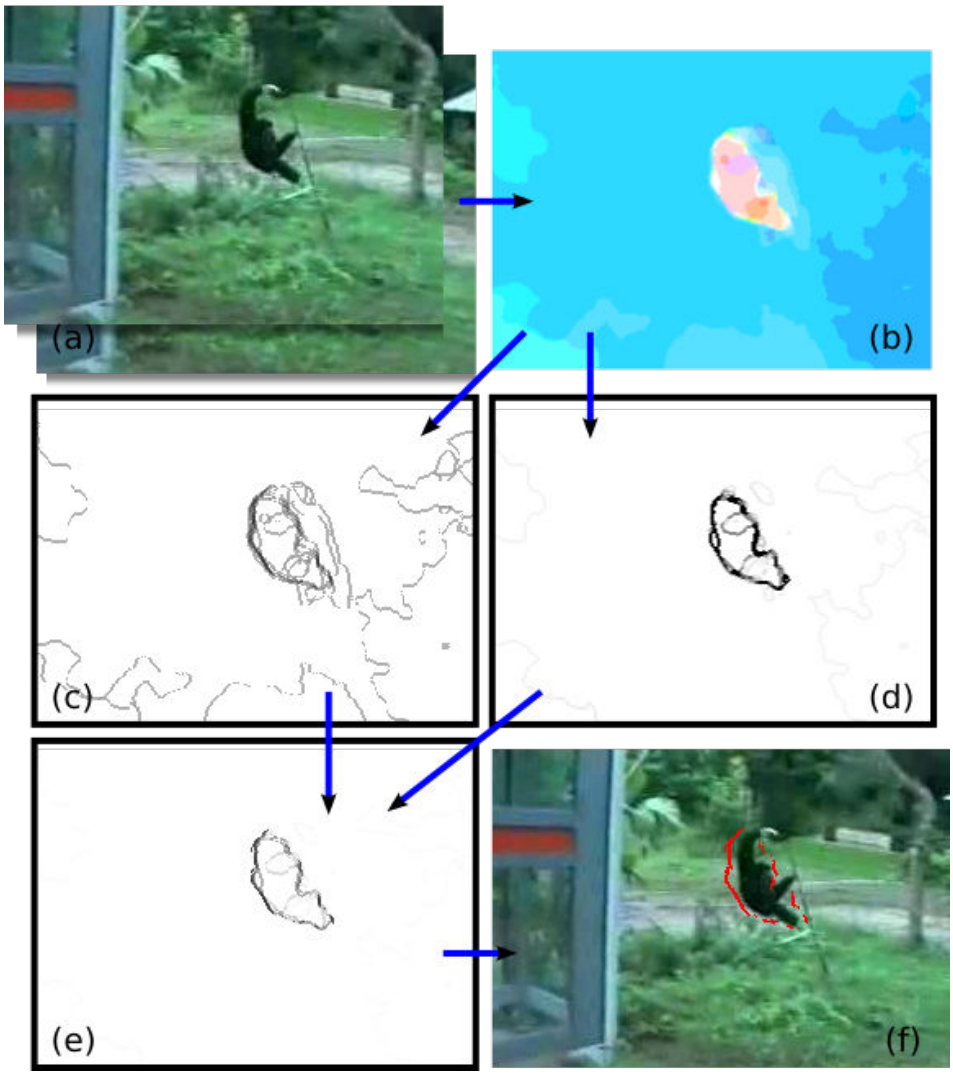

Figure 2.11: Motion Boundaries. (a) Two Input Frames. (b) Optical flow. The hue of a pixel indicates its motion direction and the color saturation its velocity. (c) Motion boundaries based on the gradient of the optical flow. (d) Motion boundaries based on the difference in the motion direction between a pixel and its neighbors. (e) Combined motion boundaries. (f) Final motion boundaries after thresholding. (**Image source : [Papazoglou 2013]**).

and 8 in the current frame. The authors show that using optical flow to determine neighborhood gives better results. This is mainly due to the fact that the standard 26 neighborhood system will not be able to capture the displacement of more than one pixel.

Some notable differences of this approach from the image based segmentation of [Felzenszwalb 2004] :

- The control parameter for minimum segmentation-region size is varied to ob-

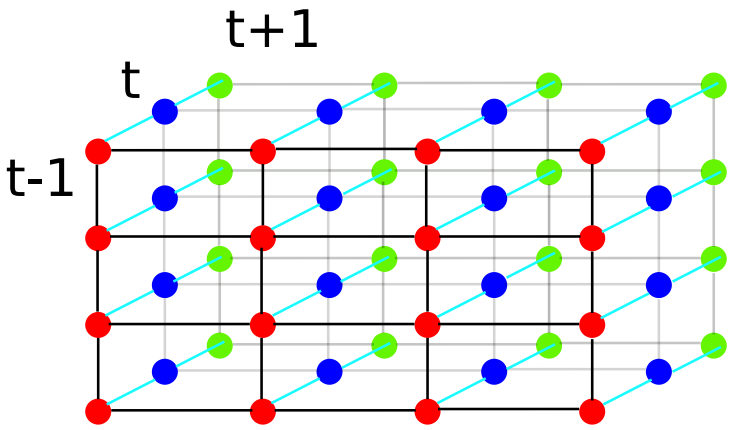

Figure 2.12: A Video grid.

tain hierarchical outputs, instead of being fixed to a particular value as in [Felzenszwalb 2004].

- Color and motion histograms are used to define a region, and the weights for edges are determined by the χ^2 distance between the histograms of the regions connected by edges.

Sample outputs for this segmentation is shown in Figure 2.13. This work is the most popular video segmentation algorithm in the recent years and outperforms other state-of-the-art segmentation techniques [Xu 2012].

Figure 2.13: Image frames from different videos and corresponding outputs at a particular hierarchy. (**Image source : [Grundmann 2010]**)

2.3 Video Segmentation with correspondences

Approaches in this category use long term point correspondences *i.e. point tracks* to obtain a coherent spatio-temporal segmentation. Owing to the recent advances in optical flow algorithms, point tracks building has become more reliable and has achieved a good level of maturity. Most notable track building algorithms are from [Sundaram 2010] and [Sand 2008]. Point tracks from [Sand 2008] are also referred to as Particle Video. Another commonly used point tracker is from [Tomasi 1991], which is commonly known as the *KLT tracker*. Authors of [Sundaram 2010] have shown that point tracks produced by optical flow of [Brox 2011] are of much higher quality than the ones produced by KLT and particle video, in terms of dense coverage and temporal length. However computing point tracks from [Sundaram 2010] is slow on CPUs. Significant computational speedup can be achieved by employing GPUs, and many open source implementations for real time KLT trackers are available [Zach 2008].

2.3.1 Clustering of Point Tracks

The segmentation techniques in this category use point tracks as the main source of information. The point tracks are computed by [Sundaram 2010]. Techniques in this category differ mainly in the manner in which the clustering is performed to extract objects (*i.e.* labeling point tracks). Point tracks provide only a sparse coverage due to unreliable motion estimates in color homogeneous regions. Hence algorithms in this category generally provide grouping for only a subset of pixels in the form of point tracks, typically covering around 3% of the total pixels in a video. Some notable trajectory clustering algorithms are from [Fradet 2009], [Brox 2010], [Fragkiadaki 2011], [Brostow 2006]. We review two recent ones in this section.

The authors of [Brox 2010] cluster point tracks by spectral clustering [Luxburg 2007]. Two trajectories should belong to the same object is they have similar motion and are close-by in space. This is expressed by equation (2.2), where T corresponds to trajectories, $O(T_i, T_j)$ denotes the overlap span of T_i, T_j. The 2D spatial location at time t is represented by x_t, and d_{sp} represents the average spatial euclidean distance between T_i and T_j.

$$\mathrm{d}(\mathrm{T_i}, \mathrm{T_j}) \propto \frac{1}{|\mathrm{O}[\mathrm{T_i}, \mathrm{T_j}]|} \left(\sum_{\mathrm{t} \in \mathrm{O}[\mathrm{T_i}, \mathrm{T_j}]} ((\mathrm{x_t^i} - \mathrm{x_{t+1}^i}) - (\mathrm{x_t^j} - \mathrm{x_{t+1}^j}))^2 \right) \mathrm{d_{sp}}(\mathrm{T_i}, \mathrm{T_j}) \quad (2.2)$$

A sample output of this approach is shown in Figure 2.14.

Another popular approach is by [Fragkiadaki 2011]. Point track clustering techniques do not need information from a detector about object hypothesis, and hence are referred to as detection free tracking by [Fragkiadaki 2011]. Given point tracks, the authors [Fragkiadaki 2011] first classify these trajectories as background or foreground by defining a saliency measure. Background trajectories are dropped from further consideration. A graph is built whose vertices are foreground trajectories. Weights for edges are attractive and repulsive forces between trajectories. Attractive forces encode motion similarity between trajectories, while repulsive forces are derived by the topology of foreground maps.

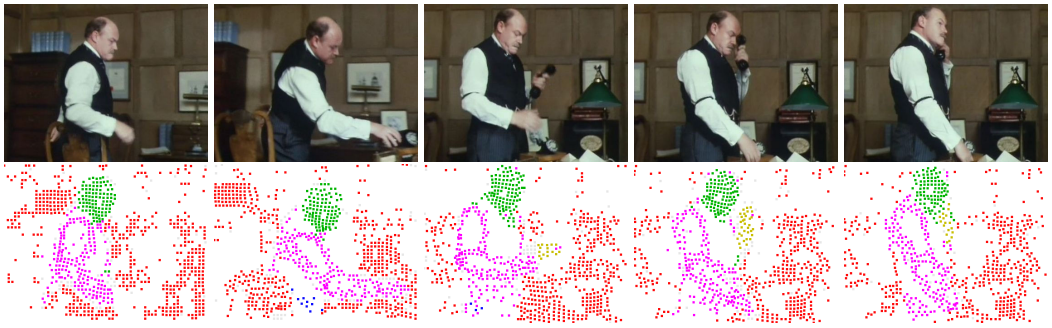

Figure 2.14: **First Row** : Sample input image frames from marple13 sequence by [Brox 2010]. **Bottom Row**: A sparse segmentation obtained by grouping of pixel trajectories (by [Brox 2010]). White zones in images do not belong to any segments.

A foreground topology is obtained by figure-ground segmentation by the initial labeling of trajectories into foreground and background. This foreground topology is a good indicator of when two trajectories cannot be grouped together. Specifically two trajectories cannot be grouped together if they belong to different foreground components.

On this graph, the clustering into objects is performed by Normalized cuts [Yu 2001]. This approach outperforms the above mentioned approach by [Brox 2010] on the dataset from [Brox 2010].

Figure 2.15: Output from [Ochs 2011]. This approach uses the sparse labeling information from [Brox 2010] (**Bottom Left**) and utilizes information from superpixels and gradients to obtain the final labeling (**Bottom Right**).

[Ochs 2011] extend the approach by [Brox 2010] to provide a dense segmentation of a video. On the results by [Brox 2010], the authors use a hierarchical variational model that propagates these labels preferably in homogeneous areas which were initially not covered by point tracks. This dense segmentation obtains a better classification accuracy than the input sparse segmentation and overcomes the over-segmentation issue of static image segmentation approaches. The workflow of [Ochs 2011] is shown in Figure 2.15.

Background subtraction using Point Tracks

Figure 2.16: Background Subtraction in freely moving cameras from [Sheikh 2009]. **First Row**: Input image frames. **Second Row**: Clustered point tracks with blue representing foreground trajectories. **Third Row**: Estimated foreground labeling. **Fourth Row**: Groundtruth for foreground separation.

Background subtraction in moving camera scenarios can also be performed using point tracks and notable works in this aspect are [Elqursh 2012], [Sheikh 2009]. The latter builds a sparse background model by estimating a compact *trajectory basis* from trajectories of salient features across the video. The background is then subtracted by removing trajectories that lie within the space spanned by this basis. From this sparse background information, a foreground background appearance model is built, and an optimal pixel-wise binary labeling is obtained by solving a MAP inference problem. The principle of building appearance models from sparse information is in sync with the approaches in object extraction in a video mentioned above by [Tianyang 2012], [Lee 2011]. An example of background subtraction can be seen in Figure 2.16.

2.3.2 Point Tracks and Hierarchical Video Segmentation

Point tracks encode important temporal information and hence can be used as seeds in a seeded region growing (*cf.* sections 2.1.1, 2.1.2). Graph based image and video segmentation (GBS) [Felzenszwalb 2004, Grundmann 2010] and seeded region growing (SRG) [Köthe 1995] algorithms share a common workflow to image or video segmentation.

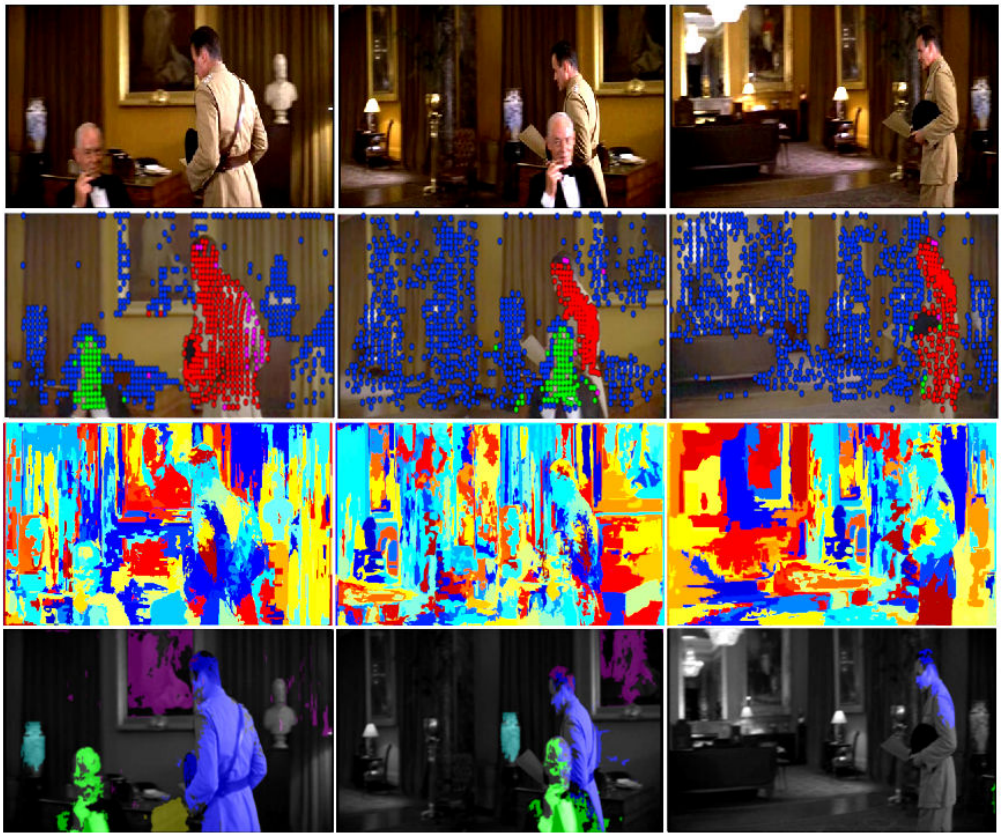

Figure 2.17: Output from [Lezama 2011]. **First Row**: Input image frames. **Second Row**: Clustered point tracks with colors denoting separate clusters. **Third Row**: Dense segmentation output. **Fourth Row**: The ground truth of object regions (left) which is automatically propagated by [Lezama 2011] to other frames (middle and right). **Image Source**: [Lezama 2011].

Both these approaches work on the weighted graph whose vertices correspond to pixels on the image grid, and edges are determined by 8 (or 26)-neighbor connectivity for a pixel. Edges are weighted according to a suitable similarity measure between nodes.

SRG starts with initial seeds and sort edges in decreasing order of edge weights. On the other hand GBS proceeds by ranking all edges in a similar fashion. Notice that in SRG, due to the presence of seeds, some regions are already fused. Hence it makes sense to incorporate some seed information using point tracks into the graph based video segmentation by [Grundmann 2010]. This is explored in video segmentation by [Lezama 2011] and [Palou 2013].

[Palou 2013] and [Lezama 2011] use point tracks as seeds on a video grid graph. More precisely, vertices belonging to a point track are assigned one label thus defining a seeded region. Subsequently [Palou 2013] and [Lezama 2011] use video segmentation algorithm by [Grundmann 2010] to obtain a hierarchical segmentation of the video. Both these approaches have shown better temporal coherency than without using the information from point tracks. The approach by [Lezama 2011] is semi-supervised, as it requires information about the exact number of objects in the scene. Sample outputs from [Lezama 2011] can be seen in Figure 2.17.

2.3.3 Tube or mesh based video representation

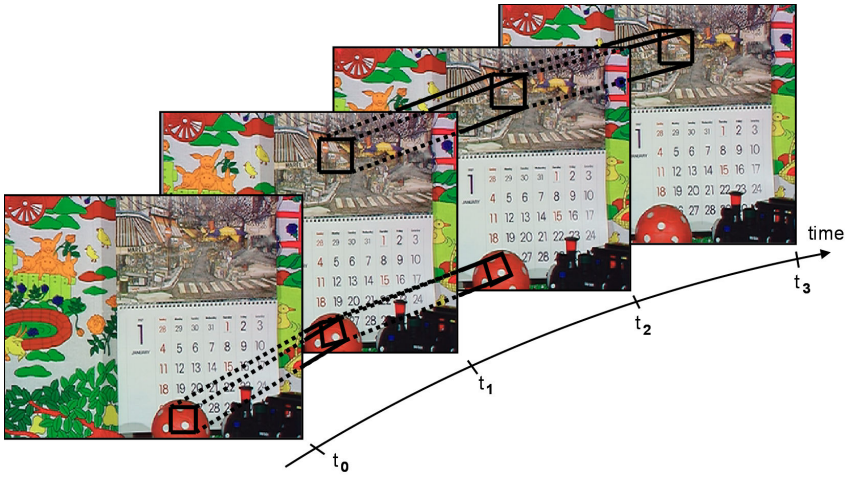

Figure 2.18: Motion tubes based representation from [Urvoy 2009]. **Image Source**: [Urvoy 2009].

Motion tubes aim at building a video representation by finding correspondences of patches (2D square blocks) across time (*cf.* Figure 2.18). By tracking textures throughout the sequence, motion tubes offer a compact representation of a video. However overlapping tubes present challenges of managing tubes and it is difficult to decide whether a tube is required for a representation or not. Although simple to implement, motion tubes do not segment an object precisely due to *block* based nature of the tubes, and do not yield pixel precise correspondences for pixels inside a patch.

Recent works by [Jain 2014], [Trichet 2013a] design tube based representations for high level vision applications such as action recognition.

Instead of building 2D square patches along time for a tube based representation, works such as by [Marc 2002] employ 2D meshes to obtain a representation for a video.

2.3.4 Motion layer segmentation

Motion segmentation aims at partitioning videos into regions (layers) of coherent motion. Classical optical flow formulation enforces local spatial smoothness for estimating optical flow, and hence does not exploit the fact that a large region can have a constant mo-

tion. Motion field in natural scenes consists of piecewise homogeneous motion, hence modeling the motion field as a piecewise a parametric field gives stronger prior to the optical flow problem [Unger 2012]. Thus spatial regularity in optical flow formulation leads to inaccurate flow field. This inaccuracy can be handled by *jointly* estimating motion and performing segmentation of a scene, and is the main focus of the approaches aiming to partition a video into layers of motion. Some of the prominent works are by [Wang 1994], [Ayer 1995], [Odobez 1998], [Gelgon 2000] ,[Smith 2004], [Cremers 2004], [Vazquez 2006], [Sun 2010], [Gelgon 2005]. [Unger 2012].

[Wang 1994] is one of the first approaches to motivate a motion layer segmentation, and a layer segmentation is performed by clustering (K-Means) regions based on pre-computed affine motion of the scene. [Ayer 1995] employ minimum description length principle to simultaneously compute motion models and spatial support in an Expectation Maximization framework. A drawback of these clustering methods is their inability to model spatial coherency in estimation for spatial support. Acknowledging these inadequacies, Odobez et al. [Odobez 1998] present a motion segmentation algorithm based on 2D polynomial motion models, a multiresolution robust estimator to compute these motion models, and appropriate local observations supplying both motion relevant information and their reliability. A segmentation is then formulated as a contextual statistical labeling problem exploiting multiscale Markov Random Field models. The authors [Odobez 1998] also introduce a detection step which allows to estimate and update the number of required motion models and thereby handling the appearance of new objects across image sequence.

Cremers et al. [Cremers 2004] present a variational approach to piecewise parametric motion segmentation. [Cremers 2004] derive a cost functional which depends on parametric motion models for each of a set of regions and on the boundary separating these regions. This functional can be viewed as an extension of the famous Mumford-Shah functional [Mumford 1989] from intensity segmentation to motion segmentation. As opposed to computationally expensive alternating minimization based approaches, the problems of motion estimation and segmentation are jointly solved by continuous minimization of a single functional.

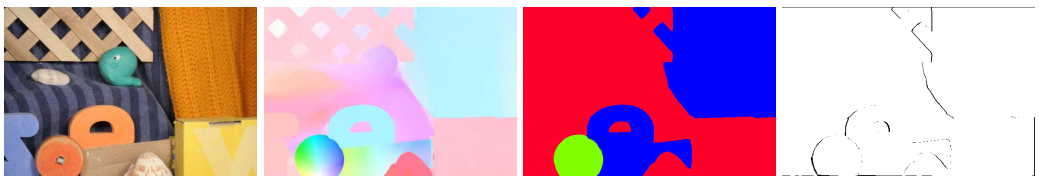

Figure 2.19: Sample motion layer segmentation and relative depth ordering from [Sun 2010]. **First** image shows an image frame. **Second** image shows flow field, **Third** image displays the segmentation output. The colors indicate depth ordering with green for front, blue for middle and red for back. Detected occlusions are shown in the **Fourth** image. **Image Source**: [Sun 2010].

Vazquez et al. [Vazquez 2006] proposed a variational approach for joint motion estimation and segmentation by *basis function* representation of motion and level set evolution. The components of motion in each region of segmentation are represented as functions in a

space generated by a set of basis functions. The coefficients of the motion components considered combinations of the basis functions are the parameters of the representation. This is in contrast with the work by [Cremers 2004] wherein the motion parameters are either estimated prior to segmentation or are considered variables independent of the boundary curve.

Layered motion segmentation also make reasoning about occlusion relationships easier and approaches such as [Sun 2010], [Unger 2012] look to incorporate long term temporal consistency and also provide depth ordering. Figure 2.19 shows a sample output from [Sun 2010]. Instead of focussing on motion smoothness assumption, the authors of [Sun 2010] focus on incorporating temporal *layer consistency*, arguing that while scene motion can change rapidly, scene structure persists over time.

Unger et al. [Unger 2012] proposes a variational method in continuous setting with a similar aim as [Sun 2010] to jointly formulate a model for segmentation, optical flow, depth ordering and occlusions. An advantage of this method over other approaches is that it can deal with hundreds of labels on GPU cards. With more labels the authors demonstrate the advantage of their approach over [Sun 2010] for obtaining meaningful segmentations. However the run time of the proposed approach is too slow and ranges from a few minutes to an hour for a couple of frames on GPU.

A common drawback for the above methods is the run time for operation. On a typical image of size 640x480 pixels, estimating layered segmentation for 4-5 layers could take minutes to hours. [Sun 2010] provides highly accurate optical flow, layer segmentation and relative depth ordering for a few layers at a whopping 5 hours run-time for two frames. Schoenemann et al. [Schoenemann 2006] proposed an approach for *binary* motion segmentation with close to real time performance at 17fps on 320x240 frame sized video.

2.4 Conclusion

In this chapter, a review of popular video segmentation approaches was presented. We selected a few state of the art approaches which form the basis for other proposed approaches in video segmentation and showed a snapshot of the information provided by them (*cf.* Table 2.1). From a high level vision algorithm perspective, it is important to consider the relative amount of information provided by the state-of-the-art approaches. Vision systems such as video tooning normally do not require temporal correspondences, while most other systems such as action recognizers require temporal correspondences for most pixels in the video [Aggarwal 2011]. Other example of systems requiring long range temporal correspondences are semantic label propagation, object tracking. Quite often, it is also desirable to have a set of association factors for each pixel expressing its segmentation reliability to neighboring segmented-regions.

Motivated from the need for temporal correspondences, the proposed video segmentation algorithm in this thesis provides the following information:

- A spatio-temporal neighborhood system to extract meaningful information. Approaches based on point tracks have no spatial coherency in the segmentation, and hence it is difficult to extract reliable spatio-temporal statistics.

Method	Correspondences	Dense Coverage	Reliability Factor
[Brox 2010]	✓	×	×
[Grundmann 2010]	×	✓	×
[Tsai 2011]	×	✓	×
Ours	✓	✓	✓

Table 2.1: State-of-the-art approaches and the information provided by them. **Correspondences** column indicate long term information about a pixel's motion across the video. **Dense Coverage** column indicates if the segmentation approach partitions the video fully *i.e.* each pixel is associated to a particular segment. **Reliability Factor** column shows if the segmentation approach provides a per-pixel factor expressing the long term quality of segmentation.

- Dense temporal correspondences for all pixels in a video. Approaches such as [Brox 2010], [Ochs 2011], [Palou 2013] provide correspondences for about 3% of the video only.
- Last but not the least, we provide a reliability factor per pixel, which is based on long term temporal assessment of a pixel's association w.r.t. its segmented region.

CHAPTER 3

Video Segmentation: Fibers

Contents

3.1 Introduction . . . 39
3.2 Fibers : Definition and Approach . . . 40
3.2.1 Formalization . . . 40
3.2.2 Criteria for a good representation . . . 41
3.2.3 Approach outline . . . 42
3.3 Sparse reliable fibers . . . 42
3.3.1 Straight fibers . . . 43
3.3.2 Initiating fibers at corners . . . 47
3.4 Full video coverage . . . 51
3.4.1 Geodesics between the sparse fibers and rest of the video . . . 52
3.4.2 Enforcing trajectory coherency . . . 53
3.5 Hierarchical representation . . . 57
3.6 Computational complexity of the fiber extension stage . . . 59
3.7 Summary . . . 59

3.1 Introduction

Optical flow estimation and video segmentation are often regarded as two different tasks in computer vision. While optical flow focuses on point-wise correspondences between frames, video segmentation is the extraction of temporally coherent regions from a video, without point-to-point temporal correspondences. The emerging need for long-term trajectories as well as for more detailed video segmentation is asking for the unification of the two fields, and requesting suitable structures to bridge the gap. The combination of these two kinds of information is needed to analyze finely videos and is a prerequisite to precise video understanding. For instance, relevant information for gesture recognition can be extracted from the identification of body parts, from their motion as well as their shape variations [Aggarwal 2011].

With respect to the state of the art in video segmentation, our contributions are:

- a new point of view on **video representation**, with a structure handling together point trajectories and hierarchical segmentation with object meshing : **fibers**,

- an iterative process to build these fibers, which can be seen as an **optical flow robustifier**, making it **reliably dense and long-term**,
- a new approach to **video segmentation**, with small complexity (quasi-linear) and thus **near real-time**.

We formalize the problem in section 3.2, build fibers in sections 3.3, 3.4, 3.5, and finally show experiments in the next chapter (section 4.2).

3.2 Fibers : Definition and Approach

A video volume is the stack of successive video frames. Fig. 3.1 shows slices of a video volume, displayed as 3D volumic scans in medical imaging. A pair of *red* colored lines mark the correspondences in the slices. The top-left image displays a standard 2D image frame from the stack, while two other spatio-temporal cuts, in planes (t, y) and (x, t), are shown in the top-right and bottom left. In this 3D representation, points on the static background form straight lines of homogeneous intensity over time, while points on moving objects form curved lines. Analogically to fibers in MRI images of human brains, we term *fibers* these straight or curved lines. We are interested in a dense estimation of fibers, involving all pixels of a video.

3.2.1 Formalization

A **video volume** $V = (I_t)_{t \in [1,n]}$ is a stack of n successive image frames I_t, each of which is defined over a same domain $\Omega \subset \mathbb{R}^2$. It can be seen as 3D data, parameterized by $(x, y, t) \in \Omega \times [1, n]$.

A **fiber** $\mathbf{F} = \left(\{T_i\}_{i\in[1,m]}, \mathcal{M}\right)$ is a set of m trajectories T_i, spatially connected with a triangular mesh $\mathcal{M}$. The Figure 3.2 shows an illustration of a fiber formed by meshed trajectories. Each trajectory T_i is a sequence of locations $\mathbf{x}_i^t \in \Omega$ during a time span $[t_s^i, t_e^i] \subset \mathbb{N}$, and thus writes $T_i = (\mathbf{x}_i^t)_{t\in[t_s^i,t_e^i]}$. The mesh $\mathcal{M}$ is a planar graph whose vertices are trajectories, and whose edges connect spatially-close trajectories, in a triangulated way.

A **regular fiber** satisfies moreover that its trajectories (T_i) do not cross each other. More precisely, the trajectory mesh $\mathcal{M}$ induces at any time t, on the set of locations $\left\{\mathbf{x}_i^t\right\}_{i\in[1,m]}$, a triangular mesh $\mathcal{M}_t$ over the region occupied by the fiber in image I_t. These meshes $\mathcal{M}_t$ are required to be spatially coherent in time, *i.e.* no triangle should be flipped from a frame to the next one.

Note that fibers may have subpixelic spatial precision. Motions from a frame to the next one are usually not congruent with the pixel grid indeed, and discretizing them spatially would make trajectories suffer from aliasing. Thus the frame domain Ω is considered as continuous.

Our aim is to search for a **partition** of the video volume into a set of regular fibers, optimizing the criteria below.

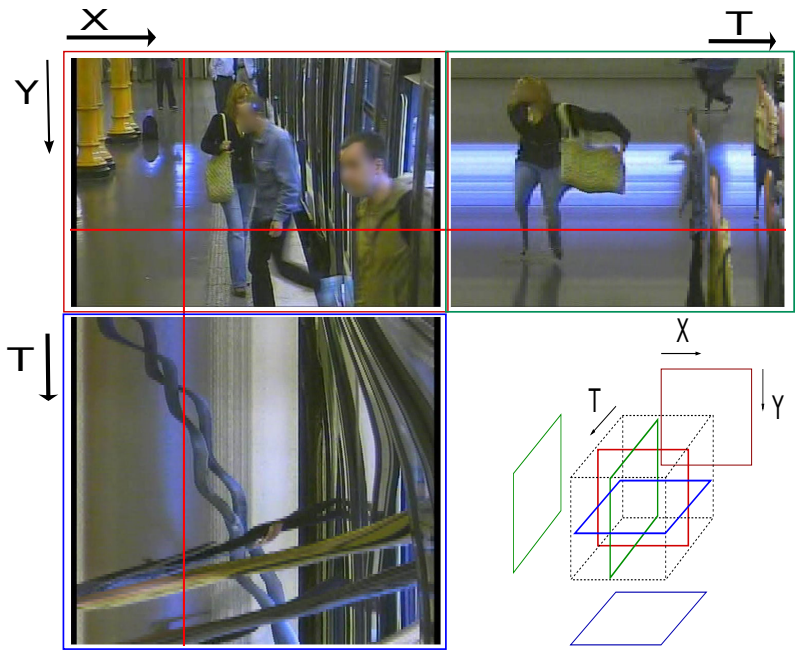

Figure 3.1: 2D+T video volume represented as a 3D data, as in medical imaging: frontal, sagittal and horizontal slices correspond to cuts along planes (x, y), (y, t) and (x, t) respectively. The Colored boxes around frames are in sync with the colored boxes in right bottom images. The red lines indicate the values of x and y chosen for the cut. This video shows people exiting a train in a train station. Note in particular the lines in the (x, t) slice, formed by people trajectories, by the train or the background. We name these sets of lines *fibers*.

3.2.2 Criteria for a good representation

The following three traits were given in [Köthe 1995] to characterize a reliable image segmentation:

(C_1) **Region homogeneity.** The segmentation should provide regions that are homogeneous w.r.t. one or more properties, *i.e.* the variation of measurements within the regions should be considerably less than variation at the borders.

(C_2) **Pixelic precision on edges.** The position of the borders should coincide with local maxima, ridges and saddle points of the local gradient measurements.

(C_3) **No oversegmentation.** Areas that perceptually form one region should not be split into several parts.

to which we add:

(C_4) **Time coherency.** The video representation should provide high coherency in time *i.e.* the identities of objects/pixels should not change while moving across time.

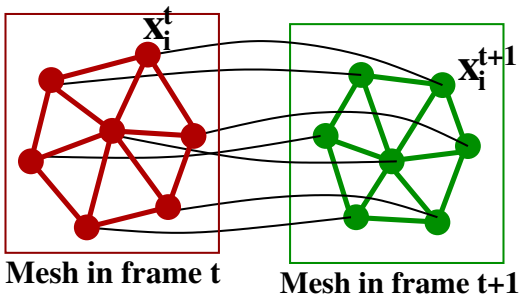

Figure 3.2: A mesh of trajectories induces a mesh of points in each frame.

(C_5) **Robustness and reliability.** The representation should not be very sensitive to noise, and its reliability should be estimated, in order to know which parts can be safely trusted and which are debatable.

Criterion (C_1) expresses the homogeneity of each fiber, *e.g.* internal color or motion coherency. Criterion (C_2) stresses that differential information, such as intensity gradients, is useful locally to reach pixelic precision. We will make use of structure tensors and of corner detectors to estimate reliable and precise correspondences. Criterion (C_3) will be dealt with by merging neighboring fibers of similar color or trajectory. Criterion (C_4) asks for regular fibers with long time-spans. Last but not least, criterion (C_5) encourages statistics over neighborhoods instead of considering single local value only, and promotes reliability estimation, which may actually be expressed from such statistics. Fibers are well-designed for this, as their meshed-trajectories structure can be seen as spatial & temporal neighborhoods, upon which statistics can be easily computed. The instantiation of these 5 criteria will be detailed in the next sections.

3.2.3 Approach outline

We propose to search for the best partition of the video volume iteratively, by:

1. **suggesting candidate fibers** at locations where motion estimation is reliable (*e.g.* corners), **selecting** the best ones (most coherent in time, longest), while **improving them** by making them regular (no triangle flip) (section 3.3),
2. **extending them** jointly (to cover the full video domain) in such a way that each video point is assigned a **valid trajectory** (section 3.4),
3. **merging** similar fibers hierarchically (section 3.5).

The next sections in this chapter will follow this order.

3.3 Sparse reliable fibers

Fibers can be detected by finding correspondences across the video volume. Many existing techniques, such as optical flow or descriptor matching, can serve this purpose. However all

these methods are unreliable in homogeneous areas and suffer from the notorious aperture problem on boundaries. Hence we first identify video regions where the correspondences are likely to be reliable (corners), then we check the quality of the trajectories in these areas while simultaneously improving their regularity. Each fiber thus built is then associated with a reliability factor.

For a stationary camera scene, points (x, y) on a static background form straight lines parallel to the time axis. We term these straight lines of homogeneous intensity *straight fibers*. Fibers which do not belong to the class of straight fibers, are termed *curved* fibers. In our approach pipeline, we first detect these straight and curved fibers at corner-like regions. Subsequently we extend these detected fibers to the other parts of the video. The following sections will provide details on detecting straight and curved fibers.

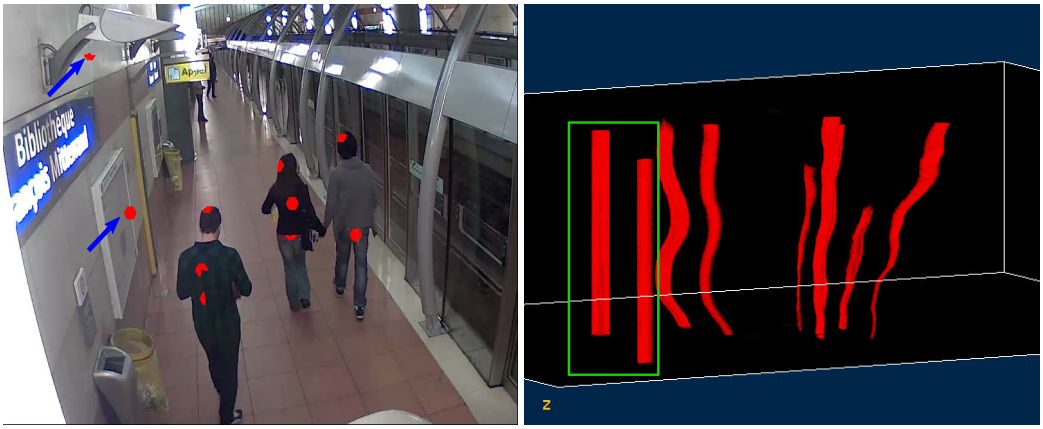

Figure 3.3: Sample fibers. **Left:** Red colored patches show sample fibers on one frame. **Right:** Sample fibers are displayed by a 3D visualizer (ImageJ). The Green box encloses two *straight fibers* which are formed by the patches on the background. Patches forming straight fibers are marked by the blue arrows on the left image.

3.3.1 Straight fibers

This section aims at providing a flavor of fibers, by introducing the process flow for obtaining straight fibers. For a static camera scene, points on the static background form straight lines parallel to the time axis. These straight lines (parallel to the time axis) which are spatially connected on a regular 2D grid, are termed *straight fibers*. Figure 3.3 shows sample straight and curved fibers.

Temporally long straight fibers are good candidates for the background. Thus a plausible approach for background detection could be to search straight fibers. A background removal aids in reducing the computational burden of the complete segmentation operation, wherein regions belonging to the background can be ignored for further processing.

Naive pixel matching across time is not robust due to intensity flickering in time. A better approach could be to correlate successive image patches, thereby exploiting spatial statistics while matching.

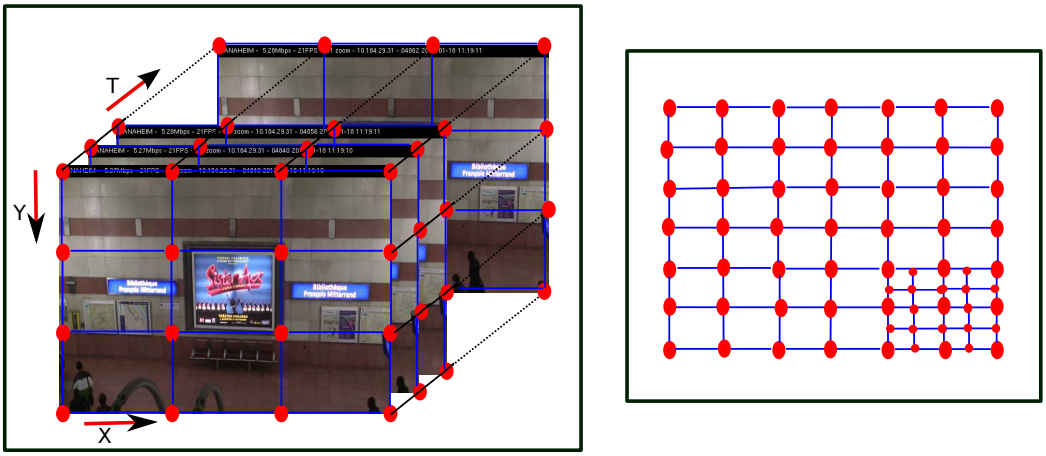

Figure 3.4: Straight fibers search. **Left**: A set of successive frames are shown, overlaid with a 2D rectangular grid. **Right**: A finer resolution grid is employed if the color correlation in coarser resolution grid is insufficient for building straight fibers.

In order to detect straight fibers, we lay a 2D grid on all image frames in V (*cf.* Figure 3.4). Straight fibers are then built using color correlation of temporally successive grid cells (patches). Color correlation of two patches P, Q is shown in equation (3.1), where $P(\mathbf{x})$, $Q(\mathbf{x})$ represent colors at 2D locations $\mathbf{x} \in (x, y)$. $\bar{P}$ denotes the mean color of a cell P.

$$CC(P,Q) = \frac{\sum_{\mathbf{x}\ \varepsilon\ Patch} (P(\mathbf{x}) - \bar{P}) \cdot (Q(\mathbf{x}) - \bar{Q})}{\sqrt{\sum_{\mathbf{x}\ \varepsilon\ Patch} (P(\mathbf{x}) - \bar{P})^2} \cdot \sqrt{\sum_{\mathbf{x}\ \varepsilon\ Patch} (Q(\mathbf{x}) - \bar{Q})^2}} \tag{3.1}$$

Grid cells sharing the same 2D spatial co-ordinates, but located on successive image frames form 2D+T cuboids (*cf.* Figure 3.4). A coarse grid in Figure 3.6 is chosen for explanatory purposes. We search for straight fibers in those cuboids which have the same temporal span as that of the video. Let T_1 denote the temporal length of the video, then we refer to the 2D+T cuboids of temporal length T_1 as 2D+T_1 cuboids. Figure 3.4 shows *nine* 2D+T_1 cuboids which have the same temporal span as that of the video. For each of these *nine* cuboids, we search for a time span such that the sum of color correlation (SCC) of contiguous grid cells is maximized (*cf.* equation (3.2)). P, CC in the following equation (3.2) corresponds to an image patch and color correlation respectively.

$$SCC(P, [t1, t2]) = \sum_{t=t1}^{t2} CC(P_t, P_{t+1}) \tag{3.2}$$

This time span $[t1, t2]$ then defines the temporal span for a straight fiber. Temporal span of a straight fiber is limited, if $CC(P_t, P_{t+1})$ becomes less than a predefined threshold. An example of two straight fibers, present at different time spans and sharing the same 2D+T_1 cuboid is shown in Figure 3.5

If no straight fibers are found for a particular grid size, then we reduce the size of grid cells (set initially to 50x50 pixels) and the search for straight fibers is performed again. This process is repeated in an iterative manner until the grid cell size becomes smaller than 10x10 pixels. Below this grid-cell size, the straight fiber search becomes unreliable due to the inefficiency in obtaining good color correlation of small cells.

The temporal length of a straight fiber is an important cue for assessing its suitability for the background. The longer these fibers are, the higher is their probability of belonging to the background. Hence the reliability of a straight fiber is directly proportional to its temporal length. Straight fibers which have similar temporal length as that of the video time span are highly probable background candidates.

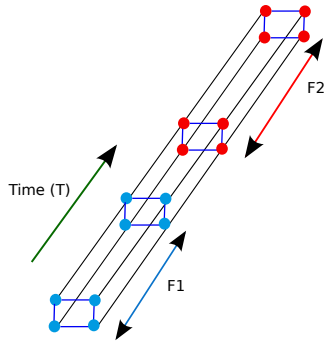

Figure 3.5: Displays two straight fibers F1 and F2. Straight fibers sharing the same 2D+T cuboid and appearing at different time frames, are merged together if they are sufficiently color-correlated.

The span of a straight fiber can be limited due to the occlusions caused by moving objects. Often the background regions appear again in the scene and this lead to pieces of straight fibers appearing at different times, but sharing the same 2D+T_1 cuboid as shown in Figure 3.5. Such fibers sharing the same 2D+T_1 cuboid but appearing at different times can be merged to form one fiber. The decision to merge two fibers into one is subject to the constraint that their mean patches have high color correlation. A mean patch of a fiber is obtained by averaging the color at each pixel-trajectory for its full time span.

Complexity of straight fibers search

This hierarchical grid based straight fiber search requires visiting every pixel in V, H times, wherein H is the number of hierarchy levels. Hence the straight fiber build process is of linear complexity w.r.t. the number of pixels in the video. This simple straight fiber detection can get rid of a lot of background in a static camera video (*cf.* Figure 3.6) and eases the computational burden of the whole chain to compute fibers. For example, straight fibers which are highly reliable can be deemed as background, and points constituting these fibers can be safely ignored for further processing.

Obtaining probable background regions from the background fibers

Thus far we have a sparse set of background straight fibers, and it is of interest to see if we can extend these fibers to the rest of the video. To put it in an another way, for every pixel which do not belong to a fiber, we would like to find a closest background fiber. Intuitively, the pixels which are close to the background fibers in terms of color and location should belong to the background objects, and the pixels which are at a higher distance should belong to the foreground objects. Later sections in this chapter elaborate extending a sparse set of fibers to the rest of the video. For the sake of completeness of exposition to background fibers, let us assume that we have a fiber extension system, and we apply it to extend background fibers in space. Using this fiber extension, we obtain a distance map for the complete video where each pixel has a cost of association (*i.e.* a distance) to the closest background fiber. This distance map is shown in the *third row* of Figure 3.6. The red colored zones in the *third row* of the Figure 3.6 are at a lower distance

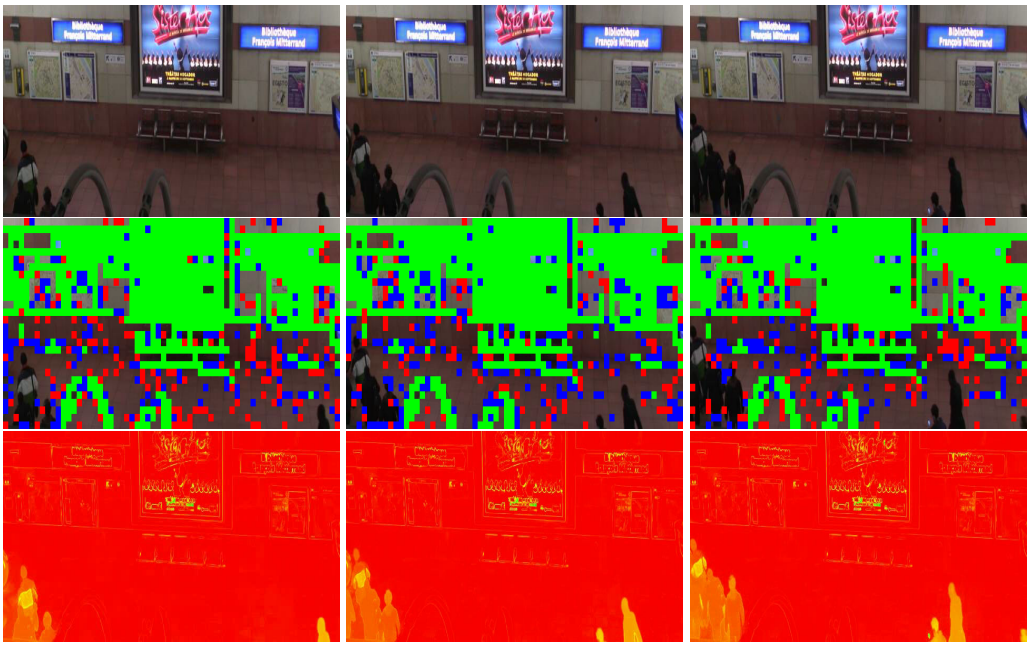

Figure 3.6: Background fibers and the background distance map for a static camera scene. **First Row**: Input frames for background fiber computation. **Second Row**: Straight fibers. Colors indicate the length of straight fibers. *Green* indicates fibers of same temporal length as that of the video. *Red* colored fibers are of very short temporal length (less than 0.5 of the video temporal length). *Blue* colored fibers are of intermediate temporal length between the red and green fibers. **Third Row**: Distance map obtained after extending (*cf.* section 3.4) only green (*i.e.* background) fibers. The *red* color in this distance map indicates lower distance from the background fibers, while lighter colors indicate higher distance from the background fibers. Observe that the moving objects are at a higher distance from the background fibers, than the static regions.

from the straight fibers as compared to the lighter (*yellow*) colored regions. It can be seen

that most of the lighter colored regions belong to the foreground objects. Hence Figure 3.6 demonstrates good background detection capability of the straight fibers as the moving objects are at a higher distance from the background fibers than the background objects.

In the next sections we elaborate on the process flow for detecting a general class of fibers *i.e.* the curved fibers.

3.3.2 Initiating fibers at corners

Curved fibers are built by computing the trajectory of an image patch satisfying the segmentation criteria (C1). Since algorithms for computing correspondences are more reliable in high structure variation regions (corners and edges) than in the homogeneous areas, we build fibers at corners or edges (*cf.* Figure 3.7), using *cornerness* as a reliability factor, defined as:

$$\lambda = \exp\left(\frac{-\gamma}{||\lambda_1 - \lambda_2||_2}\right) \tag{3.3}$$

where λ_1 and λ_2 are the eigenvalues of the structure tensor $\nabla I \times \nabla I$ averaged over the area, and γ is a constant. More details on the *structure tensor* can be found in Appendix C. γ is empirically set to a positive constant, 0.01.

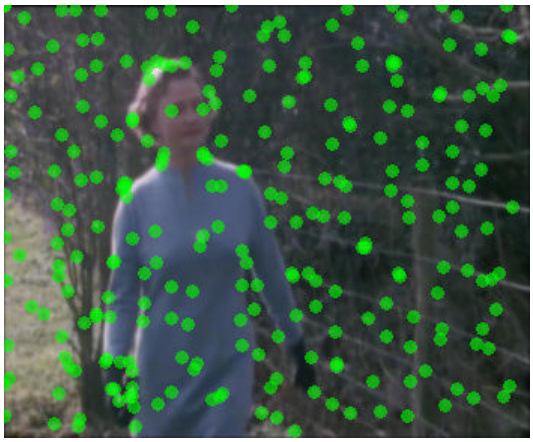

Figure 3.7: Sample interest points in an image frame. We consider a high density of interest points as corner detectors in practice often miss some important corners.

Acknowledging the criteria C5, we define the spatial neighborhood of a corner by laying a high precision sub-pixellic mesh (radius of 10 pixels) around it (*cf.* Figure 3.8). Corners are often located on the external boundary of objects or object-parts, thus involving several different overlapping regions with different trajectories. Consequently we segment a mesh around a corner into multiple regions using color cues. More precisely, we apply K-means (with $K = 2$) in the color space, to the set of colors observed at all vertices of the mesh, in order to know the dominant colors, noted $\big(D(k)\big)$. Then we apply graph cut [Boykov 2006] to achieve a spatial-consistent segmentation of the mesh, assigning, to

each vertex i, one of the k color labels : $L_i \in \{1 \dots k\}$. The energy minimized is :

$$E(L) = \sum_i \|I(\mathbf{x}_i) - D(L_i)\|^2 + w \sum_{i \sim j} \frac{\|D(L_i) - D(L_j)\|^2}{\|\mathbf{x}_i - \mathbf{x}_j\|^2}$$

where as previously $\mathbf{x}_i$ is the location of vertex i and $I(\mathbf{x}_i)$ its color. The weight w is the square of the mesh resolution over the average neighborhood size. The energy is a Conditional Random Field whose interaction terms between neighbors $i \sim j$ are *submodular*, hence graph cuts can be applied. The pairwise terms in a submodular energy function favor smooth solutions where the neighboring labels (*i.e.* outputs) are the same. Hence costs of differing labels to nodes connected by direct edges are greater than costs for agreeing labels to these nodes.

As the number of objects constituting a corner is not known apriori, a mesh is re-segmented until each sub-part satisfies a predefined allowed standard deviation of color.

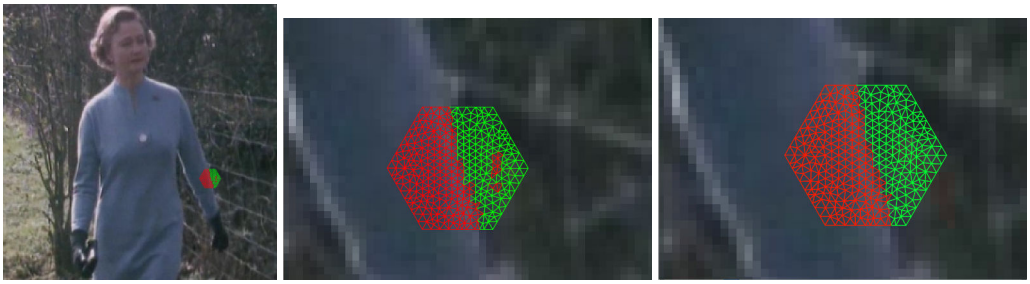

Figure 3.8: **Left**: Corner and its neighborhood. **Center**: K-means in color space. **Right**: Spatial coherency using graph cuts.

Building fibers from meshes

Each of the meshes obtained from a corner is now a separate candidate fiber. Candidate trajectories for each mesh vertex are built time step after time step by simultaneously *following the optical flow*, estimating motion coherency in space and color coherency in time, and correcting the flow if necessary, as described in the next paragraphs. Fibers which are too heterogeneous will be split, stopped or removed. We ensure a minimal density of candidate fibers by selecting the best corners in each sufficiently-large hole in the video coverage by fibers. More precisely, a fiber is initiated at a pixel if at a given time frame, there are no fibers in a 50 pixel radius around it. Thus fibers that are stopped let place in the next frame for new candidates.

3.3.2.1 Color coherency in time

Discontinuity in color at occlusion vicinity is an important cue to avoid fibers switching objects. Successive slices of a reliable fiber should have similar color; this can be measured with:

$$\frac{1}{|\mathcal{M}_t|} \int_{\mathcal{M}_t} \|\partial_t I_t(\mathbf{x}(t))\|^2 \, d\mathbf{x} \simeq \sum_{i \in \mathcal{M}} w_i \left\| I_t(\mathbf{x}_i^t) - I_{t+1}(\mathbf{x}_i^{t+1}) \right\|^2$$

where the integral over the mesh $\mathcal{M}_t$ makes the color variation estimation robust to image noise. Here w_i is a vertex weight standing for the element area (relative area of the neighborhood). The possibility of considering such statistical criteria is an important advantage of fibers, while point tracks alone would suffer from pixelic noise.

3.3.2.2 Motion coherency in space

Similarly, motion should be coherent within a mesh at all times. An incoherency in motion is a potential indicator for a bad mesh segmentation in the fiber initiation step or that a fiber comprising different objects or object-parts with varied movements.

Rather than asking for the motion variance within $\mathcal{M}_t$ to be small, which would penalize wider non-rigid fibers, we ask the motion to be continuous, and penalize its spatial variation :

$$\frac{1}{|\mathcal{M}_t|}\int_{\mathcal{M}_t}\left\|\nabla_{\mathbf{x}}\,\mathbf{m}^t(\mathbf{x})\right\|^2 d\mathbf{x} \;\simeq\; \sum_{i\sim j} w_{ij}\left\|\mathbf{m}_i^t-\mathbf{m}_j^t\right\|^2$$

where $\mathbf{m}_i^t = \mathbf{x}_i^{t+1} - \mathbf{x}_i^t$ is the estimated motion.

Once again, the mesh provided by our fiber representation proves useful to express such kinds of criteria.

3.3.2.3 Regularizing the flow

While self-occlusions are frequent at the object level (*e.g.* a person walking perpendicular to the camera optical-axis), trajectories in a same small candidate fiber (*e.g.* the knee) should not intersect each other. This is equivalent to the requirement that no triangle of the mesh $\mathcal{M}_t$ can be flipped in $\mathcal{M}_{t+1}$.

Another reason for the flipped meshes is that the optical flow algorithms are not perfect and yield wrongly smoothed flow. The flow near boundaries of objects tend to be inaccurate. An example can be seen in Figure 3.9. The mesh shown is on the background and the elongation of this mesh is due to an inaccurate flow near motion boundaries.

Unflipping a mesh can be performed in two ways. If the flipped triangles are strictly inside the mesh, we *smooth* the vertices locations in $\mathcal{M}_{t+1}$ while keeping the mesh boundary constant, until triangles are unflipped. This regularizes spatially the trajectories. If on the contrary, flipped areas include a part of the boundary, we compute instead the best affine movement A that sends $\mathcal{M}_t$ as close as possible to $\mathcal{M}_{t+1}$ for the L_2 norm (closed-form solution for $\inf_A \int_{\mathcal{M}} \|A\,\mathbf{x}(t) - \mathbf{x}(t+1)\|^2\, d\mathbf{x}$), and then replace the original motion with it.

Sample examples for flow regularization can be seen in the Figures 3.9 and 3.10.

3.3.2.4 Fiber termination or split

A lack of color or motion coherency of the mesh $\mathcal{M}_t$ at time step t indicates heterogeneity of a fiber. This can be due to an occlusion, or to a drift of the inaccurate optical flow estimates across the boundary of an object. Allowable maximums for detecting a lack of

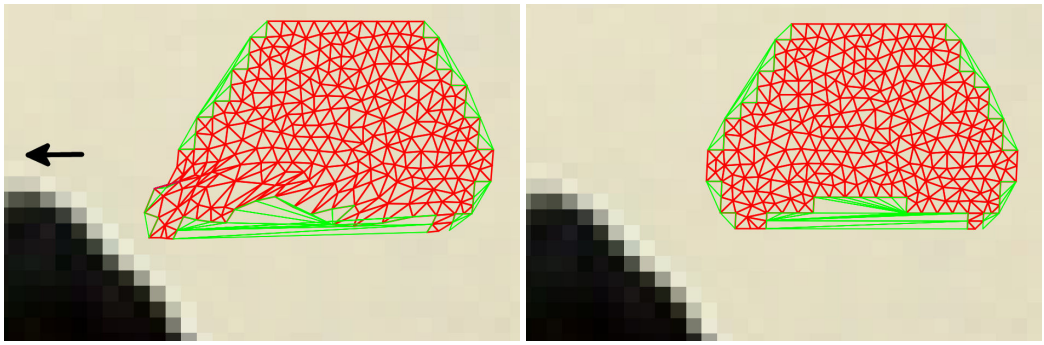

Figure 3.9: **Flow improvement** by regularizing the mesh. Left: $\mathcal{M}_t$. Right: regularized mesh M_{t+1}. The black arrow indicates the direction of the motion of the moving object. The mesh shown is on the background and the elongation of this mesh is due to the inaccurate flow near the motion boundary of the black moving object. Green edges indicate the closing boundaries of this mesh.

color and motion coherency (*cf.* sections 3.3.2.1, 3.3.2.2) are empirically set to 0.1 and 0.02 respectively.

When a lack of coherency is detected, we split the fiber into two homogeneous parts (typically, the two sides of a boundary) if possible, and pursue the build for each of the two fibers independently, provided their spatial size is significant enough. We measure this spatial size in terms of number of points inside a fiber slice, and we empirically set a threshold of 30 points, below which we ignore the fiber. Note that in this case the fibers are fully split, on their full time span, which is possible as we know their trajectories back in time. This is useful when different objects of similar color or motion behave coherently during a while before becoming distinguishable, as then by propagating the information back in time, we are able to distinguish them at all times. The Figure 3.11 shows a *fiber split* illustration wherein a motion in-homogeneity detection helps in knowing the presence of two separate candidate fibers after a certain amount of time.

In order to cover also the drifts that happen too slowly to be noticeable between two successive frames, we add another criterion: the elasticity of the mesh. A too big variation of edge length between any two mesh vertices at any two times (not necessarily consecutive) will call the fiber splitter.

A fiber is stopped, if it cannot be split in homogeneous parts coherent in time (*e.g.* full occlusion). It is furthermore deleted if its time-span thus obtained is too short (less than 10 frames), as the duration of trajectories is a good indicator of their quality.

With the employment of the above steps for all corners in a video, we obtain a set of reliable fibers at high structure variation regions of a video.

Intermediate sample output

Figure 3.12 displays the output fibers at this intermediate step of our fiber building chain. The full representation of video in terms of fibers will now need the extension of these quality fibers to the rest of the video.

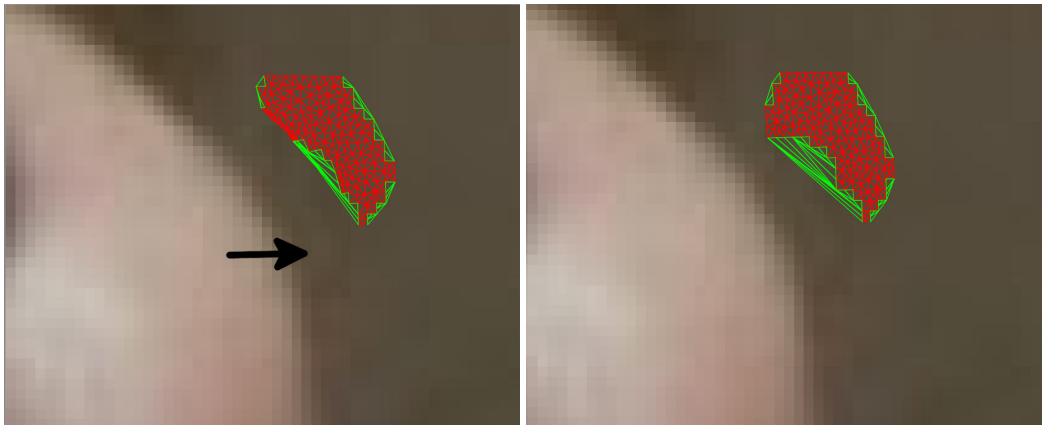

Figure 3.10: **Flow improvement** by regularizing the mesh. Left: $\mathcal{M}_t$. Right: regularized mesh M_{t+1}. The black arrow indicates the direction of the motion of the moving object. The mesh shown is on the background and the elongation of this mesh is due to inaccurate flow near the motion boundaries. Green edges indicate the closing boundaries of this mesh.

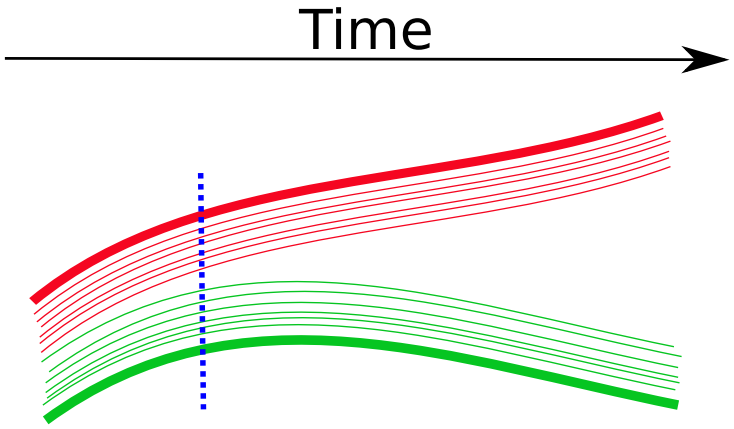

Figure 3.11: **Splitting a fiber**. *Red* and *Green* colors indicate separate fibers. During the fiber build process, for a certain time interval, these two fibers were considered a single fiber. Thanks to our motion coherency criteria, we detect a motion in-homogeneity at a certain time (indicated by dashed *blue* lines). Owing to this motion in-homogeneity detection, we split (spatially and temporally) the fiber into two, and continue the build process for each of these fibers separately.

3.4 Full video coverage

This section explains how we extend the fibers found previously to the rest of the video. For this, we first find zones for possible fiber extensions, and then rely on trajectory coherency to choose among the extension possibilities.

Figure 3.12: **Top Row**: Displays five images of a sequence from [Brox 2010]. **Bottom Row**: Left image displays the same video as a 3D volume as in Fig. 3.1, with an additional bottom right corner showing the 3D position of the xt and yt slices. Right image displays all fibers found, in different colors. Notice that the fibers are well stopped near occlusion vicinity, despite high similarity in color of the tractor and the lady.

3.4.1 Geodesics between the sparse fibers and rest of the video

The reliable fibers found so far do not cover all pixels of the video (*cf.* Figure 3.12). We would like to extend them to the full video by associating to each pixel one of the closest fibers in term of color and motion similarity. This calls for the need of computing geodesics between these fibers and the rest of the video. Efficient algorithms for computing geodesics in quasi-linear complexity w.r.t. the number of nodes in a graph are Dijkstra's shortest path computation [Dijkstra 1959] and Fast Marching Method [Tsitsiklis 1995]. The Fast Marching Method is a continuous variant of Dijkstra's algorithm. As opposed to the Dijkstra's algorithm where a path is restricted to follow the graph edges, the Fast Marching Method allows a path to pass continuously on surfaces (*cf.* Appendix A).

Owing to the simplicity of implementation, we consider the Dijkstra's algorithm on the graph of all pixels of the video, with multiple sources (the fibers). In the *multiple source shortest path* computation parlance, fibers constitute the multiple sources and the rest of the video act as destination.

The local cost considered between adjacent pixels p and q is :

$$\exp\left(\frac{-\alpha}{\|I(p)-I(q)\|_2}\right) + \lambda_{p,q}\exp\left(\frac{-\beta}{\|\mathbf{m}(p)-\mathbf{m}(q)\|_2}\right)$$

where I and $\mathbf{m}$ denote the local color and optical flow, and where $\alpha, \beta \in \mathbb{R}^+$ are constants set to the desirable standard deviation of color and motion allowed for mesh segmentation (*c.f.* section 3.3.2). The cornerness λ (defined in the equation (3.3)) expresses the local trust on the optical flow. As the flow is reliable only at corner locations, it is intuitive to modulate the motion component of the cost function by the cornerness $\lambda_{p,q}$. $\lambda_{p,q}$ is the

average of cornerness measures λ (equation (3.3)) at pixels p and q.

The initial distance of the sources is set to be the opposite of their reliability, $-\lambda$, in order to facilitate the extension of reliable fibers. During the shortest path computations, we keep track of the original sources so that we know to which fiber each pixel is the closest, and even to which trajectory of that fiber. We assign to each pixel p a trajectory T_p^F by copying the one of the closest pixel of the closest fiber (F). Thus we obtain a coverage of the video with possible extension zones for each fiber, with associated trajectories. We empirically set the values for α and β to 30 and 0.01 respectively. The schema of this process is outlined in the Algorithm 1 below:

Algorithm 1: Dijkstra with source ranking by reliabiliy

Data: Graph $G = (X, U)$, Sources S, Distances D, Labels L
Result: Associates Labels L to Sources S, Updates D
begin
  for $x \in X$ **do**
    if $x \in S$ **then**
      $D[x] = \underline{rank(s)},\ s \in S\ ;\ L[x] = s$
      Push x into Queue : Q
    else
      $D[x] = \inf\ ; L[x] = undefined$
  while $Q \neq \emptyset$ **do**
    u = lowest distance element from Q
    $\{Q\} = \{Q\} - \{u\}$, $N(u)$: Neighbors of u
    for $v \in N(u)$ **do**
      $d1 = d[u] + dist[u, v]$
      if $d1 < d[v]$ **then**
        $D[v] = d1\ ; L[v] = L[u]$
        Push v into Q
end

3.4.2 Enforcing trajectory coherency

The extension in the previous section can be susceptible to leaks due to inaccurate *local* assessment of color and motion. A segmented region is said to leak if it crosses boundaries in the original image which it should respect. This notion of boundary depends upon the segmentation objective. In case of an object based segmentation, boundaries correspond to object-boundaries.

Hence it is imperative to assess the quality of the proposed extensions and trajectories, and to ensure their spatio-temporal coherency.

To facilitate a fiber association based on trajectory, we propagate the time trajectory of the vertex as a label during geodesic computation (*cf.* Algorithm 1) in the previous step. Hence for each pixel p we have a fiber label and an associated trajectory T_p^F which

corresponds to the trajectory of closest mesh vertex of fiber F. The subsequent subsections will elaborate on assessing and proposing *long term temporally coherent* fiber extensions.

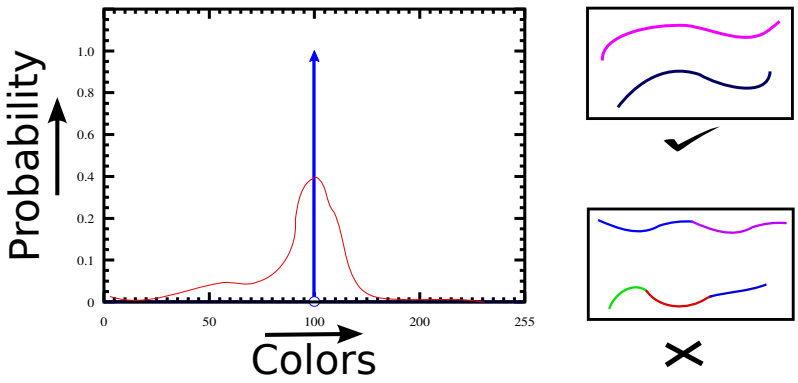

Figure 3.13: Homogeneous and heterogeneous trajectories. The color histogram of a pixel trajectory should be as close to a Dirac peak as possible. In practice, due to some lighting variations or an imperfect camera sensor, the color histogram will not be a perfect Dirac peak. The vicinity of a histogram to a Dirac peak is quantified using the Earth Movers Distance (*cf.* expression 3.4).

3.4.2.1 Expressing the quality of fiber and pixel association

Now the subsequent aim is to design a cost function to express the distance between a fiber and its associated pixels. A good trajectory T_p^F should be color homogeneous (*cf.* Figure 3.13), and its dominant color should be close to the typical color of the fiber vertex trajectory. Sample variance can be used to compute the coherency of a color trajectory. However it is susceptible to outliers and slight lighting variations which is ever-present in any camera recordings. Thus we consider the Earth Movers Distance (EMD, [Rubner 1999]) to quantify the color homogeneity of trajectory T_p^F and estimate its typical (dominant) color.

Let us denote by H_p the color histogram of T_p^F. EMD represents the least amount of work that is needed to rearrange the masses in one distribution to obtain the other distribution. The infinimum of the Earth Mover Distance over all possible color Dirac peaks:

$$\mathrm{coh(p)} := \min_i \; EMD(H_p\,,\,\delta_i) \tag{3.4}$$

as well as its $argmin$ (dominant color), denoted by $col(p)$, can be computed in linear time w.r.t. the number of histogram bins (instead of quadratic complexity naively); by iteratively pushing the content of the lightest extreme bin of H_p towards the rest of the distribution. Algorithm 2 outlines this computation of the dominant color of a trajectory and the distance of a trajectory color histogram from a Dirac peak.

3.4.2.2 Extending fibers by long-term temporal assessment

Assessing every trajectory for each pixel of V, for all possible fiber associations would be computationally intensive as each pixel through which a given trajectory of length D_T

Algorithm 2: EMD Based Trajectory Coherency Assessment

Data: Normalized Histogram H
Result: Compute i (dominant histogram bin in equation (3.4)), homogeneity
begin
  N = number of bins
  Initialize iterators for traversing H
  $itrLeft = 0;\ itrRight = N - 1; costLeft = H[0];\ costRight = H[N - 1];$
  $totalCost = 0; x\ = 0$
  while $x \neq N - 1$ **do**
    if $costLeft < costRight$ **then**
      $totalCost\ += costLeft$
      $itrLeft++$
      $costLeft\ += H[itrLeft]$
      else
        $totalCost\ += costRight$
        $itrRight--$
        $costRight\ += H[itrRight]$
    $x++$
  $i = itrLeft$
  homogeneity $= totalCost$
end

passes would be considered D_T times (since each pixel of this trajectory would ask for building and checking this trajectory).

Thus for an efficient assessment, for each fiber, we consider a representative reference frame, chosen in the middle of its time span. We project the 2D+T possible extension zone of this fiber on this 2D reference frame by following the previously assigned trajectories, as shown in the Figure 3.14. A pixel in a reference frame contains the number of points that are projected onto this pixel, and the region where a pixel has number of points greater the zero is termed *fiber footprint*. Note that this region is one connected component as shown in Figure 3.14. In practice, we smooth each fiber footprint so as to increase its spatial extent. This smoothing of reference frames leads to overlapping fiber influence zones which in turn increases the number of fiber choices available to the pixels.

In order to ensure not only trajectory quality but also spatial and color coherency within one fiber, we re-compute geodesic distances in the reference frame starting from the original fiber, where the cost of moving from a pixel p to an adjacent one q is:

$$\exp\Big(\frac{-\alpha_1}{\mathrm{coh(q)}}\Big) + \exp\Big(\frac{-\alpha_2}{||\mathrm{col(p)} - \mathrm{col(q)}||_2}\Big) \tag{3.5}$$

where α_1 and $\alpha_2 \in \mathbb{R}^+$ are constants (set empirically to 10 and 30 respectively).

We thus obtain a geodesic distance map G_F in each reference frame F, where each point on the fiber footprint has a cost of association to fiber F. Please refer to the Figures 3.15 and 3.16 for fiber footprint distance map samples.

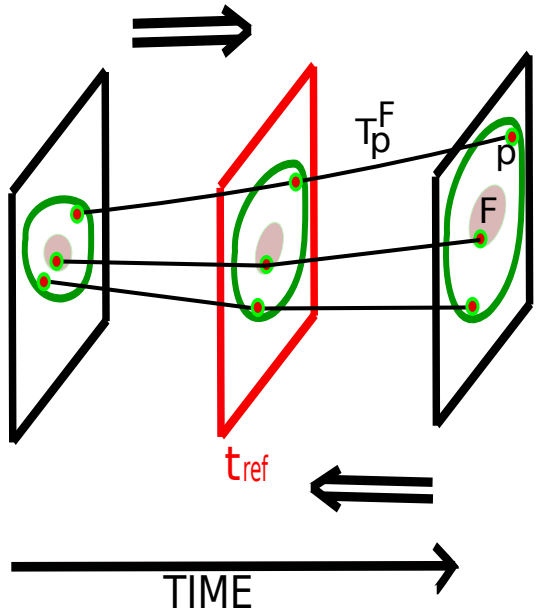

Figure 3.14: The possible extension zone (in green) of a fiber (in brown) is projected on a reference frame (in red), following the trajectories T_p^F (in blue) associated coherently with the fiber F.

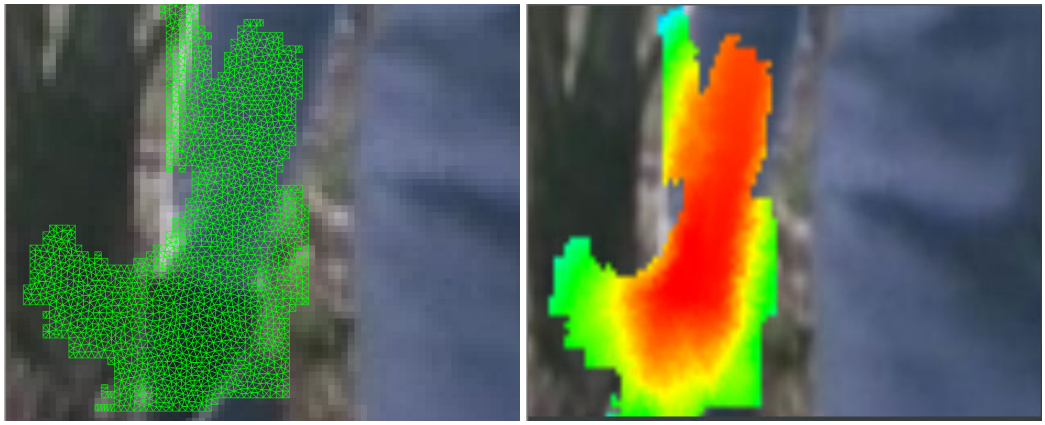

Figure 3.15: **Left** image displays a meshed region of a fiber footprint. **Right** shows the distance map w.r.t. fiber. The red regions correspond to low distances, while green and blue correspond to comparatively higher distances respectively. Note here that the leaks are at higher distance from the sources in Red color.

The cost, for any pixel p of the video, of choosing a fiber f, is then defined as the correponding geodesic distance $G_F(\mathrm{proj}_F(p))$, i.e. as the value of this map G_F at the projection $\mathrm{proj}_f(p)$ of the pixel on the reference frame of that fiber (according to its associated trajectory T_p^F). **Among possible fiber extensions, each pixel then chooses the one with the lowest cost**.

Figure 3.16: Sample distance maps for fiber footprints. The Red regions correspond to low distances, while green and blue correspond to comparatively higher distances respectively. Notice that the leaks are at higher distance from the sources in Red color.

Sample final outputs

Figure 3.17 displays extended fibers. For a simple illustrative viewing, we fuse fibers which have similar motion. The next section elaborates on building a hierarchical video representation by fusing similarly moving fibers.

3.5 Hierarchical representation

At this step we have a very fine representation of the video in term of fibers. This fine representation can be used by algorithms requiring reliable dense long term optical flow *e.g.* action recognition or video compression. Often in many computer vision applications a coarser representation of video is required, *e.g.* for background subtraction or activity recognition. This calls for the need of criteria to merge fibers. Recent works on object segmentation in videos from sparse set of input trajectories [Brox 2010, Fragkiadaki 2011] use graph spectral clustering to obtain a fixed number of labels relating to number of objects in the scene. As opposed to these algorithms we have a trajectory associated to each pixel of the video, and we describe below a simple procedure to obtain a hierarchical representation of a full video in terms of fibers.

In order to obtain a hierarchical representation, we merge fibers based on their speed

Figure 3.17: Displays extended fibers merged at a higher hierarchy. For a simple illustrative viewing, we fuse fibers which have similar motion. The next section 3.5 will elaborate on merging fibers. Darker regions refer to a low reliability of segmentation.

similarity. A similar criterion is used by [Ravichandran 2012] to express similarity of two trajectories. Alternatively a different criterion would be used, *e.g.* as in [Brox 2010] incorporating both spatial and speed distances between the trajectories. With the criterion in expression (3.6), one can expect to hierarchically merge two fibers without considering their spatial positions.

We compute a barycentric trajectory for each fiber. A barycentric trajectory has the same temporal span as that of the fiber and contains a single point at each time frame. A point of a barycentric trajectory at a particular time frame is obtained by computing the centroid of the fiber slice at that time frame.

Two barycentric trajectories are compared by using the expression (3.6), where $O[F_i, F_j]$ defines the overlap time span for two fibers F_i and F_j, and $\mathbf{m}^t_{G,i}$ represents the barycentric motion of fiber i at time t:

$$d(F_i, F_j) \propto \frac{1}{O[F_i, F_j]} \sum_{t \,\in\, O[F_i,F_j]} \left\| \mathbf{m}^t_{G,i} - \mathbf{m}^t_{G,j} \right\|^2 . \tag{3.6}$$

We consider the graph of all fibers, initially fully disconnected, where each node correspond to a fiber. The fibers are represented by their barycenter trajectories, and we empirically set an initial threshold, τ, for merging. At each hierarchy level, we connect the nodes of the graph for which the distance in the expression (3.6) is less than an associated threshold τ. The hierarchy level is changed by multiplying τ with a constant scale factor s (not depending on experiments). This approach builds a hierarchy tree, ensuring that the finer motion details are preserved at the lowest hierarchy while a much coarser motion segmentation is obtained at a higher hierarchy.

3.6 Computational complexity of the fiber extension stage

We denote by V the number of pixels in a video volume, by Ω the number of pixels in a 2D image frame, and by D the depth of the video volume (*i.e.* its duration), thus $V = \Omega \times D$. Ω_F for a particular fiber stands for the number of pixels in a fiber footprint, *i.e.* the area of the influence zone of the fiber F in its reference frame. Similarly, V_F stands for the volume occupied by the 3D influence zone of the fiber F, *i.e.* the total number of pixels in the 2D influence zones of the fiber F for all frames. If we also denote by D_F the duration of the fiber F, then $V_F \approx \Omega_F \times D_F$.

The fiber extension stage is a two step process: Obtaining the fiber influence zones and Extending a fiber in a footprint.

- Obtaining fiber influence zone : $O(V \log \Omega)$. Indeed, the Dijkstra algorithm is run on each frame independently, each time on the image grid of size Ω, and only once for each frame (for all fibers together). Dijkstra on one frame costs $O(\Omega \log \Omega)$, and hence the complexity of this stage is $O(V \log \Omega)$.

- Extending fibers in all fiber footprints, associating to each pixel a trajectory and computing its homogeneity : $O(V \log \Omega)$. Indeed, for each fiber F, a Dijkstra is run on its footprint, of size Ω_F, for which each step costs D_F (fiber temporal length). The complexity is thus $\sum_F (\Omega_F \log \Omega_F \times D_F) = \sum_F (V_F \log \Omega_F) \leqslant (\sum_F V_F) \log \Omega \leqslant \alpha V \log \Omega$, where α is a constant standing for the maximum number of fibers whose influence zone goes through one same pixel.

Hence the total time complexity of the fiber extension stage is **quasilinear**: $O(V \log \Omega)$. In particular, for a given frame size, **the computation time increases only linearly with the duration of the video**.

Note that if instead we had performed Dijkstra on the 3D video grid, to associate each pixel with its closest fiber, the complexity would have been $O(V \log V)$, which grows more than linearly with the video duration D. A factor $\log D$ is not negligible, as for $D = 100$ frames, this factor is already more than 6, slowing down significantly the process. Moreover, estimating trajectories of each pixel, and checking their homogeneity as in our approach, would increase this complexity by a factor D, reaching $O(DV \log V)$, which would be unaffordable in practice.

3.7 Summary

In this chapter we have described our first contribution of the thesis, *i.e.* a novel approach for representing and segmenting a video volume in terms of *Fibers*. A segmentation algorithm provides labeling for all pixels in the video, and a representation algorithm provides a method of accessing segmented components. This fiber based video segmentation not only provides a dense segmentation of a video, but also associates a coherent trajectory to every pixel of the video.

In order to cater to different applications requiring varied level of motion details, we produce a hierarchical representation of a video by merging fibers by comparing their

speeds. The motion of objects and their parts is a definitive component for analyzing a video, and is captured at different resolution by the fibers.

In the next chapter we delve into the implementation details, as well as comparative qualitative and quantitative experiments. We also demonstrate the usefulness of a fiber-based video representation by considering a simple video inpainting task.

CHAPTER 4

Video Segmentation: Implementation and Experiments

Contents

4.1 Implementation 61
4.2 Experiments 62
4.2.1 Datasets 62
4.2.2 Qualitative results 64
4.3 Computation times 70
4.4 Quantitative evaluation w.r.t. point tracks 70
4.5 Optical flow based evaluation 72
4.6 Video Inpainting 76
4.7 Conclusion 78

4.1 Implementation

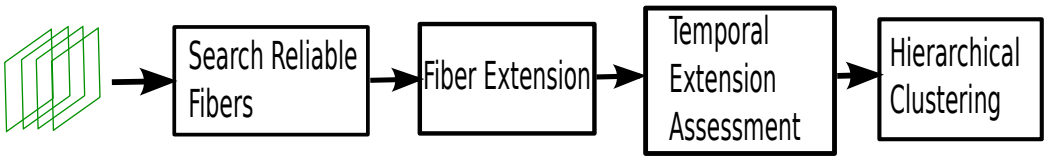

Figure 4.1: Workflow for fiber based video segmentation.

Workflow for our proposed fiber based video segmentation is shown in the Figure 4.1. For a given input video and dense optical flow for all frames, we first build fibers at corner or corner-like regions of this video volume. Subsequently we extend these initial set of fibers to all parts of the video. We incorporate a motion based hierarchy which makes our representation interesting for various computer vision domains such as action recognition or video compression.

The complete code is built in C++. Open source libraries used for various purposes are listed below:

- CIMG (Cool-Image; by INRIA, CNRS): Offers several image processing routines, and most of these routines can be applied to both two and three dimensional images.

The three dimensional visualization used for most of the qualitative results in this thesis is obtained by this library.

- CGAL (Computational Geometry Algorithms Library by INRIA): This library include routines for various algorithms in computational geometry. All of the meshing requirements in this thesis are handled by CGAL.

- OpenCV (Open Computer Vision): OpenCV is the most used open source library in computer vision. We use several components from this library *e.g.* histogram matching, image matching, descriptor matching.

- Vigra (by Heidelberg University): This is another open source library for computer vision and image processing, and it heavily utilizes the underlying concepts of Standard Template Library of C++. Notable algorithms include shortest path computation in N-Dimensional images, Random forest, Anisotropic Diffusion.

The parameters are set empirically and the outputs in this thesis are obtained with the same unique set of parameters.

4.2 Experiments

Dense groundtruth annotation of video datasets is rarely available as it is an enormous task. Since we analyze long term motion and color coherency, we considered the datasets containing long sequences of around 100 frames.

4.2.1 Datasets

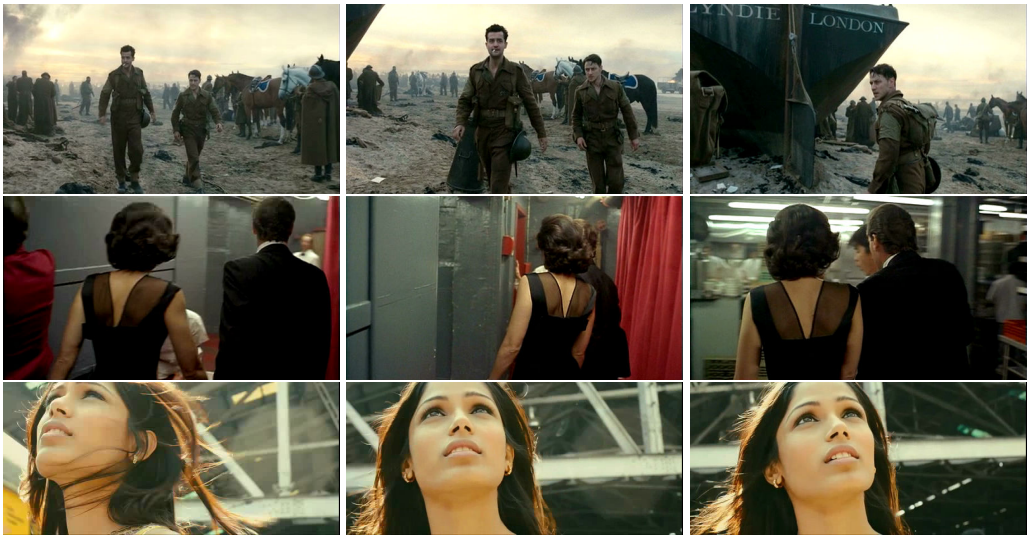

Figure 4.2: Sample frames from the dataset provided by [Grundmann 2010]. This dataset does not provide groundtruth annotations.

The dataset proposed by [Grundmann 2010] consists of videos from *Hollywood*, and has a few very long sequences comprising more than 1000 frames. Sample video frames from this dataset is shown in Figure 4.2. Its downside is that it is highly processed with artificial effects in some frames, and the scene changes are quite rapid compared to natural motion, with frames skipped irregularly, making it impractical for long term coherency check. Moreover this dataset and the one proposed by [Sand 2008] do not have any groundtruth annotations.

One of the datasets used by [Xu 2012] (`xiph.org`) consists of groundtruth annotations. However this dataset does not reflect the current day video usage, as the frame resolution is mere 240x160 pixels. Moreover most video pixels remain static from the first frame to the last frame (*cf.* Figure 4.3).

Figure 4.3: Sample frames from `xiph.org` dataset. Each row indicates three regularly spaced frames from a separate video. This dataset does not represent current day video usage as the frame resolution is mere 240x160, along with a poor frame rate. Importantly in many videos, there is no motion and the first and the last frames are identical (*cf.* last row).

A very popular video dataset is by [Brox 2010] and sample video frames from this dataset is shown in Figure 4.4. This dataset is popularly referred to as *moseg* or motion segmentation dataset. The authors provide groundtruth annotation for a few frames in each sequence. This also comprises an evaluation software which can take as input a set of trajectories and outputs object based segmentation assessment, *e.g.* density of point tracks in a video, object oversegmentation.

Figure 4.4: Sample video frames from the motion segmentation dataset (*moseg*) provided by [Brox 2010]. This dataset provides pixel wise groundtruth annotation for a few frames in each sequence. We consider many videos from this dataset and also use their evaluation metrics (and software) to provide quantitative results.

4.2.2 Qualitative results

We consider the videos from [Brox 2010, Sand 2008]. We present our video segmentation, with spatio-temporal slices, to *display* the label coherency in time, and more classically with image frames. **Darker regions in output images show higher cost of association of the corresponding pixel to the fibers, *i.e.* lower reliability of its association to fibers**.

Fibers at various motion hierarchy levels

Figures 4.5, 4.6, 4.7 and 4.8 show outputs at various motion hierarchy levels.

- Figure 4.5 shows different hierarchy levels of a fiber based representation (both extended and unextended fibers) for a video from *moseg* dataset. Notice that as we move upwards in the hierarchy, we obtain a coarser level of motion details in the segmentation.

- Figure 4.6 displays output for another video from the *moseg* dataset. The colored arrows in the bottom slice of the input image (second row in Figure 4.6), points at

Figure 4.5: Example from the *marple3* sequence till the 50th frame. **Second-Third Rows**: Un-extended fibers at different hierarchy of merging. **Third-Sixth Rows**: Increasing hierarchy in merging of extended fibers.

the parts of the tractor and the lady. The lady occludes the tractor for some time span and the tractor re-appears again. Notice that a local assessment of color and motion will not be able to separate two objects in a dense segmentation, due to their color similarity. Following are the key observations:

- Sparse un-extended fibers (*cf.* second row in Figure 4.6) do not cross the boundaries of the tractor and the lady. Owing to the *spatial statistics* encompassed by the fibers, we computed criteria such as motion variation which have helped in computation of reliable long range fibers.
- *Long term* temporal assessment (*cf.* Figure 4.1) while building fibers for the full video, helps in obtaining a dense segmentation wherein the fibers from the lady do not *leak* to the tractor, and vice versa.

- Figure 4.7 shows another result on a video from [Brox 2010]. We show a particular hierarchy level from our hierarchical representation. This hierarchy level is of particular interest as different motion clusters of the moving person are separated clearly. Parts of the foreground and background are undistinguishable during the first frames of the video (same color). Yet, the two objects follow later significantly different

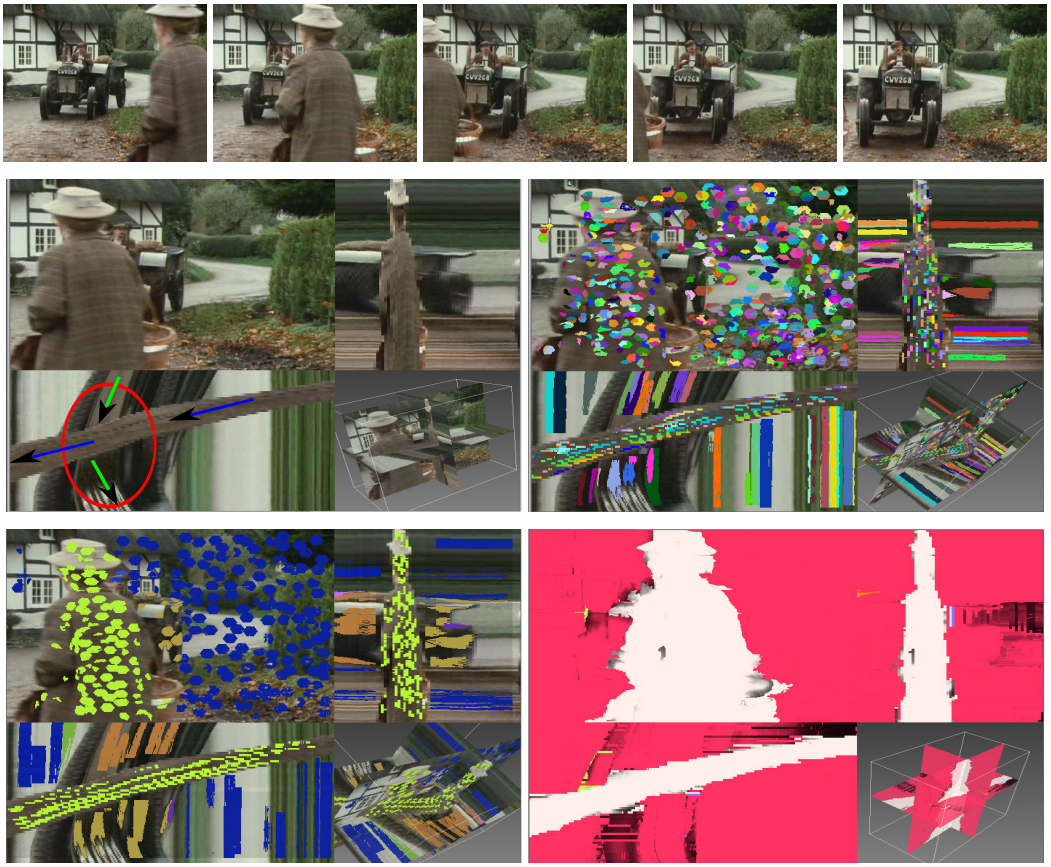

Figure 4.6: **First Row**: Displays five images of a sequence from [Brox 2010]. **Second Row**: Left image displays the same video as a 3D volume as in Fig. 3.1, with an additional bottom right corner showing the 3D position of the xt and yt slices. Colored arrows on the bottom slice indicate the tractor (*green arrows*) and the lady (*blue arrows*). Right image displays all fibers found, in different colors. **Third Row**: Left image displays a high level of their hierarchical clustering. Right Image displays the highest level of the hierarchical clustering, with fiber extension. This result compares favorably to the state-of-the-art of video segmentation in Figure 4.10.

trajectories, which enables us, when propagating this information back in time, to separate them as different fibers in all frames (*cf.* section 3.3.2).

- Figure 4.8 shows an output from a video of [Sand 2008] dataset. Despite fast moving cars we obtain a good segmentation of the cars and the background.

- We also consider a more practical scenario, such as data from subway (metro) stations. A sample video data is shown in Figure 4.9. This data is obtained from the **European Project Vanaheim**. We perform well on computing fibers on the moving objects. Furthermore, the outputs at a lower hierarchy levels, cluster the background and various motion groups appropriately. However at a high hierarchy level (last row in Figure 4.9), some parts of legs of persons get clustered with the background.

Figure 4.7: **First Row**: Sample input image frames (12, 24, 46, 53, 66) from marple13 sequence by [Brox 2010]. **Bottom Row**: After 5 steps of hierarchical merging. Parts of the foreground and background are undistinguishable during the first frames (*until the 12th frame,* i.e. *the leftmost image in the above figure*) of the video (same color). Yet, the two objects follow later significantly different trajectories, which enables us, when propagating this information back in time, to separate them as different fibers in all frames (*cf.* section 3.3.2).

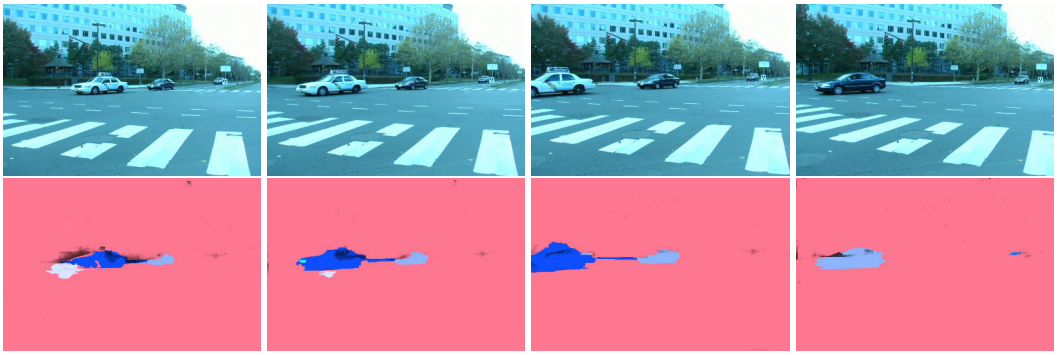

Figure 4.8: Output from [Sand 2008] dataset (frame numbers 1, 10, 20, 40). Fast moving cars are appropriately segmented at a particular hierarchy.

This is due to the hierarchical merging similarity criterion for fibers, as this criterion is purely based on speed of the fibers. Based on speed, points near the ground level (*i.e.* on the legs) are closer to the background, than the other motion clusters of moving objects. To this end, the similarity criterion of fiber merging can be changed depending upon applications, *e.g.* object discovery, action recognition. For example, it might be of interest to modulate the cost function by spatio-temporal distance between fibers, so as to cluster those fibers together who share similar speeds and are close-by in space during their time spans ([Brox 2010]).

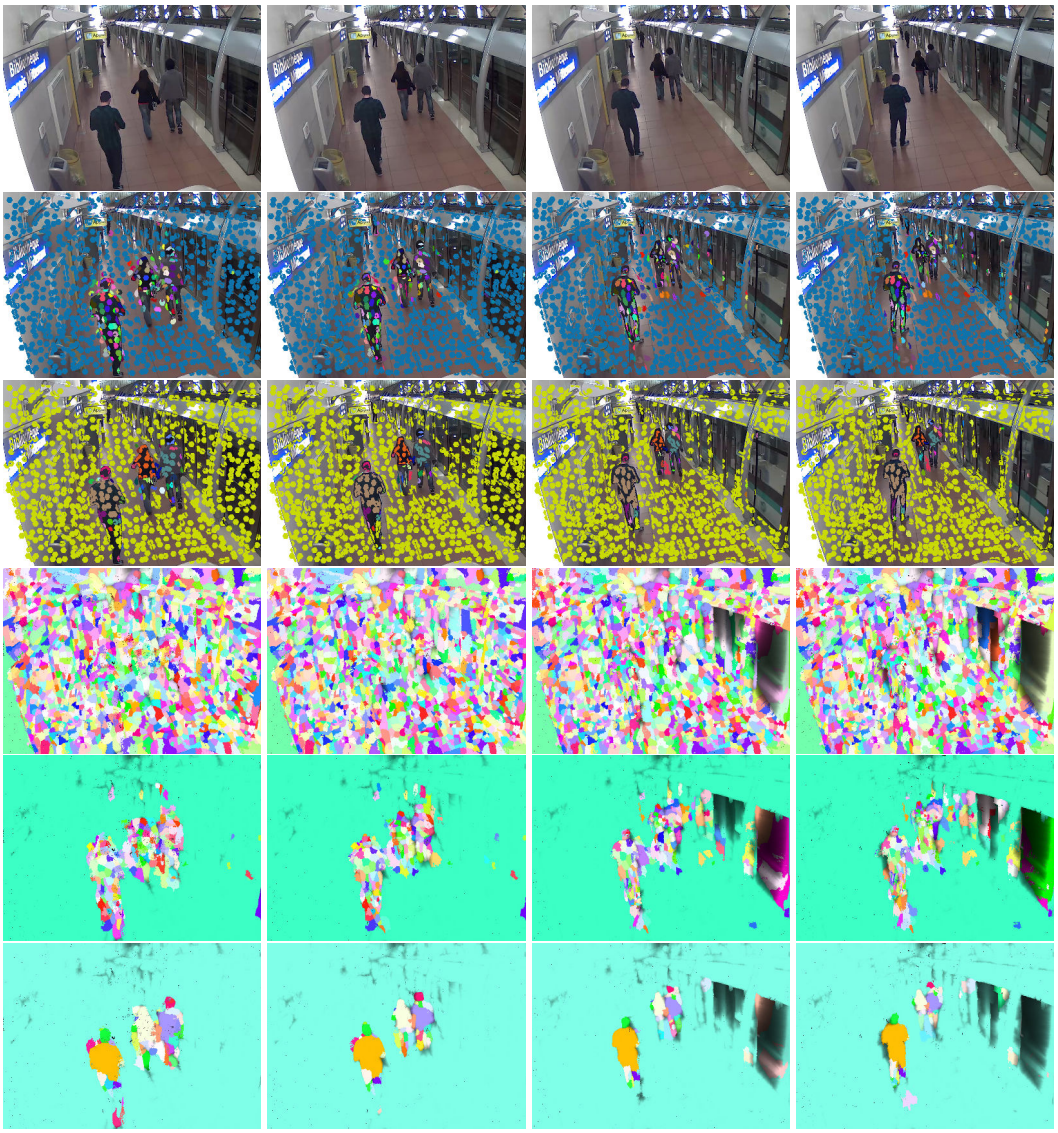

Figure 4.9: **First Row**: Sample input image frames at spacing of 40 frames till 110th frame. **Second Row**: Fiber merging at a lower hierarchy. Notice that the background belongs to one cluster now. **Third Row**: Fiber merging at higher hierarchy. Notice that different parts of the moving persons (*e.g.* head, torso, elbow) belong to different clusters. **Fourth Row**: Dense fiber flow after trajectory based association (3.4.2). **Fifth Row** : Extended Merged Fibers at a lower hierarchy. In this and following images, *zones segmented with low reliability are darker in color*. **Sixth Row**: Extended merged fibers at a higher hierarchy level. Some part of leg now belongs to the background. This due to the hierarchical merging cost function we employ. The un-extended fibers, however, have well respected boundaries as can be seen in other images.

Importance of long term motion for segmentation

Figure 4.10 shows the result, on the same video as in Figure 4.6, obtained by the state-of-the-art in video segmentation [Grundmann 2010]. This approach by [Grundmann 2010] uses optical flow and color information, but not trajectories. The inaccuracy of the segmentation (by [Grundmann 2010]) can be seen by the leaks from the lady to the tractor in Figure 4.10. Moreover this approach does not provide any long term correspondences. This example shows that long term motion is indeed vital for segmentation, and that local optical flow does not help sufficiently, hence the interest of our approach. The main foreground objects, the lady and the tractor, are already merged at a relatively low hierarchical level (third). It may appear that increasing the number of hierarchy levels could solve this leak. However the two objects in the *red* circled zone are of similar color, and a local color and motion assessment is incapable of indicating the presence of two different objects.

The algorithm by [Grundmann 2010] obtains better segmentation by employing both optical flow and color information, than using only color information. This is noted in their paper [Grundmann 2010]. However recent comparative evaluations by [Galasso 2012], [Xu 2012] against [Grundmann 2010] ignore the usage of motion information for [Grundmann 2010]'s implementation.

Figure 4.10: Output from [Grundmann 2010]. **Left**: Input image and its spatio-temporal slices. Colored arrows on the bottom slice indicate the tractor (*green arrows*) and the lady (*blue arrows*). **Right**: Result obtained by the state-of-the-art segmentation of videos [Grundmann 2010], using also optical flow, for comparison. The main foreground object and the background are already merged at a relatively low hierarchical level (third). It may appear that increasing the number of hierarchy levels could solve this leak. However the two objects in the circled zone is of similar color and a local color & motion assessment is incapable of indicating the presence of two different objects. Our result for this video can be seen in Figure 4.6.

Summary: All of the above results demonstrate the robust incorporation of long term motion detail by fibers. Fibers not only provide segmentation of a video into different motion groups, but also provide a trajectory for all pixels in a video.

4.3 Computation times

Sample computation times for a fiber based video segmentation on the *moseg* dataset are shown in Table 4.1. The results on all videos are obtained with the *same unique set of parameter values*. The total computational times including flow computation for typical videos from [Brox 2010] range from 70∼120 seconds for 1 second of video (20 frames), which is **very fast** compared to usual approaches and can still be easily improved. We use the GPU-based optical flow [Werlberger 2009] on a basic Nvidia graphics card.

The last column in Table 4.1 shows the *expected* times after parallelization of successive chunks in a video stream on a standard 8 processor machine.

Time for	1s of 20 Hz video 350×300 px		
	typical	worst case	optimized (**expected**)
Flow	20 s	20 s	real-time
Fiber build	30 s	60 s	4 s
Extension	20 s	40 s	2.5 s
Total	70 s	120 s	7 s

Table 4.1: Computational time: Typical times observed for one second of a standard video, worst case times (fast background motions), and time expected if using the last GPU card and parallelization of successive chunks in a video stream on a standard 8 processor 2.4 GHz machine.

4.4 Quantitative evaluation w.r.t. point tracks

Quantitative metrics from [Brox 2010] aim at evaluating object extraction algorithms in videos using long term motion analysis, and hence are not suitable (in direct form) for our hierarchical motion based video representation. In our motion based representation, the static regions will get merged to one another in the first level of hierarchy. For example, a static chair in a static camera scene will get merged to the background in the first merging step of our hierarchical representation. Furthermore fusing fibers based only on motion is susceptible to merging regions without any heed to their spatial locations at higher hierarchy levels. Our aim is not to extract object in videos, but to have a more general representation of a video, which not only provides trajectories for all pixels, but also facilitates accessing spatio-temporal neighborhoods using meshes. This neighborhood size varies hierarchically with motion.

Although metrics from [Brox 2010] are not suitable for a direct comparison with object extraction algorithms, these can be utilized to show some meaningful quantitative results. We use 15 videos from this dataset, and compute the evaluation metrics for the first 50 frames. [Brox 2010] comprises the following four metrics for evaluating an object extraction algorithm:

- Density: Corresponds to the percentage of pixels covered by the trajectories.

- Per pixel Error (PPE): This determines the temporal correctness of the trajectories *i.e.* a trajectory should not encompass multiple objects.
- Per Region Error (PRE): This quantifies the average error per region. So discarding a region will penalize 100% error for the corresponding region.
- Oversegmentation Error (OE): Determines the number of clusters that need to be merged to obtain groundtruth annotation.

An observation: Trajectories that do not cross two labeled groundtruth frames bias the above metrics *e.g.* PPE and PRE, *i.e.* it is possible to have low PRE and PPE by using trajectories of one frame span which are present only at annotated (groundtruth) frames. Hence we remove all trajectories which do not pass at least two groundtruth (annotated) frames in their time span. However we could not perform the same on [Brox 2010]'s results.

To obtain meaningful results we need trajectories with labels. A label for a trajectory encodes its clustering identity. For example, if at a particular hierarchy, we have F fibers, then all trajectories will have a label which can be any number from 0 to $N-1$.

Each fiber has its own reference frame and each element of a reference frame has a trajectory associated to it. Fiber trajectories for the complete video can be obtained from fiber reference frames in the following manner:

- Using the trajectory at a particular element of a fiber reference frame, we move forward and backward in time. This forward and backward sweep for a pixel is continued until we reach the end of the video volume, or if we reach a pixel which does not have the same fiber label. This forward and backward movement then defines a trajectory. We repeat this process for all pixels in all reference frames.
- **We ensure that each trajectory crosses at least two groundtruth frames.** This is done to make sure that we do not input short trajectories to reduce per pixel error.

In order to obtain meaningful comparison metrics, we **fix** the oversegmentation to be less than 100, and produce the metrics for the resultant clustering *i.e.* the hierarchy where the number of clusters goes below 100 for the first time.

Note that with the above approach we are not using our reliability measure for tracks. So all trajectories are equally reliable for obtaining these above quantitative metrics. Quantitative results are shown in the table 4.2.

Method	Density	PPE	PRE	OE
Ours	65.0	11.3	32.0	42.6
[Brox 2010]	3.3	7.13	34.7	0.53

Table 4.2: Quantitative results on the *moseg* dataset [Brox 2010]. Above metrics are the averaged over 15 videos (of 50 frames each).

We have the following inferences:

- We manage a considerably low average pixel error and per region error. Hence the flow encompassed by the fibers is reliable. Also PRE is competitive and favourable for us.

- Our density is exorbitantly high despite removing trajectories which do not cross a minimum of two groundtruth frames. Input fiber trajectories are at least 10 frames long in plus of the restriction that they cross at least two groundtruth frames.

- Other recently proposed approaches [Ochs 2011], [Galasso 2012] mention density of labels as 100%. However the temporal correspondences are only available for 3% of the total pixels in the video. Labels are propagated from point tracks to the rest of the video considering superpixels and hence there is no addition of temporal correspondences, apart from those which are already computed as input to these approaches.

- Our merging criterion is solely based on comparison of fiber speed, and hence we incur high error for static regions like chair, which are merged to the background in the very first hierarchy level of ours.

To summarize, this evaluation demonstrates that fibers do not miss regions (low PRE), encompass good flow (low PPE) while maintaining a high density of correspondence coverage. We could not include the popular video segmentation approach by [Grundmann 2010] in above evaluation as there are no long term correspondences provided by this approach.

4.5 Optical flow based evaluation

In a fiber based video representation, we have dense point trajectories for all pixels in a video. Hence it makes sense to evaluate this representation w.r.t. optical flow evaluation datasets. One of the earliest popular optical flow evaluation dataset is the *Middlebury dataset* [Baker 2010]. This dataset contains sequence of 8 frames only, and hence is not suitable for our long term fiber based video representation.

Another very recently proposed, artificial optical flow dataset, is by [Butler 2012], termed *MPI-Sintel dataset*. This dataset is derived from the open source 3D animated short film *Sintel*. Sample consecutive frames from this dataset are shown in Figure 4.11. MPI-Sintel has very large motion in consecutive image frames of sequences, and also the objects get occluded frequently in a space of a couple of frames. Hence this dataset is not suitable for evaluating long term spatio-temporal motion coherency.

We finally use the dataset by [Liu 2008], commonly referred to as the *MIT dataset*. This dataset provides optical flow for all sequences, and the sequences are much longer in time (close to 50 frames for all videos) than the Middlebury dataset. Sample image frames can be seen in Figure 4.12.

Figure 4.11: Sample image frames from the MPI-Sintel optical flow evaluation dataset [Butler 2012]. Each column displays consecutive image frames from one sequence. Large motion and frequent occlusions prohibit the usage of this dataset for our fiber based video representation.

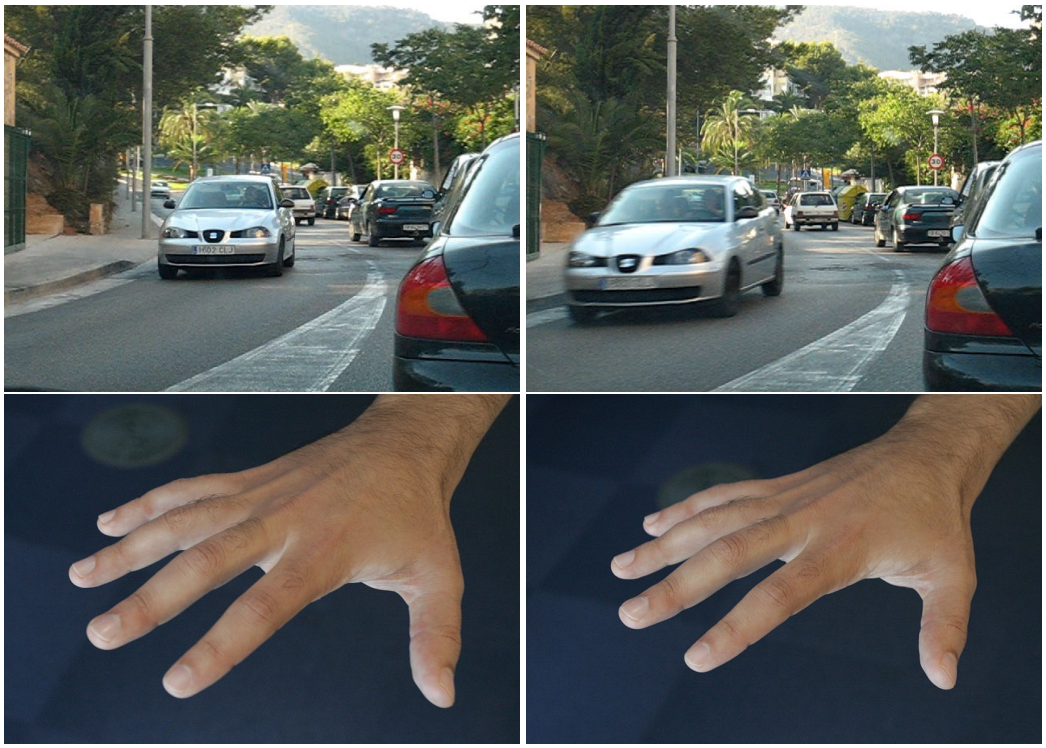

Figure 4.12: Sample frames from the MIT optical flow dataset [Liu 2008]. This dataset includes groundtruth annotations.

Evaluations

For every sequence, we compare the mean square error (MSE) of the flow encompassed by the fiber trajectories w.r.t. the groundtruth optical flow over the whole video volume. Fiber

trajectories are computed in the same manner as in the previous section for evaluation on the *moseg* dataset of [Brox 2010]. Table 4.3 shows quantitative evaluation of fiber-trajectories on this dataset.

We consider the following quantities:

- We compute the MSE of the input optical flow to our fiber based segmentation w.r.t. the groundtruth, over the full video volume (*cf.* Table 4.3).
- We measure the MSE of the optical flow encompassed by the fiber trajectories w.r.t. the groundtruth over the full video volume (*cf.* Table 4.3).

Sequence name	Input flow error	Fiber flow error
Cars	0.55	0.59
Toy	0.98	1.20
Hand	6.05	6.20

Table 4.3: Optical flow evaluation on the MIT dataset [Liu 2008]. Each row shows quantitative optical flow evaluation for a particular video, in terms of **per pixel error**, averaged over the whole sequence.

Observations

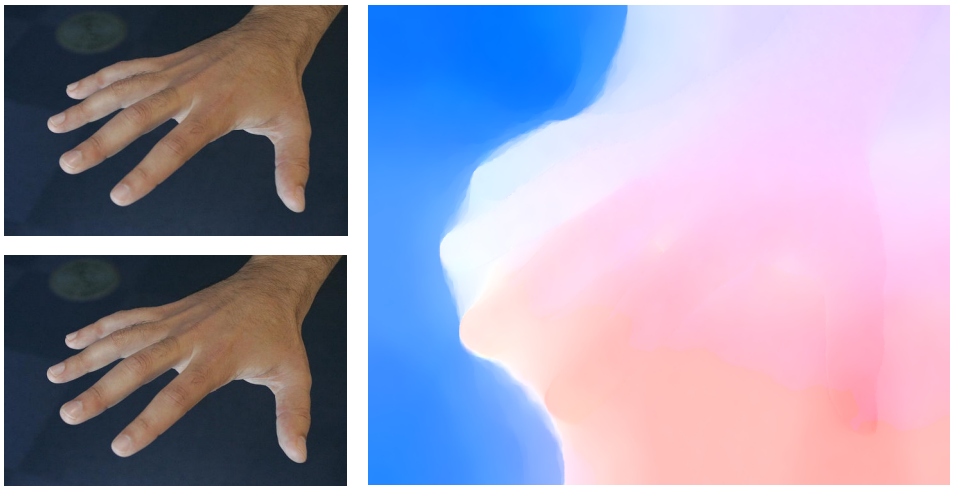

Figure 4.13: Optical flow computed by [Werlberger 2009] for the two consecutive frames on the left. Colors on the *right* image is the standard *Middlebury dataset* encoding of the optical flow. We use this optical flow algorithm as input flow for our fiber based video representation. Notice that the flow is wrongly estimated for areas between the fingers of the hand.

Extending fibers have increased the optical flow error, by a small margin. This is mainly attributed to the following:

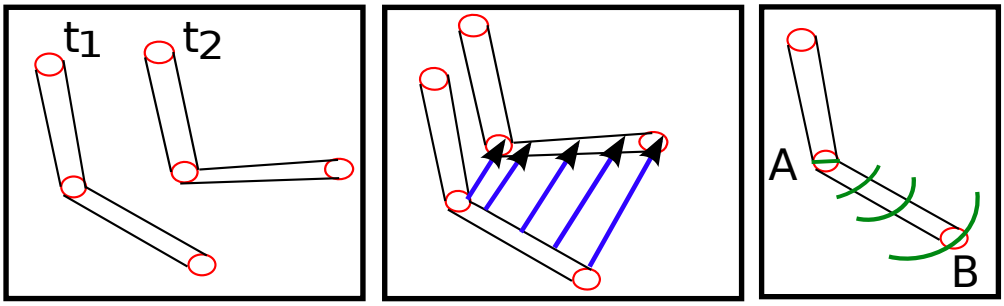

Figure 4.14: Interpolation of trajectories between articulated object parts. **Left**: A rotating object part. t_1 and t_2 mark the time instants and show poses of object. **Middle**: Expected flow is shown by blue arrows. Notice that a linear interpolation is sufficient, while a piecewise-constant interpolation cannot work in the case of rotations. **Right**: Interpolating trajectories from two trajectories (**A**, **B**) located at articulated object parts.

- Long term motion assessment: For the long term temporal assessment in our segmentation workflow (*cf.* Figure 4.1), we *copy* the closest fiber trajectory to a point. This is a good long term motion approximation for a nearby point in a fiber influence zone, and is accurate for translational motion of the object. However this copying of fiber trajectories is not precise at locations of rigid/non-rigid movements (*cf.* Figure 4.14).

  In case of a rotating object, a better long term motion estimate would be obtained by interpolating trajectories between different object parts (articulations). Initial trajectories will be provided by unextended fibers on various object parts (*cf.* Rightmost inset in Figure 4.14 for an illustration). The challenges here lie in knowing which fibers to choose for interpolation at a point/region, and keeping track of points which repetitively appear and disappear because of occlusions.

- Furthermore, this assessment at a point for its association to a fiber, is based on copying a fiber trajectory of a *fixed* temporal span, *i.e.* the copied trajectory has the same start and stop time as that of the fiber. This is again a good approximation in nearby regions in space, but this will lead to an inaccurate long term motion assessment in videos such as the hand sequence where points are appearing and disappearing frequently.

  For points appearing or disappearing in a scene, it will be of interest to consider *splitting* of trajectories into shorter color coherent segments.

This evaluation is based only on the basic two-frame flow measurement, and does not consider other advantages of fibers for long term spatio-temporal statistics. In order to demonstrate practical usage of a fiber based representation, we consider a *video inpainting* task in the subsequent section.

4.6 Video Inpainting

We now show the usefulness of fibers in practical settings such as video editing. The hierarchical representation of fibers allows the selection of moving objects in videos very efficiently. In Figure 4.15 we perform a video inpainting task, for which we need to remove the disturbing girl behind the reporter. For this task the video zones to remove or to keep are selected in only **very few clicks**. This is to be compared with the state of the art [Granados 2012] which requires manual segmentation of all frames.

Figure 4.15: Input frames for the inpainting demo (Frame numbers: Top Row: 0, 22. Bottom Row: 44, 61).

Steps for this inpainting process is shown below:

(S_1) Compute fibers: Fibers are computed for the input video, along with the hierarchy for motion.

(S_2) Select the hierarchy to use: The user can select visually the hierarchy. For the example shown we consider the finest hierarchy (*i.e.* at a hierarchy where no two fibers are fused). Figure 4.17 display this finest hierarchy level. It can be seen that fibers belonging to the girl and the reporter are curved and are not parallel to the time-axis, whereas the background fibers are straight and are parallel to the time-axis.

(S_3) Select the fibers to keep: Both persons in the scene are moving and hence fibers on them are curved and these fibers are not parallel to the time-axis. In order to know which fibers (or pixels) need to be kept, a user clicks on the foreground (reporter) object (shown as green dots in Figure 4.16). Pixels in the original video corresponding to the fibers which are inside (or located at) the contour formed by green dots need to be kept in the final output video.

Figure 4.16: Inpainting task. **Left**: original video (top) and xt slice (bottom) showing trajectories. **Right**: Our result. Clusters of fibers were computed and selected with only 7 mouse clicks to distinguish the disturbing girl from the reporter and background. The girl was removed and the hole was filled by extending the background fibers in time.

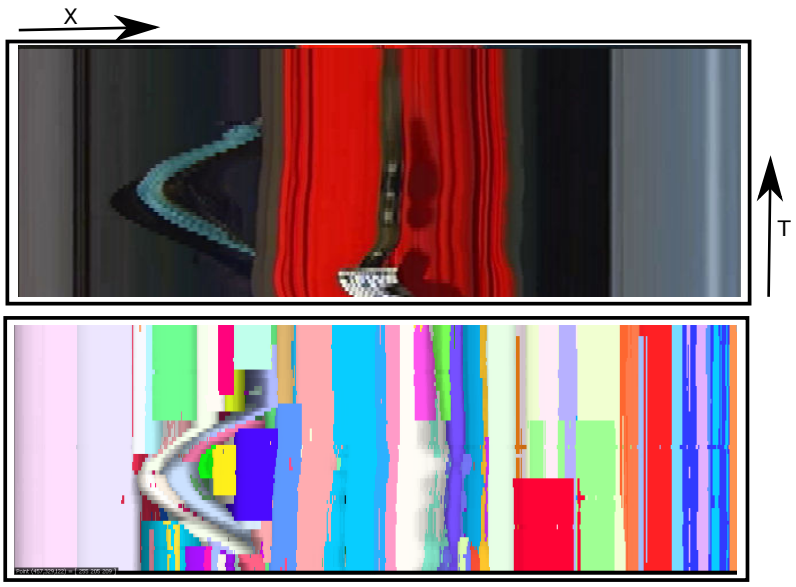

Figure 4.17: xt slices for the input (*top* row), and dense fibers (*bottom* row) at the finest hierarchy. The spatial statistics help in getting good boundary contours of the foreground object (*i.e.* reporter) by only a few user mouse-clicks. The straight fibers parallel to the time axis help in completing the hole formed by removing the girl. Fibers belonging to the girl are identified by finding fibers which are not parallel to the time axis.

Notice that if we had used a **point track based representation**, we would have needed more clicks from the user to precise the boundary of the reporter. Instead, the spatial statistics (mesh) of fibers have helped us in reducing the number of user

inputs significantly. Moreover the temporal coherency of fibers have helped us in marking the object in only one frame.

Also note that a **supervoxel based representation** would be incapable of determining background regions due to the lack of temporal correspondences.

(S_4) Remove the moving fibers outside the user-selected zone: Fibers which are not straight-in-time (or parallel to time-axis) and are not inside the contour formed by *green* dots, are removed. This step now removes the girl, and forms a hole in the video volume.

(S_5) Filling of the hole: Leveraging on the correspondences for all pixels in the video, we extend the pixels belonging to the background (straight) fibers in time (to fill the hole formed by removal of the girl). The final result can be seen in Figure 4.16.

4.7 Conclusion

We presented a novel representation of videos in terms of **fibers**, practical to handle jointly temporal aspects (such as motions and trajectories) and spatial aspects (such as meshes and segmentation into regions). We build these fibers in **quasi-linear complexity** $O(V \log(\Omega))$, which makes them affordable in practice for real applications.

The advantage of our approach over classical segmentations is that **we associate not only a label to each pixel, but also a coherent trajectory**. Thus the segmentation is more robust to noise. Moreover, we provide, in plus of the segmentation, a reliability map of our result.

Meshes of trajectories prove useful in many places. First, they allow to define vertex-dependent quantities, such as motion or depth, while ensuring the continuity of their variation. Furthermore, they provide a dense, organized video coverage, to the contrary of most approaches offering only sparse tracks, short tracks, or frame-by-frame estimations. The range of possible criteria to optimize with our representation is much greater, as it allows us to express statistics, both in time and in space, making estimated quantities more robust to video noise.

Another strength of this framework is the incorporation of **hierarchical clustering, with meshes**. For action recognition applications it is often desired to keep the finest representation in term of fibers (long term dense optical flow) while for domains like background segmentation or foreground estimation (in freely moving cameras), a much coarser representation can be selected. This proved very useful in the video editing task. Fibers are a middle-level entity bridging the gap between low-level pixels and high-level activity recognition: usual practical problems in computer vision, like lighting variations, shadows or occlusions, are difficult to face at the pixel level, and require more semantic information from the scene. Shadows or occlusions will result in several bits of homogeneous fibers, that can easily be merged later at a higher level, based on global trajectory similarity or rules (*e.g.* a darker fiber at the bottom of a foreground object and following it is a shadow).

Part II

Multiple Object Tracking

CHAPTER 5

Multiple Object Tracking: Related Works

Contents

5.1 Maximal cliques and independent sets for tracking 82
5.2 Network flow based tracking models 84
5.3 Stochastic sampling for tracking 85
5.4 Multiple hypothesis tracking and Probabilistic data association techniques 87
5.5 Discrete-Continuous optimization for tracking 88
5.6 CRF based approaches 88
5.7 Probabilistic occupancy maps for tracking 89
5.8 Practical Considerations 90
5.9 Summary 91

This chapter presents an overview of most popular state-of-the-art approaches for multiple object tracking.

Multiple object tracking is the task of finding trajectories of objects in a scene (*cf.* Figure 5.1). It is a key computer vision problem in a number of vision application domains, such as action recognition, video editing, surveillance, and autonomous vehicles. In this thesis, we address the problem of multiple object tracking in videos obtained from a single camera. As the standard validation benchmarks in this field are centered on pedestrian tracking, this chapter focuses on the case where the objects to track are persons. About 30 novel multi-object tracking approaches get published in major computer vision conferences every year. Given this enormity of the literature, it is impossible to have a thorough review of the state-of-the-art. Hence in this chapter we present an overview of the recent and popular approaches in multiple object tracking.

Online and Batch-based tracking

Tracking algorithms can be broadly divided into two categories: online and batch-based. Online methods infer the object state using the current frame and the previous ones; this can be achieved for instance with particle filtering [Breitenstein 2009, Okuma 2004]. On the opposite, batch-based tracking considers the entire temporal sequence at once, or at least a significant part of it (hundreds of frames). Such methods are also referred to as

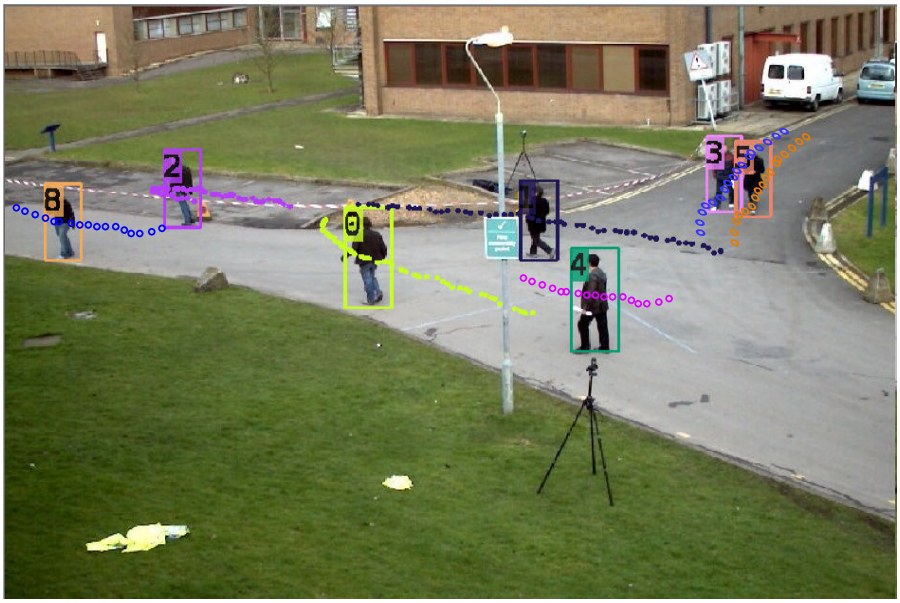

Figure 5.1: Multiple Object (Person) Tracking. The labels for each bounding box is the unique identity of a person. Dotted points show the trajectories of bounding boxes in a few previous and successive frames.

global tracking in the literature. A detailed survey on online tracking methods and their parameter adaptation to various videos can be found in [Chau 2012].

Global tracking should be in principle less prone to ID switches (confusion of several different persons) than online approaches, and provide better occlusion management as they have both present and future frames available to determine the state of objects. This extra information brought by the 2D+T volume has led to the development of many different approaches utilizing all kinds of color and motion cues, as well as many different optimization methods.

Our approach belongs to the class of batch-based tracking, and hence we focus on the related work in the realm of batch-based tracking.

5.1 Maximal cliques and independent sets for tracking

[Zamir 2012] proposes the search for maximal cliques in a graph to solve the tracking data association problem. From the detection responses, the authors build a graph for a given temporal window. The nodes in this graph correspond to the detections and the edges incorporate suitable affinity functions. The graph is fully connected, except that nodes in the same time frame have no edges between them. Person trajectories are then found by greedily computing the best maximum weighted cliques in the graph. A clique of a graph is a subset of nodes in which every pair of node is connected by an edge. A maximal clique is a clique that cannot be extended by including one more adjacent vertex. Given a weighted graph, it is of interest to find maximal weighted cliques; a maximal weighted clique (MWC) is a clique which has the highest sum of edge weights, and it cannot be

extended by including more adjacent vertices.

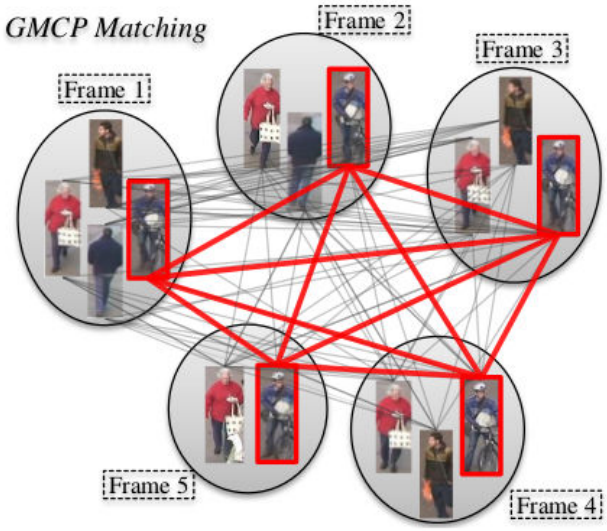

Figure 5.2: A maximal cliques-search based tracker. Gray and colored edges represent input graph and optimized subgraph respectively. **Image source : [Zamir 2012]**.

Each MWC in a graph corresponds to a person trajectory. The intuition for proposing tracking as a search for MWCs is that if the edges in the graph encode good affinity measure, then for a particular output trajectory, we can find a subset of nodes such that the sum of edge weights connecting them will be maximized (*cf.* Figure 5.2). This subset of nodes achieving maximum score will correspond to an output trajectory. The affinity measure between nodes is based on body & body-parts similarity. The authors [Zamir 2012] use a heuristic optimization technique, named Tabu search [Glover 1997] to find MWCs of a graph in an iterative manner. Nodes belonging to a maximal clique are removed from the graph and the Tabu search is re-run to compute next best maximal clique from this reduced graph. Since the affinity measure between nodes can be noisy due to persons of similar appearance and overlapped bounding boxes, the authors use motion models to remove outliers from a RANSAC [Fischler 1981] based approach. For a streaming trajectory computation, the authors divide the video into multiple batches, and person trajectories for each batch are computed independently by finding MWCs. Temporal merging of trajectories is performed by computing MWCs of a new graph. In this new graph, a node corresponds to a person trajectory of a batch, and edges encode similarity of two trajectories. Similarly to the graph for computing trajectories, there is no edge between two trajectories from the same batch.

The task of multi-object tracking can also be seen as finding maximum weighted independent sets in a graph [Brendel 2011]. The independent set problem and the clique problem are complementary: a clique in graph G is an independent set in the complement graph of G and vice-versa. Detection hypotheses are used to build a graph of detection pairs, termed as tracklets. Each detection is characterized by a descriptor and the best matching detections across frames are transitively linked into distinct tracks. A hard con-

straint explicitly imposes trajectory exclusion such that no two tracks share a detection. Two-frames tracklets are then used to build a graph. Node weights encode the similarity of the corresponding matches and edges connect nodes whose corresponding tracklets violate the hard constraints. Given this attributed graph, data association is formulated as the maximum-weight independent set (MWIS) problem. MWIS is the heaviest subset of non-adjacent nodes of an attributed graph. Long term occlusions are addressed by re-iterating the process of a trajectory split and merge so as to increase the total weight of MWIS until convergence.

MWIS is a well-researched combinatorial optimization problem, known to be NP-hard, and hard to approximate. Numerous heuristic approaches exist for this problem. In this work [Brendel 2011], the authors derive a new MWIS algorithm that iteratively refines the solution using first order dynamic. Convergence to a local maximum is guaranteed with complexity of $O(n^2)$, where n is number of nodes of the graph.

5.2 Network flow based tracking models

One of the first network flow based global optimization scheme for multiple object tracking is by [Jiang 2007]. The authors use an Integer linear programming formulation to visual tracking of multiple objects.

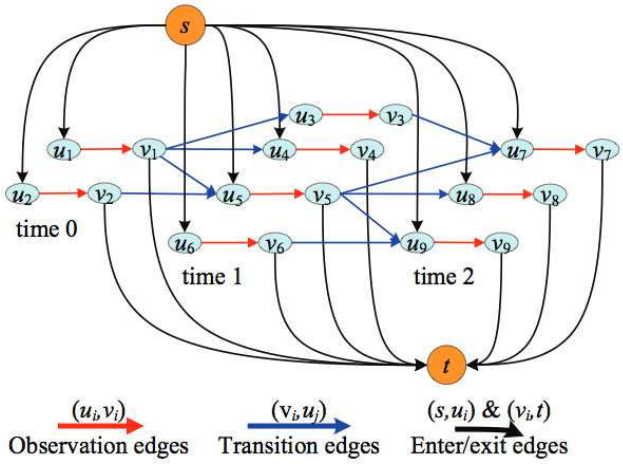

Figure 5.3: Network flow model graph for [Zhang 2008]. Nine detections (including false positives) from three successive frames are shown. Trajectory computation is performed by maximizing the flow from the source s to sink t. Nodes which are not connected (in the above Figure) to either of s, or t are deemed as false positives (post optimization) **Image Source:**[Zhang 2008].

A popular network flow based tracking approach is by [Zhang 2008]. For every detection hypotheses two nodes are created which are connected by an arc. One of the two nodes is connected to the source node and the other is connected to the terminal node. Each transition is connected by an arc as shown in Figure 5.3. Finding optimal trajectories in this model correspond to sending the flow from source s to sink t that minimizes the cost.

Each flow path can be interpreted as an object trajectory and the amount of flow sent from s to t is equal to number of object trajectories. The authors impose a flow conservation constraint which ensures no sharing of edges across multiple paths, and hence no overlapping trajectories. Temporal occlusion is handled by iteratively adding hypothetical nodes in the graph and re-running the optimization process.

Work by [Pirsiavash 2011] also uses a network flow based model similar to [Zhang 2008]. As opposed to [Zhang 2008] which uses push relabel based optimization, the authors of [Pirsiavash 2011] use K-shortest path based optimization. By greedily computing shortest paths, they reach near-linear time complexity in computing the global minimum of successive sub-problems, which however does not guarantee a global optimum on the real global problem. For N video frames, the optimization algorithm of [Zhang 2008] has complexity of $O(N^3 logN)$ to compute K tracks, while the complexity of the optimizer of [Pirsiavash 2011] is $O(KNlogN)$. This reduction in complexity is achieved by exploiting the structure of graph in 5.3. More precisely this network flow graph has edges of unit capacity and is directed acyclic. These graph traits allow the usage of dynamic programming to compute K-shortest paths efficiently.

Network flow models suffer indeed from the disadvantage that possible occlusions or false negatives are not handled from the start, but in a post-processing step by introducing iteratively new nodes in the graph, which asks for re-iterating the optimization process until convergence. Importantly all network flow approaches and [Brendel 2011] do not incorporate acceleration information for computing trajectories. Acceleration knowledge is vital for ensuring smooth velocity in trajectories, and atleast three nodes (*i.e.* detections) located at different time frames are required for computing acceleration. Recently a proof-of-concept work by [Collins 2012] have shown that a network flow model can only deal with pairwise factors and hence cannot incorporate a third (or higher order) factor. The authors in [Butt 2013] extend a typical network flow model by adding higher order connections (*cf.* Figure 5.4). However in doing so they have to resort to a relaxation (discrete to continuous) based optimization schema, as their model can no longer be minimized by efficient network flow approaches such as Graph Cuts or K-shortest paths. For time critical vision applications such as tracking, a discrete to continuous relaxation approach should be avoided, as these do not generally provide a usable solution at all instants during their run times. There is no guarantee that the solution at next time instant is atleast as good as the current one. Hence stopping the optimizer might return a non usable solution.

5.3 Stochastic sampling for tracking

A noteworthy approach is from [Benfold 2011] which uses person detection hypotheses along with head detections to find output object trajectories. The approach is based on the principles of Minimum Description Length (MDL) [Rissanen 1978] by attempting to find the trajectories which allow for the most compact representation of the observed detections.

Apart from associating input detection hypotheses, the spatial position of bounding boxes are also optimized. The authors also use information from KLT point trackers to find probable detection matches in successive frames. The data association stage consists of

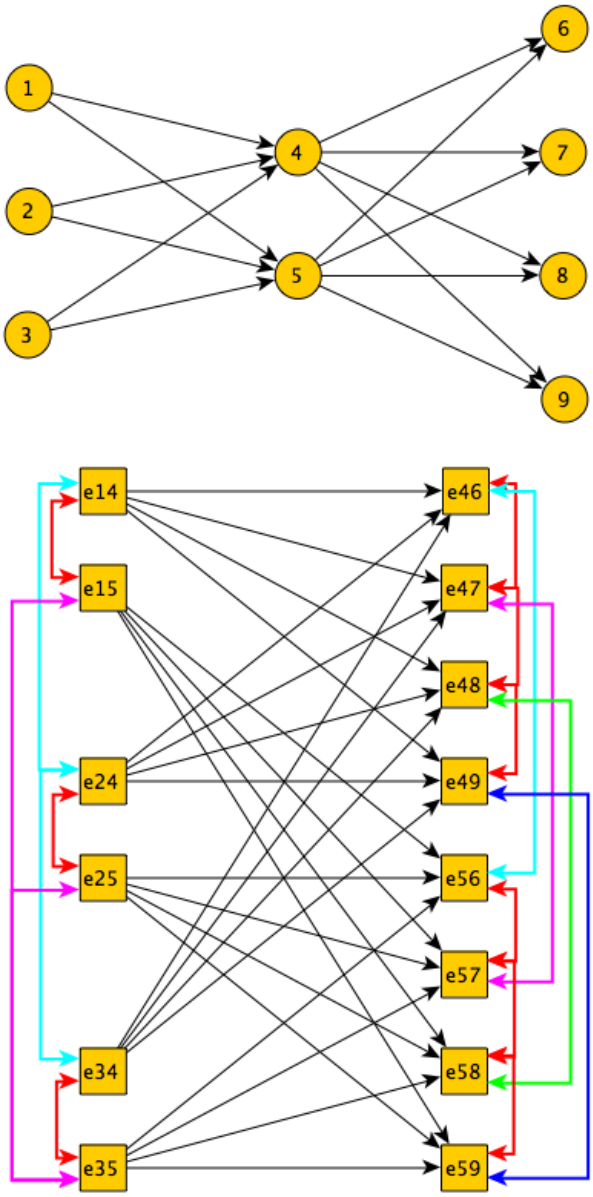

Figure 5.4: Higher order connections in network flow model by [Butt 2013]. **Top**: Graph depicting a three frame sequence with three observations in the first frame, two in the second and four in the third frame. **Bottom**: Graph proposed by [Butt 2013]. Candidate match pairs from top graph form nodes in this graph and thin black edges are added between match pairs that share an observation in frame 2, and thick colored hyperedges represent additional constraints that mush be enforced so that each observation is used only once in the matching solution. **Image source**: [Butt 2013].

selecting a hypothesis which divides the set of detections into disjoint subsets, where each subset corresponds to a trajectory. Exhaustively evaluating the space of hypotheses is too

slow and hence authors use Markov Chain Monte Carlo (MCMC) sampling [Andrieu 2003] to efficiently explore the space of data associations by generating a sequence of sampled hypotheses. These sampled hypotheses are then distributed according to their relative probabilities which are defined by the likelihood function. The random nature of MCMC helps to prevent the search from becoming stuck in local maxima of the likelihood function.

A strength of this framework [Benfold 2011] is in explicit false positive modeling. This is performed by creating a separate model for false positives, and subsequently combining the data association step with the identification of false positives. A false positive track has no movement, while a true positive track is supposed to have at least a small movement. Streaming trajectory computation is performed by a sliding window approach. The authors achieve a real time performance by a carefully implemented multi-threaded approach. The model parameters are learned using Metropolis Hastings algorithm [Metropolis 1953], [Ge 2008] which takes 1-2 hours of computational time.

Other notable works utilizing MCMC based approaches are by [Choi 2010], [Khan 2005].

5.4 Multiple hypothesis tracking and Probabilistic data association techniques

Multiple hypothesis tracking (MHT) was first proposed by [Reid 1979]. The core idea is to memorize several data association hypothesis at several time steps and choose the best at each time step. This enables the tracker to resolve ambiguities from the distinguishing evidence obtained at later time steps. Although this technique is simple to implement, the algorithm is seldom employed due to its high computational complexity (NP-hard).

Probabilistic multiple hypothesis tracking (PMHT) proposed by [Gauvrit 1997] is an offline (or batch mode) data association technique wherein a posterior estimation of the state space is made only when all the measures (hypotheses) are available.

Gelgon et al. [Gelgon 2005] introduce a multiple object tracker which utilizes 2D segmentation across a video batch to stitch trajectories of objects in a scene. Plausible object trajectories are computed by incorporating object motion and temporal geometry constraints. The data association is formulated in a PMHT manner and the optimization is performed using Expectation Maximization. An advantage of this PMHT tracker is that it avoids the NP-hard computational issue of the MHT approaches.

Apart from weak computational complexity for large number of objects in a scene, standard PMHT approaches are usually outperformed by Joint Probabilistic Data Association Filter (JPDAF) [Vermaak 2005]. However JPDAF requires the knowledge of number of objects in a scene, and provides suboptimal solutions as the association hypotheses are summarized onto one frame *i.e.* the current frame. Willet et al. [Willett 2002] argues that PMHT should be the preferable data association technique when the environment is benign and a track is unlikely to be lost.

5.5 Discrete-Continuous optimization for tracking

Works by [Andriyenko 2012, Milan 2013c] use a discrete-continuous optimization model to optimize detections and trajectories. Both these approaches require an overcomplete set of potential trajectories as input. To this end the authors of [Milan 2013c] use the output of [Pirsiavash 2011] as input potential trajectories. Hence these approaches could also be considered as offline trajectory refinement. The authors incorporate local and global cues on a Conditional Random Field (CRF). This discrete part of the model is then optimized by α-expansion [Boykov 2001] and message passing (TRW-S by [Kolmogorov 2006]); whereas continuous optimization is performed by a simplex-based search over the continuous parameters.

Local cues between detections include appearance and motion similarity, and are modeled using pairwise factors. An important contribution of [Milan 2013c] is the introduction of exclusion constraints for detections in the same frame. This exclusion constraint dissuade two bounding boxes in the same frame to obtain the same label. Trajectories (locations of bounding boxes) correspond the variables for the continuous part of the optimization. The knowledge of an overcomplete set of tracks facilitates the authors to have a prior on detection boxes which measures how well trajectories follow detector evidence. Global factors for trajectories include a higher order trajectory term which penalizes temporal shape changes based on acceleration and velocity.

The complete CRF model comprises thirteen parameters, and statistical data analysis on eight video sequences is used to tune these parameters.

5.6 CRF based approaches

Conditional random field models have been very successful in computer vision and has a long history in the tracking community. Some of the recent prominent CRF models for multiple object tracking are by [Yang 2011], [Yang 2012], [Heili 2013], [Heili 2014b].

Yang et al. [Yang 2011] presents a hierarchical approach to tracking where pairs of input low level tracklets form graph nodes in a CRF. Another work by Yang et al. [Yang 2012] not only consider producing discriminative models for all targets but further consider discriminative features for distinguishing difficult pairs of targets. An additional feature of this work [Yang 2012] is that it can learn discriminative target specific models online.

Heili et al. [Heili 2014b] have recently introduced a model with the following novelties:

- Similar to [Zamir 2012], the authors connect all possible pairwise links (excluding links between detections of the same frame) between detections belonging to a batch. However as opposed to a hierarchical track merging from different batches in [Zamir 2012], the approach by [Heili 2014b] consider explicit linkage between successive batches; thus avoiding probable error propagation from one batch to another and at the same time being more amenable to a streaming multiple object tracking than [Zamir 2012].

- As opposed to other CRF based models, [Heili 2014b] not only consider similarity hypothesis between trajectories but also consider dis-similarity hypotheses between

trajectories. Furthermore the reliability of the features are also considered explicitly in this model.

- An additional feature of this work is an *unsupervised* way of learning scene-specific model parameters.

More details on this work can be found in [Heili 2014a] (thesis).

5.7 Probabilistic occupancy maps for tracking

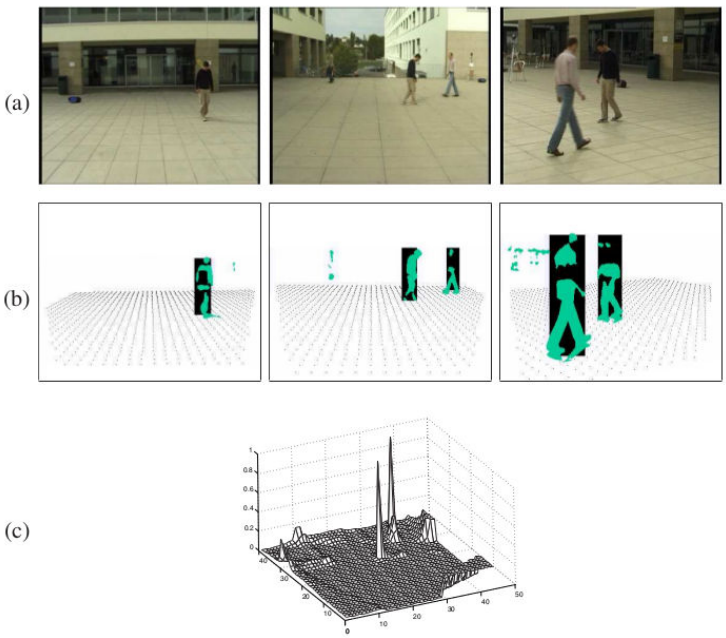

Figure 5.5: Probabilistic Occupancy Maps. **a:** Original Image from three cameras. **b:** Probabilistic Occupancy Maps for images in above row. **c:** Figure represents the corresponding occupancy probabilities on the grid. **Image Source:**[Berclaz 2011]

Almost all of the above approaches including our proposed approach is categorized under the *tracking using detections* class in the literature. One of the prominent approaches which does not use object detection hypotheses from detector is by [Berclaz 2011]. The authors use *probabilistic occupancy maps* (POM) which are similar to the foreground blobs obtained from a background subtraction process (*cf.* Figure 5.5). Subsequently the scene space is discretized into grid cells, and targets are forced to move along the grid. This leads to a temporally ordered directed graph based optimization model, wherein a global optimum is computed using K-shortest paths. This discretization is possible only for situations wherein the camera calibration parameters are known and hence these approaches cannot be applied to moving camera scenarios. The approach by [Berclaz 2011] is computationally fast but is prone to identity switches due to the lack of incorporation of appearance features and higher order motion information.

[Shitrit 2011] extends this work by adding global appearance constraints which provides robustness to identity switches. For identity preservation the authors add layers for each separate identity on the graph of [Berclaz 2011]. This requires the knowledge of the number of identity clusters. Edges are added to those grid cells which belong to same cluster. The optimization for this model is based on integer programming. Integer programming is NP-hard, and approximate or relaxed solutions generally require much memory and huge computational time. To this end the authors first employ the optimization by K-shortest paths from [Berclaz 2011]. This first run provides the knowledge of active grid cells and subsequently helps in reducing the problem size. Relaxed integer programming is then solved on this reduced problem size.

Another notable algorithm which does not use detector responses to obtain multiple object trajectories is by [Fragkiadaki 2011], where the authors first compute point tracks, and subsequently cluster them into person trajectories by Normalized Cuts.

5.8 Practical Considerations

Computational cost

A relevant aspect of tracking algorithms is their computational cost in practice. Often a multiple object tracker is used for high level computer vision tasks such as intrusion detection. Hence a tracker should be able to provide reliable trajectories in an efficient manner. Most approaches performing well [Milan 2013c, Zamir 2012, Shitrit 2011] mention speeds of several seconds per frame.

Optimizer Scalability

Another important practical consideration to select a tracker is the memory cost for the optimizer chosen on the model. Often a tracker is used for applications which run on general purpose laptops, for instance video in-painting. Hence an optimizer should be less memory intensive. Furthermore the optimizer should scale well w.r.t. the increase in the number of frames for a batch. Various exact optimization methods [Kappes 2013b], on tougher problems, consume much memory, and in our experiments are not able to work with HD videos of 500 frames batch size (10-15 persons per frame), on a 16 GB computer. Furthermore if the problem is tough to optimize, then the optimizers (chosen) should be allowed to be stopped before convergence, while still providing reliable outputs. Various exact methods, discrete-to-continuous relaxation methods, Lagrange relaxations do not provide guarantee that the solution is usable at all instants of their run-time. Said in another way, the solution obtained at a later time instant by relaxation-based optimization methods is not always guaranteed to be better than the current time solution. On the other hand, optimizers based on move-making methods are usually amenable to stopping to respect time-criticality requirements, as the solution obtained in subsequent iterations is as good or better than the solution at the current time. However most of these methods are sensitive to the initialization.

Method	Optimization	TF	GAF
Network Flow ([Pirsiavash 2011])	K-Shortest Paths	×	×
Maximal Cliques ([Zamir 2012])	Tabu Search	×	✓
Trajectory Refinement ([Milan 2013c])	α-Expansion, TRW-S	✓	×
Minimum Description Length ([Benfold 2011])	Markov Monte-Carlo	×	×
Occupancy Maps ([Shitrit 2011])	Integer Programming	×	✓
Modified Network Flow ([Butt 2013])	Lagrange Relaxation	✓	×
CRF ([Heili 2014b])	Hungarian and ICM	×	✓

Table 5.1: State-of-the-art approaches and their incorporation of local and global clues. TF stands for the curvature factor (triplet). GAF denotes global appearance factor.

5.9 Summary

In this chapter we have presented an overview of various state-of-the-art models for multi-object tracking. A summary of the state-of-the-art approaches and their incorporation of various local and global cues are shown in Table 5.1. TF corresponds to a triplet factor and GAF refers to a global appearance factor. TF penalizes high curvature changes and is a necessary factor to ensure smooth velocity [Collins 2012]. The network flow models lack this factor, and recently [Butt 2013] proposed a modification of the flow based tracking model to incorporate a TF for detections on three successive frames. This modification, however makes the energy non-submodular and hence efficient max-flow algorithms cannot be employed. [Andriyenko 2012, Milan 2013c] incorporate a global higher order factor to penalize high jumps (such as TF) in their discrete-continuous optimization model. Given a potential set of trajectories, their continuous model optimization deals with factors such as global trajectory straightness. Hence this discrete-continuous model is a form of a *labeling* problem as the prior is informative. A problem is referred to as a *partitioning* problem if the prior is uniform or non-informative.

GAF facilitates occlusion modeling in a principled way for a tracking model. The absence of which forces the re-run of optimizers by introducing hypothetical nodes corresponding to the occlusions. Network flow models by [Zhang 2008], [Pirsiavash 2011] lack this term.

Subsequent chapters in this part of the thesis describe our proposed approach for the task of multiple object tracking. Chapter 6 provides detail of our proposed model and optimization technique for multiple object tracking. Subsequently Chapter 7 includes implementation detail of our proposed tracker. This chapter also provides extensive evaluation of our tracker w.r.t. the state-of-the-art. Finally this part of the thesis will conclude with Chapter 8, which deals in applying a recent statistical measure for computing correlations between random variables to the task of person re-identification.

CHAPTER 6

Multiple Object Tracking using Graph Partitioning

Contents

6.1 Introduction 93
6.2 Model 94
6.2.1 Objective for a multi-object tracker 94
6.2.2 Criteria 95
6.2.3 Approach and Formalization 95
6.2.4 Repulsive constraints 96
6.2.5 Temporal neighborhoods and Point tracks 96
6.2.6 Appearance connections 98
6.2.7 Favoring smooth trajectories 99
6.3 Optimization 101
6.3.1 Unifying cues 101
6.3.2 Graph reduction 102
6.3.3 Optimizer selection 103
6.3.4 Parameter setting 104
6.3.5 Graph cleaning 105
6.3.6 Streaming trajectories computation 105
6.4 Summary 106

We now present the second contribution of this thesis in the realm of multiple object tracking. This chapter describes the model ingredients and the optimizers used for our tracking approach. Subsequently in the next chapter we visit the experimental details and evaluations w.r.t. the state-of-the-art approaches.

6.1 Introduction

The tracking of objects, persons or animals is required for high-level vision inference systems, from action recognition to animal behavior analysis. Other applications of tracking include video editing, *e.g.* inpainting, wherein a good tracking system can help in reducing the manual effort to annotate the parts of the video to keep or to remove.

In this chapter, we address the problem of multiple object tracking in videos obtained from a single camera. As the standard validation benchmarks in this field are centered on pedestrian tracking, this chapter focuses on the case where the objects to track are persons.

We suppose that we are already given person detections in each frame, from any pre-trained detector. This detector is not assumed to be perfect, but may produce false negatives (persons missed) and false positives (extraneous detections). Our objective is to group these detections into consistent trajectories, as well as to eliminate false negatives and false positives. Furthermore, we aim to build a robust and computationally efficient tracker, which can provide a stable output.

With respect to the literature (*cf.* Chapter 5), our approach belongs to the class of tracking-using-detections and has the following advantages:

- A principled way of unifying natural constraints that arise in tracking.
- Thanks to an apt combination of optimizers (section 6.3), a novel triplet search for trajectory-curvature penalization (section 6.2.7), and graph reduction (section 6.3.2), the proposed approach is computationally efficient (in terms of both memory and time), and can operate on hundreds of frames of HD videos of town street on a simple machine.
- Last, but not least, the proposed tracker is self contained and unlike [Milan 2013c, Andriyenko 2012], we do not require beforehand the trajectories from an external tracker.

The following sections introduce our proposed model and the optimizers used. Subsequently the next chapter details upon the implementation, experimental results and processing times.

6.2 Model

This section describes the intuition and rationale behind the proposed tracking model. Firstly, we will visit the basic ingredients which every tracking model should incorporate. Subsequently the following sub-sections provide the details about the spatio-temporal cues utilized in our tracking model.

6.2.1 Objective for a multi-object tracker

Our proposition for a multi-object tracker is based on the following objectives:

- The outputs from a tracker should be of high accuracy.
- The tracker should be able to provide outputs in a computationally efficient manner. Furthermore, the outputs should be stable (of similar quality) when the tracker is re-run on the same problem.

6.2.2 Criteria

A reliable tracking-using-detection model should incorporate the following criteria:

(C_1) **Uniqueness**: An object label (identity) should never appear more than once inside any frame over the whole sequence; *i.e.* a person cannot appear at two locations at the same time frame (section 6.2.4).

(C_2) **Smooth Trajectories**: The trajectory of each object should be as smooth as possible in time. In particular, there should not be jumps in an object location in consecutive frames (sections 6.2.4 & 6.2.7).

(C_3) **Robustness to Occlusions and False Negatives**: The trajectories should not be lost or confused during occlusions or when several detections are missed (sections 6.2.5 & 6.2.6).

(C_4) **No Ghost Objects**: False Positives from the detector should be removed and not form output trajectories (section 6.3.5).

The incorporation of these traits into our model is elaborated in the following sections.

6.2.3 Approach and Formalization

A person detector is used to obtain detection responses for all frames of a batch. A batch is a buffer of a few seconds ($2 \sim 6$ seconds) of a video, and typically comprises 50 to 200 frames. We then build a graph $G = (V, E)$ whose nodes correspond to detection responses. Suitable edges are incorporated between nodes by considering spatio-temporal cues around detections.

Our aim is to find an optimal partition of G such that each part corresponds to the trajectory of one person.

The notations used in this chapter are presented below:

L	:	Number of possible labels (i.e. of graph parts).
$\mathcal{X}$	:	Set of all possible partitions $\subset L^{\lvert V\rvert}$.
i	:	A node of the graph, *i.e.* a person detection.
x_i	:	Label of the node i, to be found.
w_{ij}	:	Edge weight between nodes i and j, i.e. benefit for them to have the same label.

For a set of L labels, each of the $|V|$ nodes can then take any of L states. A labeling $\mathbf{x} = (x_i)_{i \in V} \in \mathcal{X}$ then defines a partition of V into subsets T_l assigned to each class $l \in L$. The quality of such a labeling, to be maximized, is the sum of weighted edges connecting vertices with the same label:

$$\arg\max_{\mathbf{x} \in \mathcal{X}} \sum_{ij} w_{ij}\, \delta_{x_i = x_j} \quad = \quad \arg\min_{\mathbf{x} \in \mathcal{X}} \sum_{ij} w_{ij}\, \delta_{x_i \neq x_j}$$

In the following sections, we define the quantities relevant to the problem of tracking, and show how to set the interaction terms w_{ij} to express them.

Notice the difference between a partitioning and a labeling problem. In a **labeling problem**, for each variable a prior information about its affinity towards labels is available. While a problem is referred to as a **partitioning problem** when this prior is non-informative or unavailable. The global swap of labels do not change the energy for the case of the partitioning problem.

One of the differences between our approach and [Milan 2013c] is the availability of the prior information. The latter approach needs a potential list of trajectories to have a prior on detection bounding boxes (*i.e.* graph nodes), and for this purpose the authors [Milan 2013c] use the output from another tracker. On the contrary ours is a partitioning model and we do not require any prior label information for detections.

6.2.4 Repulsive constraints

Adhering to the criterion C_1, it is imperative to have exclusion constraints in the model. The goal of these constraints is to prevent nodes in a same frame from being allocated the same label. To this end we set highly negative weights w_{ij} between all pairs of detections (i, j) from the same frames. We refer to this as *intra-frame* exclusion constraint.
In regard to the criterion C_2, one should also prevent objects from jumping from one location in a frame to a complete different location in the next frame. Otherwisely said, two nodes far apart in consecutive frames should not have the same label. To this end, we define a maximum object speed ($4m/s$) and set highly negative weights w_{ij} between detections (i, j) in consecutive frames if the speed required for such a displacement is higher. The speed between two such detections is converted from pixels/frame into m/s by multiplying them with the factor $2f/\min(H_i, H_j)$, where f is the frame rate and where H_i is the height (in pixels) of the bounding box of detection i. Indeed, $H_i/2$ is a simple yet practical and sufficient approximation of the number of pixels per meter, and it has the advantage of adapting automatically to the depth in the video. We denote these no-jump constraints as *inter-frame* constraints.

Figure 6.1 illustrates both exclusion constraints.

6.2.5 Temporal neighborhoods and Point tracks

This section describes the first affinity functions (based on long-term point tracks) for determining edge weights between nodes.

Motion estimates for each detection provide vital information regarding its possible temporal neighbors. Without such estimates, temporal neighborhoods could become significantly big; subsequently, in the literature, they are approximated by assuming linear motion or by fitting complicated motion models to the location of detections, which in turn restricts the amount of frames processed in one batch due to the large number of variables. Moreover, detections may be infrequent (false negatives) and may contain false positives, which prevents motion models from obtaining good motion estimates.

Owing to these reasons we do not rely only on the location of the detections, but instead we compute point tracks (*cf.* Figure 6.2), based on optical flow, which provide motion estimates on a pixel (or sub-pixellic) level. The proportion $F(i, j)$ of common point tracks

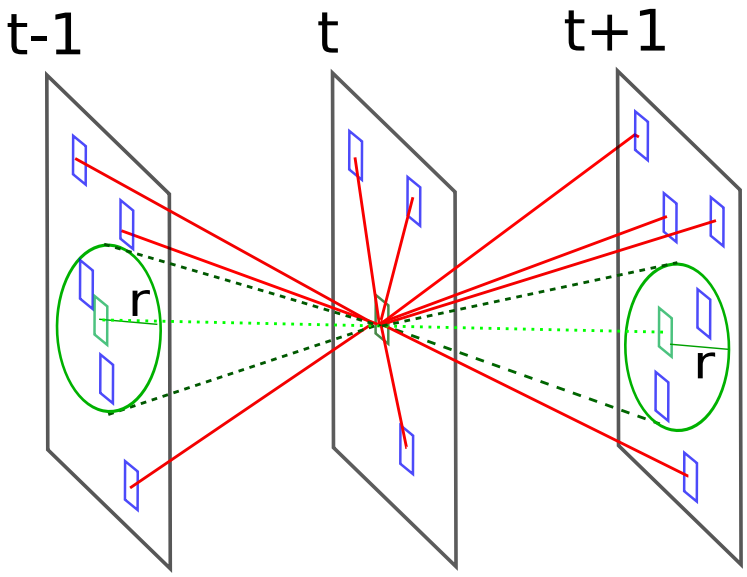

Figure 6.1: High repulsive exclusion constraints, shown for the central green bounding box in frame t, with **red** edges. Intra-frame exclusions prevent this detection from having the same label as any other detection in the same frame t, while inter-frame exclusion constraints prevent it from having the same label as detections in previous and next frames $t-1$, $t+1$ that are too far. The maximum radius **r** is detection-dependent, in that it corresponds to the maximum physically-plausible displacement within a duration of one frame, and that speed estimation depends on object depth.

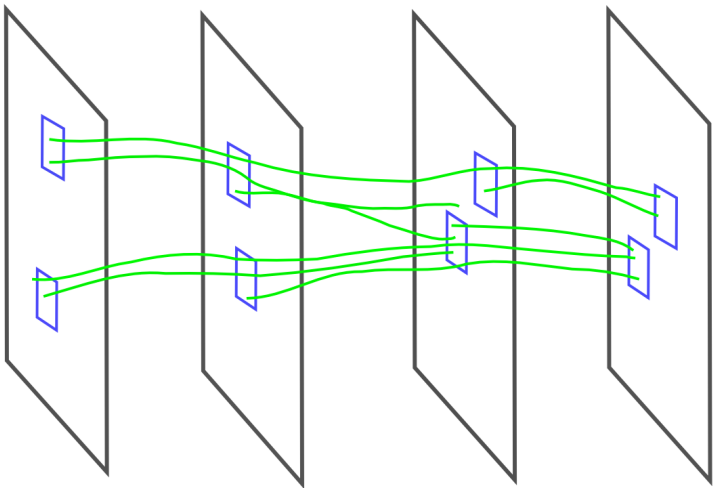

Figure 6.2: Point tracks and detections.

between two nodes i and j, i.e. of points tracks which go through two given bounding boxes B_i and B_j from different times, defines a first similarity measure between these nodes. This similarity measure, which we will consider in our energy functional (equation (6.2)), has the advantage of being robust to false detections (positive as negative) that occur between the two nodes. Also, in the case of partial occlusions, point tracks on the non-occluded part will still manage to follow the object and define a meaningful similarity, while overlapping detections make appearance-based affinities noisy in such partial occlusion cases.

The purpose of point tracks is thus threefold:

- Reduction of the temporal neighborhood of each node.
- Better similarity measure between nodes in cases of false negatives/positives or of partial occlusion.
- Reduction of the size of the original graph (*cf.* section 6.3.2).

Note that point tracks are obtained in a streaming manner, *i.e.* the implementation only have the knowledge of the current and one past frame.

6.2.6 Appearance connections

In the previous sections, apart from defining exclusion constraints, we have added connections between the nodes which share common point tracks, as an incentive for them to choose the same label and consequently be part of the trajectory of the same object. But point trajectories usually do not cover the full temporal span of the video and tracks may get lost after some time due to occlusion or pose changes. This calls for the need of long temporal range connections in the graph.

To define such a similarity between detections from any frames, possibly far apart in time, we consider the appearance of objects. To cater this, we compute an appearance feature for each bounding box, and cluster these appearances into K groups using K-means. The appearance cluster of node i is denoted by A_i. The appearance similarity

between any nodes belonging to a same cluster is then defined as a constant weight $\beta > 0$, while the one between nodes from different cluster is 0, which will lead to the quantity $\beta\ \delta_{A_i=A_j}$ in equation (6.2).

Each object gets a separate label, so K should not be set to be lesser than the number of objects in the scene. For appearance clustering we do not need to know the exact number of objects in the scene. The number of detections in a typical frame of a batch provides an important cue about the number of objects in a scene, as this quantity is not going to change dramatically in a space of couple of seconds. In our experiments we set K to twice the maximum number of detections observed in a single frame of the loaded video-batch during processing.

For all our experiments, we set the number of labels L (*cf.* section 6.2.3) in a similar manner as K.

To compute robust appearance features, we extract the image patch of the corresponding bounding box and lay a rectangular grid on this patch. For each cell in this grid, we compute its color histogram, dealing with each of the three color channels RGB independently. The appearance descriptor is then formed as the concatenation of the histograms of all cells. The distance between two appearances is computed by the L_2 norm. More details on the implementation of appearance feature is provided in the next chapter (Chapter 7).

Figure 6.3 shows some sample clustering from one video of the PETS09 dataset.

6.2.7 Favoring smooth trajectories

When the trajectories of people with a similar appearance cross each other, and that, in plus, full occlusion or detection mistakes (false positives or negatives) occur at this crossing, then the tracking based on the previous quantities may fail and induce small jumps in location or identity switches. This can be mitigated by penalizing high curvature of the trajectories. To this end we define an energy term which favors smooth trajectories, involving three detections j, i, k from different frames regularly spaced in time $(t - \delta t, t, t + \delta t)$, as shown in Figure 6.4. Given such detections, we compute a triplet factor $R(i, j, k)$ expressing the regularity of the associated trajectory. It is based on the Laplacian of the centroids p_i of the three bounding boxes:

$$\Delta_{ijk} \ = \ \frac{f}{H_i\ \delta t} \ \ \|p_j + p_k - 2p_i\|$$

and is expressed as:

$$R(i, j, k) \ = \ \frac{1}{\sqrt{\delta t}} \ \ \max(\tau - \Delta_{ijk}, \ \ 0). \tag{6.1}$$

Note that the Laplacian was normalized in order to be invariant to the video resolution and to the time interval, using the frame rate f, the frame interval δt and the bounding box height H_i used as previously as a depth/resolution indicator. R denotes the benefit for triplet j, i, k to belong to a same trajectory. The threshold τ indicates the maximum speed difference, or curvature, above which there is no gain anymore. We empirically set $\tau = 1m/s^2$, and is fixed for all experiments.

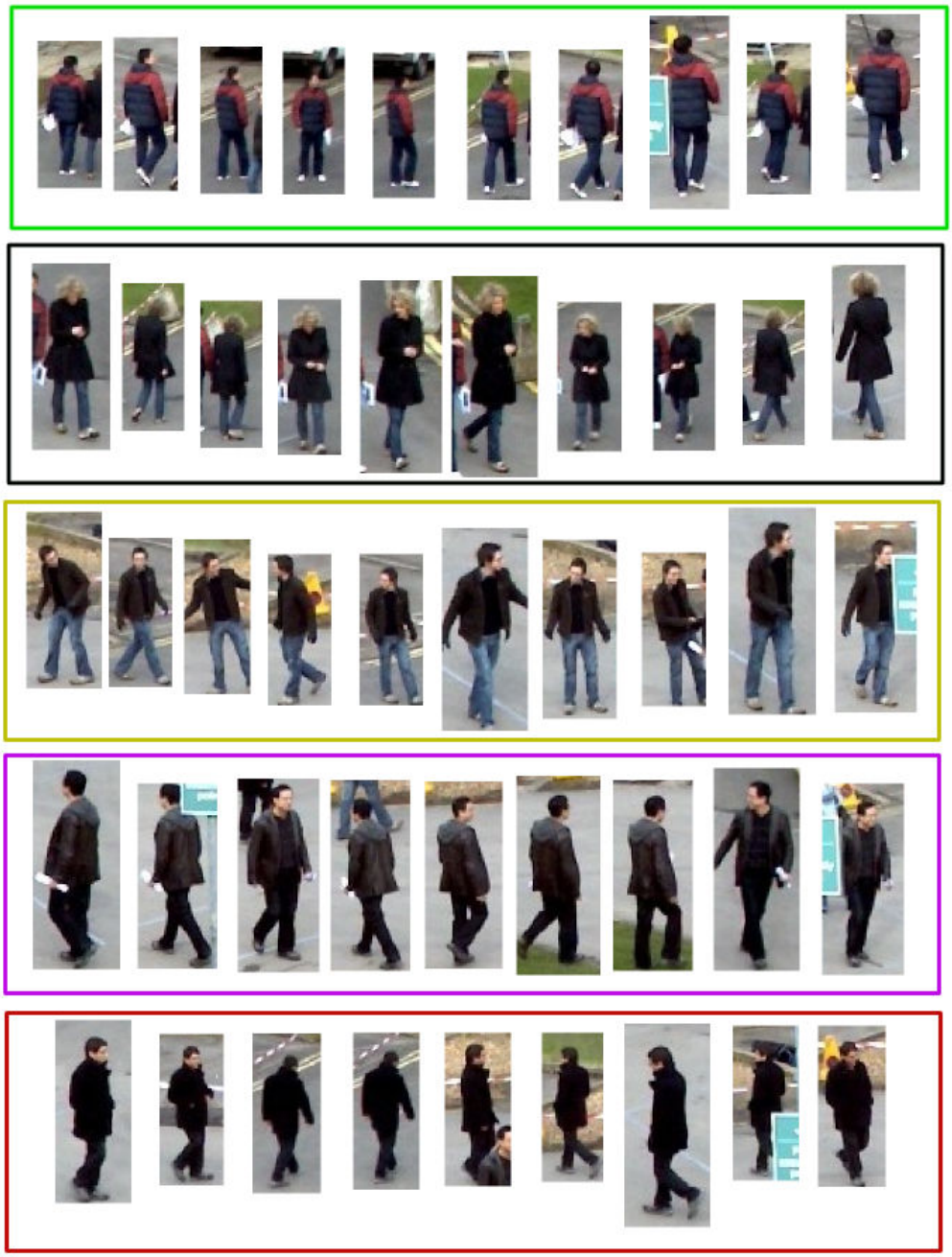

Figure 6.3: Sample Appearance Clusters for PETS09 S2L1: Each row displays a few samples belonging to the same cluster. Different sizes of cropped detections indicate the distance of persons from the camera. It can be seen that the appearance features are robust to scale changes.

Efficient triplet search

Naive search for all possible triplets in the graph is computationally prohibitive as there are $|V|^3$ possible triplets ($|V|^2$ symmetrical triplets) to enumerate. Also searching only temporally successive triplets ($\delta t = 1$) will be sensitive to missed detections. Thus we compute this factor on various temporal scales, with $\delta t \in \{1, 2, 3, 4, 5, 10\}$. As object trajectory is more predictable at shorter temporal ranges, we decrease the importance of the smoothness assumption for longer time spans with the factor $1/\sqrt{\delta t}$.

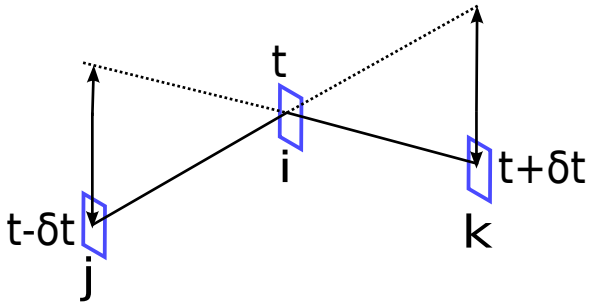

Figure 6.4: Computing the trajectory straightness (equation (6.1)) for a triplet of detections j, i, k at the temporal scale δt.

6.3 Optimization

This section provides detail on computing object trajectories using the cues defined in the previous sections. We first define a criterion unifying the spatio-temporal cues to compute multi-object trajectories, and subsequently provide a discussion on choosing an optimizer applicable to this criterion.

6.3.1 Unifying cues

We have the following similarity or dis-similarity between two or three nodes:

- Point tracks based similarity between nodes: $F(i, j)$.
- Similarity based on the global appearance: $\delta_{A_i=A_j}$, where $\delta_{\text{true}} = 1$ and $\delta_{\text{false}} = 0$. A_i denotes cluster identity for node i.
- A curvature smoothness term involving three nodes: $R(i, j, k)$.
- Repulsive constraints between nodes which cannot be the part of the same output trajectories (*inter frame* and *intra frame*).

Summing up all the quantities defined above, we obtain the following similarity criterion to maximize over possible labellings $\mathbf{x}$:

$$S(\mathbf{x}) = \alpha \sum_{ij} F(i,j)\ \delta_{x_i=x_j} \quad + \quad \beta \sum_{ij} \delta_{A_i=A_j}\ \delta_{x_i=x_j} \\ +\gamma \sum_{ijk} R(i,j,k)\ \delta^T_{x_i,x_j,x_k} \quad - \quad \Omega \sum_{(i,j)\in\mathcal{C}} \delta_{x_i=x_j} \tag{6.2}$$

where $\delta^T_{x_i,x_j,x_k} = \frac{1}{3}\left(\delta_{x_i=x_j} + \delta_{x_i=x_k} + \delta_{x_j=x_k}\right)$ is a good 2nd-order approximation of $\delta_{x_i=x_j=x_k}$. $\mathcal{C}$ denotes the set of all pairwise inter-frame or intra-frame constraints, and $\Omega > 0$ is sufficiently high to be dissuasive. Our aim is to compute a labeling $\bar{\mathbf{x}} = \mathbf{x}$ which maximizes $S(\mathbf{x})$. **Maximizing the similarity criterion $S(\mathbf{x})$ in equation (6.2) is equivalent to minimizing $E(\mathbf{x}) = -S(\mathbf{x})$.**

Submodular energy function: The pairwise terms in a submodular energy function favor smooth solutions where the neighboring labels (*i.e.* outputs) are the same. Hence costs of differing labels to the nodes connected by direct edges are greater than the costs for agreeing labels to these nodes. A submodular energy function can be efficiently optimized using available techniques such as *Graph cuts* or α*-expansion*. Our proposed model criterion $E(\mathbf{x})$ is not submodular due the repulsive constraints.

Note that if one denotes by $\mathcal{X}$ the set of feasible labellings, i.e. satisfying all constraints, then solving $\arg\min_{\mathbf{x}\in L^V} E(\mathbf{x})$ with high Ω is equivalent to solving $\arg\min_{\mathbf{x}\in\mathcal{X}} E(\mathbf{x})$ with $\Omega = 0$, and that $E(\mathbf{x})$ in this latter formulation is submodular on $\mathcal{X}$. However to our knowledge there is no optimizer dedicated to this.

6.3.2 Graph reduction

Point tracks can be used to reduce the initial graph, and consequently speed up the optimization, by fusing nodes for which there is no tracking ambiguity. More precisely, two

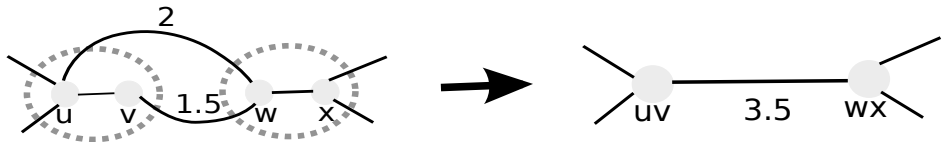

Figure 6.5: Edge weights between the fused nodes

nodes are fused if *all* non-background point tracks inside both bounding boxes move into one another. After fusing the nodes i and j, their edge weights (towards any other node k) are accumulated, as shown in the Figure 6.5 : $w_{\{ij\},k} = w_{ik} + w_{jk}$. Thus the criterion optimized does not change.

The ratio of the fused graph size to the original graph size translates directly to the optimization speed, as we observe a reduction in the computational cost by the same ratio. The Figure 6.6 shows a simple illustration for the speed gain due to this graph reduction step (*cf.* Figure 6.6 caption for details).

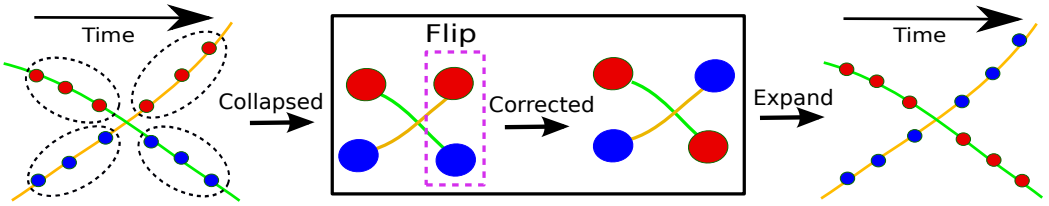

Figure 6.6: Graph reduction and flipper speed gain. Leftmost drawing shows the labeling of 12 nodes. Node colors indicate a labeling (stuck at a local minima) and the colored lines (*green* and *yellow*) correspond to the groundtruth trajectories. Middle inset drawing shows the reduced graph by fusing nodes inside the *dotted* ellipse in the Leftmost drawing (*cf.* sec. 6.3.2). Since the graph is reduced, we only need to search for flips of subgraphs of size = 2 (*i.e.* $O(|V|^2)$), instead of enumerating over all possible subgraphs of size = 6 (*i.e.* $O(|V|^6)$). The subgraph-flip required to correct the initial labeling is shown by *magenta* colored rectangular box in the middle inset. The Rightmost drawing correspond to the final correct output in the original graph.

6.3.3 Optimizer selection

In this section we discuss traits of some of the applicable state-of-the-art optimizers $E(\mathbf{x})$. The criterion in equation (6.2) is a Conditional Random Field[1]. A host of optimization techniques can be applied to this model.

The considerations for choosing an optimizer are outlined below:

- Fast convergence: The optimizer should be able to provide solutions in a computationally efficient manner.
- Scalability: The optimizer should scale well w.r.t. the problem size *i.e.* the optimizer should be able to handle batches of large sizes.
- Stable outputs: Outputs provided by the optimizer should be stable (*i.e.* of similar quality) in multiple runs.

Based on the above considerations, we discuss below the traits of the applicable optimizers, and the rationale behind our selection of the optimizers.

Heuristic approaches such as Tabu Search or Simulated Annealing can be used to optimize $E(\mathbf{x})$. However these suffer from at least two notable problems to be applied to a tracking model. Firstly, they do not provide any optimality indication certifying vicinity of a solution from being optimal. Secondly, in time critical applications it is often desirable to use techniques which can provide a usable solution (*i.e.* a solution satisfying all constraints) whenever required. These heuristic approaches along with relaxation techniques (*e.g.* Lagrangian relaxation), Polyhedral methods, do not provide a usable solution at all time instants of their operation.

Therefore we employ a fast message passing variant: Sequential Tree Re-Weighted Message Passing (**TRW-S**) [Kolmogorov 2006]. TRW-S considers the full graph for labeling and provides good optimums (along with the optimality bound). To make further

[1] An outline on Markov Random Fields can be seen in Appendix B.

improvements, we design local moves based on an Iterated Conditional Modes (**ICM**) variant, that will satisfy the constraints and are likely to be useful.

ICM [Besag 1986] is an iterative procedure, wherein an optimum labeling for a variable is chosen conditioned on all other variables. The process is repeated for all variables until convergence. Owing to this local nature of moves it is practically difficult for ICM to find a usable solution on its own. However with a good initialization, ICM can find more meaningful solutions quickly. [Andres 2012] proposed a variant of ICM: Lazy Flipper (**LF**). As opposed to ICM, a LF takes into account a large connected subset of variables up to a prescribed size. Upon convergence, the LF's labeling is guaranteed to be optimal within a Hamming distance equal to the subgraph size.

Inspired from LF, we search for flips of variables which reduce the energy, by considering a single or a subset of variables at a time. For a given subset size we repeat the above process until the energy cannot be reduced further. From our experiments, we find that considering a subset size of 3 or more is rarely needed. We also limit our search to only those subset of variables which have a joint temporal distance of less than 20 frames, i.e. the maximum time scale in triplets. This structural information incorporation brings significant computational speed up on this part of the optimization process.

Notice that this flipper is different from the Block-ICM proposed by [Collins 2012]. The authors in [Collins 2012] sweep in temporally forward manner, and check for flips **locally in two frames**. The flipper in our approach is more general as the nodes considered for flips have no time ordering restrictions.

We had experimented with recently proposed optimizers based on dual decomposition [Martins 2011] and linear programming relaxations [Sontag 2012]. Both these techniques were excruciatingly slow on finding one usable solution. In a recent work by [Kappes 2013b], the authors show that combinatorial methods (such as Integer Linear Programming, Max Cut using Reweighted Perfect Matching) based on branch-and-bound and cutting plane techniques do not always scale well with the problem size.

To sum up, in order to optimize $E(\mathbf{x})$, we first employ a fast *message passing* algorithm, namely TRW-S to compute an initial labeling. We improve this labeling by further optimizing $E(\mathbf{x})$ by using a move making method inspired from the Lazy Flipper ([Andres 2012]).

6.3.4 Parameter setting

We have three constant factors in the equation (6.2), namely α (point tracks), β (appearance) and γ (curvature). These have to be set considering the relative importance of each term, and this setting should be invariant to videos. The rationale in setting these parameters is outlined below.

- The main contribution of point tracks is to obtain motion estimates which helped in fusing the graph. On the fused graph, the nodes are either close to occlusion vicinity or false negatives. In both these situations, we need to look at appearance and trajectory curvature in time to decide about the labeling. Hence it is imperative to have more weightage on triplets and appearance clusters, than on point tracks.

- Near occlusion vicinity, it is important to consider both motion difference and appearance. However trajectory curvature in short temporal scale should have higher weight than appearance due to following reasons: Firstly, trajectories are not supposed to change their path substantially in course of a few frames, and secondly, appearance of persons can be similar either due to presence of noise or persons having same appearance.
  Hence we set positive values $\in \mathbb{R}^+$, for α, β and γ, in an increasing order, with α being the least. Precise values for α, β and γ can be found in the next chapter.

6.3.5 Graph cleaning

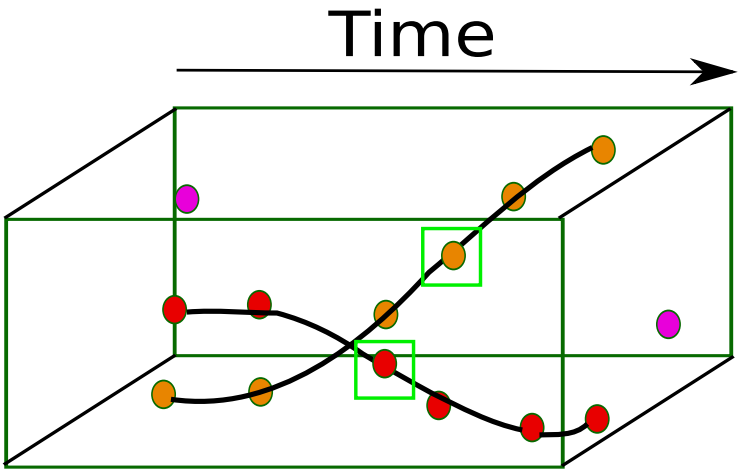

Figure 6.7: Illustration of the *graph cleaning* (*cf.* section 6.3.5) step. The colors of the nodes represent their associations, and same colored nodes belong to the same output trajectory. Nodes inside *green* boxes indicate interpolated detections for false negatives. *Magenta* colored nodes are false positives.

After computing trajectories for a batch, we now need to fill in false negatives along trajectories. Moreover, we also need to remove the false positives from the detector.

Any temporal gap along a trajectory due to false negatives is filled by assuming linear motion between the closest detected bounding boxes on either side of the temporal axis. False positives from detections usually form short trajectories and with long temporal gaps. Figure 6.7 provides an illustration of this graph cleaning step. In this figure, after the optimization step, we obtain 3 labels (*i.e.* trajectories). *Magenta* colored nodes form a trajectory with a long temporal gap, and hence we remove this trajectory (false positives). Nodes surrounded by the *green* colored boxes are the estimation for false negatives using linear interpolation.

6.3.6 Streaming trajectories computation

In order to apply this proposed tracker to a streaming video, we compute trajectories in batches of 50 to 100 frames (*i.e.* 2 to 4 seconds of HD video). Temporal merging of

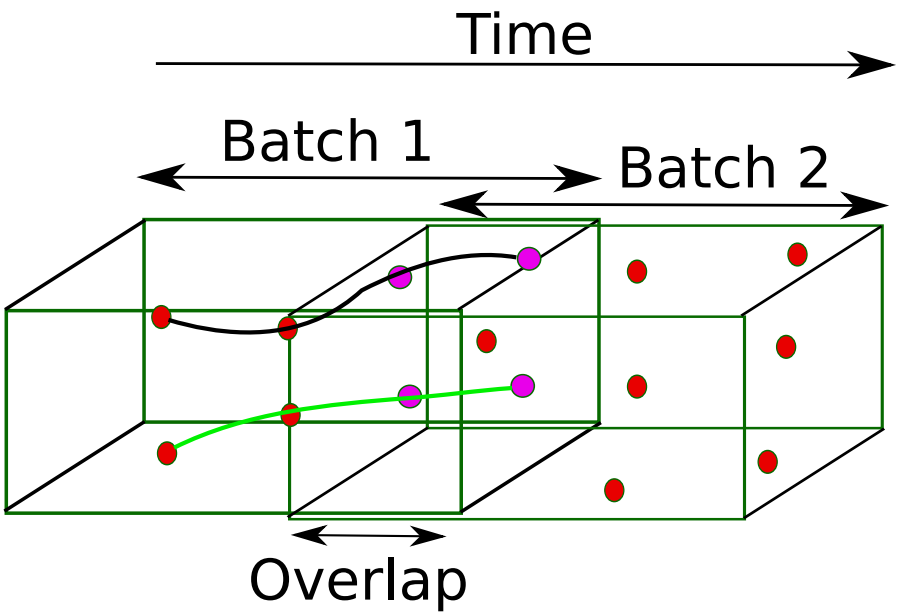

Figure 6.8: Streaming trajectory computation. Green and black colored lines represents two trajectories computed in batch 1. *Magenta* colored nodes in the overlap time span, which belong to the same trajectory (from batch 1) are **fused** together for batch 2.

trajectories is performed using a *sliding window approach*, with an overlap of 15 frames from the previous batch. Once trajectories for a batch are computed, the temporal span of the successive batch is chosen so as to have an overlap with the current batch as shown in the Figure 6.8. Thanks to our graph fusing step (*cf.* section 6.3.2), we are able to fuse all detection boxes (in the current batch) belonging to the same trajectory from the previous batch. The Figure 6.8 presents an illustration of a streaming video computation for the two temporally successive batches.

6.4 Summary

This chapter describes the second contribution of this thesis relating to the task of multiple object tracking. Our approach belongs to the class of batch-mode tracking wherein we stack successive frames of a video to infer object trajectories. An out-of-the-box person detector is applied to all frames of the video for obtaining detection hypotheses. The tracking objective is then modeled as a *graph partitioning problem*. Various cues from point-tracks, global appearance and trajectory straightness are then incorporated onto a discriminative markov random field. Subsequently this model is optimized by a combination of message passing and a move making variant. This proposed model has two degrees of freedom and all parameters are set empirically.

In the next chapter we describe the implementation of this proposed model, along with demonstrating qualitative and quantitative results. We also provide a detailed quantitative analysis of the optimizer used, and a thorough comparison with the state-of-the-art.

CHAPTER 7

Multiple Object Tracking: Implementation and Experiments

Contents

7.1 Implementation 107
7.2 Datasets 111
7.3 Inconsistency in evaluations by different proposed approaches 112
7.4 Evaluating the proposed tracker 114
7.5 Experiments 115
7.5.1 PETS S2L1 115
7.5.2 Towncenter 117
7.5.3 Parking Lot 118
7.6 Optimizer Convergence 119
7.7 Processing Time 120
7.7.1 Usefulness of Graph Reduction 120
7.7.2 Comparing processing times w.r.t. the state-of-the-art 121
7.8 Conclusion 126

7.1 Implementation

The complete workflow of our proposed multi-object tracking algorithm can be seen in Figure 7.1. A model summary showing various hard and soft constraints is shown in Figure 7.2.

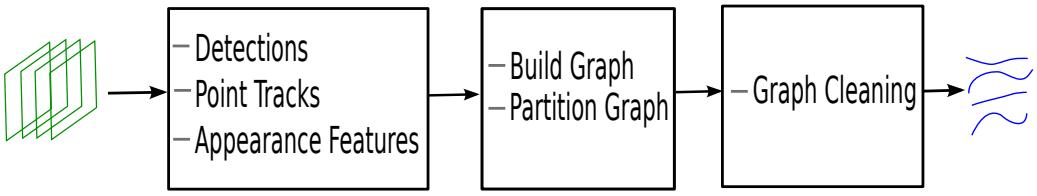

Figure 7.1: Workflow of the proposed approach.

- The **First block** in Figure 7.1 refers to the input video *pre-processing* to compute the ingredients necessary to build the graph (model). These are computed in the following manner:

  – Detections: We employ Deformable Parts Model (DPM, [Felzenszwalb 2010]) based person detector to obtain detection hypotheses. The C++ source code for the DPM based object detector can be obtained from [Dubout 2012]. Another option is to use OpenCV's implementation for HOG (Histogram of Oriented Gradients) [Dalal 2005] based pedestrian detector.

  For comparisons with the state-of-the-art we use the detections made available by these approaches. Information about exact detections used is provided as and when required.

  – Point tracks are computed by the algorithm from [Sundaram 2010]. The authors [Sundaram 2010] have shown that the point tracks obtained by their approach are more accurate than those obtained from the KLT based feature tracker.

  Given the optical flow (both forward and backward) for a video volume, the computational complexity of obtaining point tracks is linear w.r.t. the number of pixels, and any optical flow algorithm can be used for this purpose. More precisely, point tracks are built by checking forward and backward consistency of flow at each particular point.

  We use the optical flow algorithm from [Werlberger 2009] due to its efficiency in computational speed w.r.t. the *large displacement optical flow* by [Brox 2011]. The GPU code is available from the author's webpage.

  Pre-processing point-trajectories for background trajectories removal: Point tracks are computed on corner-like regions of the video and these are longer on the moving objects due to high structure variation at the boundaries of objects.

  Point tracks which lie on the background (corners on background) can be removed by simply removing tracks which are parallel to the time axis. In a stack of successive frames (*i.e.* a batch) of a video, points on static background form straight lines parallel to the time axis. Hence by identifying lines parallel to the time axis, we are able to remove the point tracks which are formed on the background-corners.

  Alternatively point tracks can be computed by finding corners inside the bounding boxes (as in [Benfold 2011]), which have very high probability of lying on the objects.

  – Appearance features: To compute robust appearance features, each detection bounding box is re-scaled into a fixed size window of 64×192 pixels. A set of rectangular sub-regions $\mathbf{P}$ is produced by shifting 32×32 regions with a 16 pixels step. This gives $|\mathbf{P}| = 33$ overlapping rectangular sub-regions. From each sub-region and for each of the RGB channels, we extract a 10-bin color histogram. Color dissimilarities caused by variations in lighting conditions are

minimized by applying histogram equalization to each color channel independently.
Other possibility is to use combination of several features *e.g.* gradients, colors, texture. More details about various state-of-the-art appearance descriptors for person-appearance matching can be found in the next Chapter.

- The **Second block** in Figure 7.1 refers to the tracking objective modeling and optimization phase.
  - Build Graph: Using the above ingredients, the graph is built with appropriate edge connections and weights.
  - Partition (Infer Trajectories): OpenGM library [Kappes 2013a] provides a host of discrete optimization routines. Once the problem is modeled (implemented) in OpenGM format, any of the provided optimization routines can be used on it. This facilitates testing multiple optimizers with ease. For our problem we use TRW-S code by [Kolmogorov 2006] wrapped in OpenGM.
    To minimize the energy further in order to obtain better outputs, we apply a *label flipper* based on the Lazy Flipper [Andres 2012] on the output by TRW-S.

- **Post-Processing** (or Graph Cleaning): We filter out the trajectories of short temporal span (5% of the video time span). Trajectories with long temporal gaps (10% of the video time span) are split into separate trajectories.

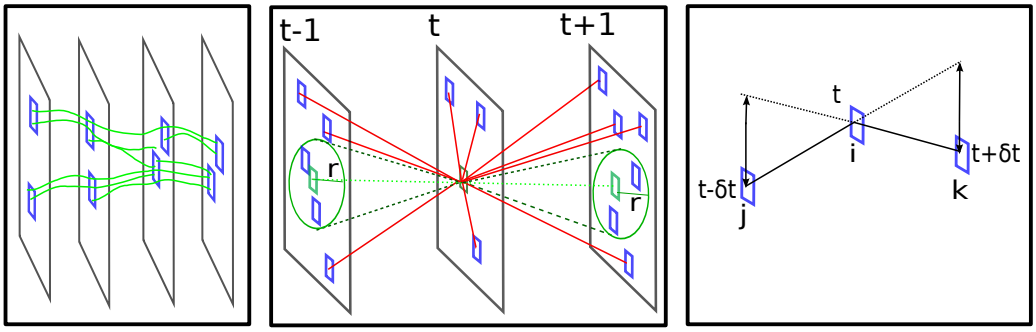

Figure 7.2: **Model Summary**. **Left**: Point Tracks and detections. **Middle**: High repulsive exclusion constraints, shown for the central green bounding box in frame t, with **red** edges. Intra-frame repulsive constraints prevent this detection from having the same label as any other detection in the same frame t, while inter-frame repulsive constraints prevent it from having the same label as detections in previous and next frames $t-1$, $t+1$ that are too far. The maximum radius **r** is detection-dependent, in that it corresponds to the maximum physically-plausible displacement within a duration of one frame, and that speed estimation depends on object depth. **Right**: Computing the trajectory straightness (*cf.* equation (7.2)) for a triplet of detections j, i, k at the temporal scale δt.

Energy Parameters

The values of the parameters in the energy $E(\mathbf{x})$ (*cf.* equation (7.1)) are quoted below. These values are set empirically and a small perturbation of these values do not change results considerably. The rationale behind such parameter setting is outlined in the previous chapter (*cf.* section 6.3.1).

$$\begin{aligned} E(\mathbf{x}) = -\alpha \sum_{ij} F(i,j)\ \delta_{x_i=x_j} \quad - \quad \beta \sum_{ij} \delta_{A_i=A_j}\ \delta_{x_1=x_j} \\ - \gamma \sum_{ijk} R(i,j,k)\ \delta^T_{x_i,x_j,x_k} \ + \ \Omega \sum_{(i,j)\in\mathcal{C}} \delta_{x_i=x_j} \end{aligned} \tag{7.1}$$

- α = 0.01
- β = 0.5
- γ = 50
- Ω = 10000000 (constraint)
- τ = 0.5 in equation (7.2)

$$R(i,j,k) \ = \ \frac{1}{\sqrt{\delta t}} \ \ \max(\tau - \Delta_{ijk}, \ \ 0). \tag{7.2}$$

Optimizer Parameters

- We set the number of iterations for TRW-S to 20.
- The *label flipper* neighborhood size is set to 2.
- This *label flipper* is considered as converged if the energy difference between two successive iterations is less than 1e-07.
- To adhere to the **time critical requirements** from a vision system perspective, we can stop the optimizer in two ways:
  - If the *label flipper* is running, then it can be stopped anytime.
  - If TRW-S is running, we wait until it finds a usable solution and stop the process. In all our experiments (400 frames of HD videos, 10-15 persons per frame), TRW-S never took more than 10 iterations for finding a usable solution.

7.2 Datasets

It is imperative that a vision system should demonstrate its applicability by showing comparative results with the state-of-the-art approaches. This underlines the need for diverse real word datasets. The importance of an object tracking task has led to many publicly available datasets, and we outline some of them below.

PETS [Ellis 2009]: IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (a.k.a. PETS) is held every year to provide a common ground for creating a benchmark datatset. This dataset from the year 2009 is one of the most used dataset for comparing tracking approaches. This data was recorded at the campus of University of Reading, United Kingdom. A total of 8 cameras were used for this recording. This dataset

Figure 7.3: Sample frames from the PETS dataset.

has videos with varying level of crowd density. Sample frames from one video of this dataset is shown in Figure 7.3. There is no official groundtruth for this dataset and many different annotations exist. We use the annotations provided by [Milan 2013a].

Figure 7.4: Sample frames from the TownCenter dataset.

TownCenter [Benfold 2011]: This dataset is recorded at a busy town-center street at Oxford, United Kingdom. This consists of one video of approximately 5000 frames. The video is high definition (HD, 1920x1080) with frequency of 25 frames per second. The groundtruth consists of 71500 hand labelled head and full body bounding boxes locations. On an average, 12-16 people are visible at any frame in this video. Figure 7.4 shows sample frames from this dataset.

Figure 7.5: Sample frames from the Parking Lot dataset.

Parking Lot [Shu Guang 2012]: This dataset consists of one high definition video of frequency 29 frames per second. The crowd density is modest with an average of 15 people visible for almost complete span of the video. The challenging aspect of this video is long-term inter-object occlusions, appearance similarity among objects and camera-jitter. Figure 7.5 shows sample frames from this dataset.

7.3 Inconsistency in evaluations by different proposed approaches

This section details on the pitfalls relating to an objective ground truth evaluation for various multi-object trackers. Comparative quantitative evaluation is challenging due to the lack of benchmark datasets, such as [Butler 2012] for optical flow evaluation purpose.

There is no official ground truth for many multi-object tracking datasets (including PETS) and annotations provided by various authors are sometimes inconsistent. The quality and level-of-detail can vary significantly across annotations, even for the same video sequence. Recently, [Milan 2013b] has thrown light on the ambiguities in evaluation due to dissimilarities in groundtruth annotations used by different tracking approaches. The authors [Milan 2013b] have systematically investigated the influence of different ground truth annotations, evaluation scripts and training procedures on publicly available data.

Owing to the importance of this recent study, we believe it is worthy to mention the key insights from this study. These are presented below:

- In an experiment, the authors evaluate the same tracking algorithm on different ground truth annotations. The results show that the gap in tracking metrics is noticeable when the ground truth annotations are changed.

- In order to gauge the difference in ground truth annotations, the authors evaluate three different ground truth annotations for the same video data w.r.t. one another. The match between two annotations w.r.t. the tracking metrics remains below 70%, and is indeed disturbing. This is mainly due to different bounding box sizes across annotations, leading to a lesser overlap and hence many boxes get counted as False Positives.

- On an another set of experiments the authors analyze the impact of different codes (implementations) for evaluation metrics. The figures for the evaluation metrics do not deviate substantially for the same tracking algorithm output. However, understandably the authors stress on the importance of using the exactly same evaluation software for a meaningful comparison.

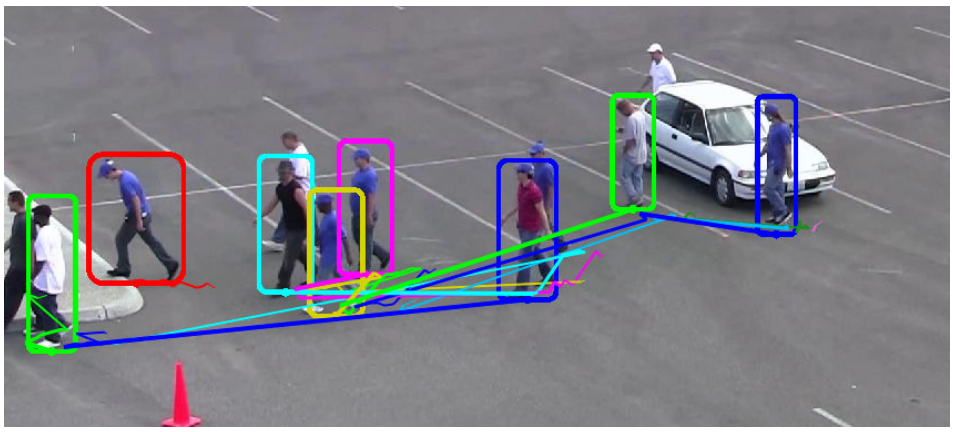

Figure 7.6: A sample frame from the PNNL ParkingLot sequence showing major deficits in the provided annotation. Besides mixing up the identities, several people are not marked in this ground truth. **Image source**: [Milan 2013a].

Also owing to the huge manual effort required, the available groundtruth annotations are often performed in a semi-supervised manner, leading to mistakes or incompleteness. A typical example is the Parking Lot dataset (*cf.* Figure 7.6); wherein the partially occluded pedestrians are not marked. Also the annotations for this dataset had some serious issues such as same identity label for two persons on a same frame. Another example of mis-annotation can be seen in Figure 7.7, wherein a person is not annotated for around 17 consecutive frames.

Figure 7.7: **Missing persons in annotation** : Frame numbers 509 and 526 from PETS S2L1 video. The person below the red arrow is not annotated for a length of 17 frames.

Owing to these findings, we provide the details about the detector used, and the code

employed for obtaining evaluation metrics. We will also provide the outputs for various videos on our web server.

7.4 Evaluating the proposed tracker

It is imperative to evaluate a proposed tracker w.r.t. the state-of-the-art approaches. This calls for the need of expressing the quality of tracking. Multiple object tracking has been an intensive area of research and hence several evaluation metrics already exist in the literature. For our evaluation purposes, we use the metrics proposed by [Li 2009] [1] and [Bernardin 2008].

Evaluation metrics by [Li 2009], provided below, focus on measuring the quality of output trajectories.

- **Recall**: percentage of correctly matched detections w.r.t. total detections in groundtruth.
- **Precision**: percentage of correctly matched detections w.r.t. total detections in tracking.
- **FAF**: average false alarms per frame.
- **GT**: Number of trajectories in groundtruth.
- **MT**: Ratio of mostly tracked trajectories, which are successfully tracked for more than 80% of their groundtruth time span.
- **ML**: Ratio of mostly lost trajectories, which are successfully tracked for less than 80% of their groundtruth time span.
- **IDS**: Number of times a tracked trajectory changes its matched id.

Another set of widely adopted metrics is from [Bernardin 2008]. It comprises two metrics namely Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP). These two metrics from [Bernardin 2008] are also popularly referred to as *CLEAR MOT* metrics.

MOTA takes into account the number of false positives, false negatives and ID switches. Equation 7.3 expresses MOTA, where m_t, fp_t, mme_t are the number of misses (a.k.a. False Negatives), of False Positives, and of mismatches (a.k.a. identity switches) respectively for time t.

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t} \tag{7.3}$$

MOTP quantifies the average distance between the groundtruth and estimated target locations. Refer to equation 7.4 for the expression of MOTP.

[1] We use the code provided by [Andriyenko 2012] to compute these metrics.

Out of the two CLEAR MOT metrics, MOTA is the preferred metric for us as we do not make any changes to the locations of input bounding boxes. We add a few bounding boxes for dealing with false negatives from the detector.

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t c_t} \tag{7.4}$$

7.5 Experiments

We performed extensive set of experiments in order to validate our tracking model. The datasets mentioned above are used for comparative evaluation purposes. Following sections provide qualitative and quantitative results.

7.5.1 PETS S2L1

For the PETS dataset, we use the S2L1 view. This video has a sparse level of crowd density and is challenging due to non-linear motion, and persons of similar appearances occluding each other. Since the optimization is fast, we show that it is possible to use the full video for testing our proposed approach [2].

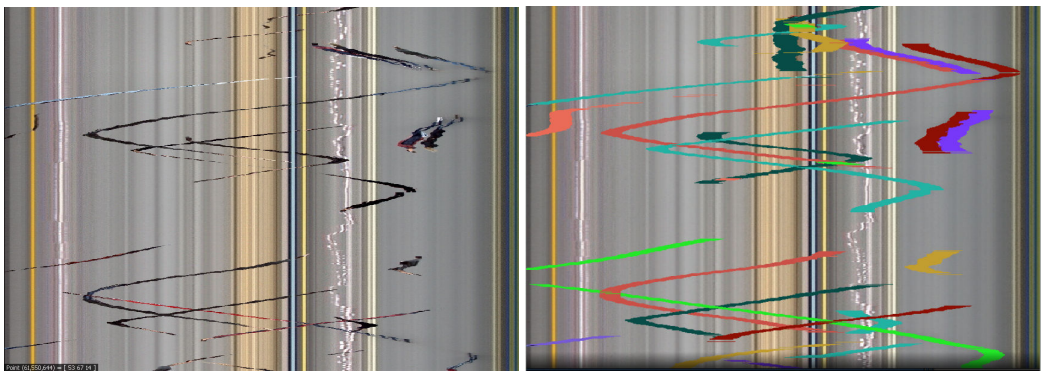

Figure 7.8: Video slices along the (x, t) plane, i.e. for fixed y, for PETS S2L1, showing that labels for trajectories before and after occlusions are maintained.

For this dataset, quantitative results are shown in Table 7.1 and visuals can be seen in Figures 7.8 and 7.9.

Following are the key observations:

- With our incorporation of global appearance features, we can keep the same ID for person exiting and re-entering the scene (*cf.* Figures 7.8 and 7.9 for visuals).
- From the values of evaluation metrics in Table 7.1, we can see that our approach performs well and is on par or better with the best in the metrics.

[2]Detections from http://iris.usc.edu/people/yangbo/downloads.html

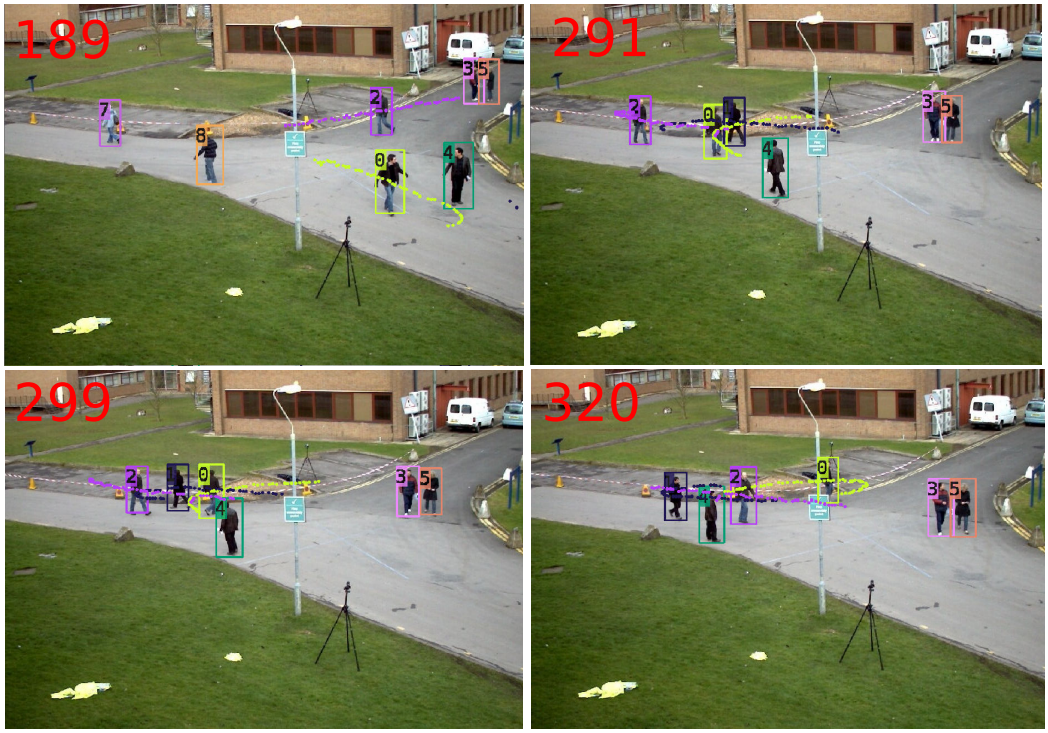

Figure 7.9: PETS S2L1 Output: The numbers in *red* on each image frames show the corresponding frame numbers from this video batch. The dotted points shows the trajectories of bounding boxes in previous and successive frames. To reduce clutter in display, we show 3 trajectories and their *few* past and future tracks. Notice that due to the global appearance incorporation, ID 4 is kept intact for the person as he leaves and comes back in the view. Also trajectories before and after crossing are fully consistent. This immaculate consistency despite False Negatives near occlusion vicinity is obtained due to the triplet factors.

- Unlike [Milan 2013c, Shitrit 2011, Berclaz 2011] we do not require any camera calibration. Furthermore contrary to [Milan 2013c], we do not rely on an external tracker. Also the model of [Milan 2013c] has 11 parameters and requires an explicit parameter estimation step for a video.
- [Zamir 2012] uses Tabu Search on their model and hence there are no guarantees for a stable output in multiple runs. Moreover [Zamir 2012] do not cater to a proper video streaming, as merging trajectories from batches is based on solving a clique problem by considering multiple batches. Typically better cliques can be found if more batches are considered for merging, which in turn adds to the latency time.
- A comparative processing speed is shown in Figure 7.15 (section 7.7), and it can be seen that we obtain high accuracy in a computationally efficient manner.
- The work by [Heili 2014b] was done in parallel with our work and [Heili 2014b] was published during the thesis review period. Nevertheless for the sake of completeness

Method	Rec	Prec	FAF	GT	MT	ML	IDS	MOTA (3D)	MOTA (2D)
[Milan 2013c]	96.9	94.1	0.36	19	18	0	22	90	83.6
[Shitrit 2011]	-	-	-	-	-	-	19	-	82
[Berclaz 2011]	-	-	-	-	-	-	28	-	80
[Zamir 2012]	-	69.4	-	-	-	-	8	-	90.3
[Heili 2014b]	96.7	98.3	0.10	19	18	0	**5**	94.9	88.7
Ours	95.8	90.8	0.28	19	18	0	13	90	81.8

Table 7.1: Comparison with recent proposed approaches on PETS S2L1 Video. The metrics on the first and last row are obtained using the same code, while the middle ones are taken from the respective papers as their result files are unavailable.

we include [Heili 2014b] in our evaluations. As per the comparative run times, our proposed approach is **4x** faster than [Heili 2014b].

Usefulness of Triplets: Triplets are necessary to maintain consistent identities in time. Indeed without using *triplets*, our results for MOTA (3D) for our approach would decrease to 84.4 and the *identity switches* would increase to 45.

7.5.2 Towncenter

We use this dataset extensively to show the evaluations concerning accuracy, speed and optimizer convergence. Figures 7.10, 7.11 and Tables 7.2 & 7.3 show qualitative and quantitative evaluations on this video data.

We select the approaches by [Benfold 2011, Zhang 2012] as the baselines for comparison. Although [Zhang 2012] is a frame-by-frame method, we compare to this approach as the output files are available, and it outperforms other approaches proposed prior to 2012.

Note that [Zamir 2012] outperforms [Zhang 2012] by gaining close to 3 more points in MOTA. However owing to the heavily stochastic nature of the optimizer used in [Zamir 2012], the authors of [Zamir 2012] could not provide us the output files, hence we chose to ignore this.

The key observations from Tables 7.2 and 7.3 are summarized below:

- Compared to [Benfold 2011], we achieve better MOTA. [Benfold 2011] uses a head detector and camera calibration along with motion estimates to make precise the location of body and head bounding boxes, hence achieves higher precision. Furthermore [Benfold 2011] require camera calibration and a parameter estimation step which take a couple of hours.

- While being competitive, we are 6 times faster on a single core implementation than the dual core implementation of [Zhang 2012].

- **Optimizer Scaling w.r.t. problem size**: In order to show the scalability of the proposed combination of optimizers, we consider a batch of 400 frames (for a density

Towncenter			
Method	MOTA	MOTP	Detector
Benfold et al. [Benfold 2011]	55.9	84.7	[Benfold 2011]
Zhang et al. [Zhang 2012]	64.9	76.4	[Benfold 2011]
Heili et al. [Heili 2014b]	69.4	71.8	[Felzenszwalb 2010]
Ours	64.6	75.0	[Benfold 2011]

Table 7.2: Quantitative Results on Towncenter Video batch for frames 1-1000. The authors of [Benfold 2011] use information from head detector along with other information such as point-tracks to refine the locations of bounding boxes, and hence the quantity MOTP is higher. MOTA is the preferred metric for us as we do not make any changes in the locations of input bounding boxes, apart from adding bounding boxes for dealing with false negatives from the detector. **Note** that the work by [Heili 2014b] was published during this thesis completion period.

Towncenter			
Method	MOTA	MOTP	Detector
Benfold et al. [Benfold 2011]	55.8	84.4	[Benfold 2011]
Zhang et al. [Zhang 2012]	66.4	75.2	[Benfold 2011]
Heili et al. [Heili 2014b]	72.5	70.0	[Felzenszwalb 2010]
Ours	66.5	75.8	[Benfold 2011]

Table 7.3: Quantitative Results on a crowded scene batch of 400 frames, starting from the 450th frame.

of 15 persons/frame). On this batch, we obtain MOTA (*cf.* Table 7.3) of 66.5 as opposed to 55.8 and 66.4 by [Benfold 2011] and [Zhang 2012] respectively. The memory consumption is also efficient as we use a maximum of 5GB of memory during the whole process.

- **Stable output on multiple runs**: We performed 5 trials on 50 frame batches for the first 1000 frames, and obtain a stable MOTA with a standard deviation of 0.7. In these trials we stop the optimizer to meet a requirement of **5-10 fps** (by stopping the flipper). Such stability guarantees can not be provided by heuristic (or stochastic) optimization approaches *e.g.* Tabu Search.

7.5.3 Parking Lot

This is a challenging dataset for both detection and tracking. Persons are severely occluding each other for most of the time span and hence detectors have difficulty in localizing them.

Quantitative results for this video are shown in Table 7.4[3]. We use the groundtruth files from the respective authors webpage.

[3]MOTA code from https://github.com/glisanti/CLEAR-MOT

Parking Lot 1			
0-200 frames			
Method	MOTA	MOTP	Detector
[Zamir 2012]	75	80	[Felzenszwalb 2010]
Ours	83.6	74	[Shu Guang 2012]
0-255 frames			
[Zamir 2012]	78.2	78	[Felzenszwalb 2010]
Ours	76.1	76	[Shu Guang 2012]
0-690 frames			
Method	MOTA	MOTP	Detector
[Zamir 2012]	89	78	[Felzenszwalb 2010]
Ours	64	68	[Shu Guang 2012]

Table 7.4: Quantitative Results on Parking Lot Video. Note that the groundtruth annotations exclude some non-occluded pedestrians and there is ID ambiguity at some frames.

From the results it can be seen that we achieve competitive accuracy in comparison to [Zamir 2012] in first two batches. Note that our approach is much more deterministic as opposed to [Zamir 2012] where in short tracklets in smaller video batches (about 50 frames) are found using Tabu Search (TS), and the subsequent merging of tracklets is also performed using TS. Hence there is no stability guaranty in obtaining similar results in multiple runs.

Along with providing no guarantees of a stable output on multiple runs, the approach by [Zamir 2012] do not cater to a proper video streaming, as merging trajectories from smaller batches is based on solving a minimum clique problem by considering multiple batches. Typically better cliques can be found if more batches are considered for merging.

The detections provided by [Shu Guang 2012] has a lot of false negatives and many persons are missed for a period of more than 20-30 frames. This impairs our triplet search and hence motion clues for curvature penalizing gets rarely incorporated for this video. A better triplet search considering density of people nearby could help. Also from the dataset, it appears that upper body detector and a head detector would be better suited for localizing persons.

7.6 Optimizer Convergence

All affinities in the model are expressed with negative numbers, and the repulsive forces are very high valued positive numbers (10^7). Hence if any of the constraints (inter/intra frame, repulsive) is not satisfied, then the energy $E(\mathbf{x})$ will have a positive value. However, if all of the repulsive constraints are satisfied, then $E(\mathbf{x})$ will have a negative value. Thus a *usable solution*, *i.e.* a solution satisfying all constraints, will have a negative energy.

Figure 7.12 shows the convergence speed of TRW-S on a problem size of 2000 variables, consisting of 300 frames of Towncenter video. The number of variables before *graph reduction* is 3700. Following are the key observations:

- TRW-S is able to find a usable solution very fast in its first iteration.
- The solution is usable and satisfies all constraints, and hence it justifies our usage of label flipper (based on ICM [Besag 1986], [Andres 2012]) to find better optimums swiftly.
- Other recently proposed state-of-the-art methods such as MPLP [Sontag 2012] is unable to find one usable solution in its first 100 iterations (*cf.* Figure 7.13).

7.7 Processing Time

Computational times are one of the major factors in choosing a tracking method. This section aims at providing relevant detail about processing times for our tracker.

For a batch of all frames of PETS S2L1 video, we use a maximum of 2.5 GB of memory (5 GB if the graph is not reduced) during the whole processing time on a single core Pentium 2.4 GHz machine. This video has a total of 4438 detections in 794 frames. The complete processing time is shown in Table 7.5.

# Frames	*App*+k-means	TRW-S	Flipper	Total
400 (With Graph Reduction)	7	40	90	137
794 (With Graph Reduction)	35	95	370	500
794 (Without Graph Reduction)	35	350	1500	1885

Table 7.5: **Optimizer scaling w.r.t. problem size**. Computation times (in **seconds**) for different batch sizes for PETS S2L1 video. Times in column *App*+k-means measure both appearance feature computation and clustering processing. For a more practical streaming trajectory computation on 50 frame batches, our optimizer takes 0.016 s/frame (on an average). Refer to the Figure 7.15 for comparative details on processing times.

7.7.1 Usefulness of Graph Reduction

Graph reduction using point tracks (*cf.* section 6.3.2), as a pre-processing step, not only helps in reducing the problem size, but also aids in swift enumeration for possible flips which in-turn speeds up the optimization process. Point tracks are an integral part of many vision systems and can be computed in real time on standard GPUs. [Senst 2012] provides high quality dense point tracks at 25 fps for high resolution videos on standard GPUs. Alternatively famous KLT trackers from [Zach 2008] achieve a speed of 200 fps using GPU, and are real time for low-end mobile graphic processors (on videos of frame size similar to PETS 09). Also as ours is a batch based tracker, a simple dual-threaded implementation with a dedicated thread for computation of point tracks for the successive batch can remove any probable latency.

In order to demonstrate the advantage of graph reduction, we consider the complete batch of 794 frames of the PETS S2L1 video, with or without the graph reduction step (*cf.*

Table 7.6). We observe a **4x** increase in efficiency with the graph reduction step. Similar comparative details for the Towncenter video can be seen in Figure 7.14.

# Frames in a batch	TRW-S	Flipper	Total	Memory
794 (With Graph Reduction)	95	370	**500**	2.5 GB
794 (Without Graph Reduction)	350	1500	**1885**	5 GB

Table 7.6: **Usefulness of graph reduction**: PETS S2L1 computation times (in **seconds**) for the cases when the graph reduction step is either chosen (*Row 1*) or discarded (*Row 2*). The column ***App+*k-means**, shows cumulative times for both appearance feature computation and clustering. The quantitative results for both rows are similar.

7.7.2 Comparing processing times w.r.t. the state-of-the-art

It is difficult to compare exact run times, as peers often do not give them (nor the code) or quote only average times over a set of varied videos. The run time of [Milan 2013c] optimizer for a 50 frame batch is 1-2 s/frame. In order to facilitate a direct comparison we compute times for all possible batches of 50 frames over all frames (PETS S2 L1). Our optimizer takes 0.016 s/frame in the worst case (8 persons/frame) on one 2.4 GHz core, which is **60x** faster than the best case of [Milan 2013c] in 2013. Average time for [Zamir 2012] is 0.088 s/frame on four 2.4 GHz cores. On videos similar to PETS S2L1, [Shitrit 2011] takes 4 seconds per frame on 3 GHz CPU. Please refer to Figure 7.15 for processing times comparison of our approach w.r.t. the state-of-the-art.

We perform optimization at a speed of 5-11 fps (on 50 frames batches the Towncenter with 10-17 persons/frame), which is **20x-40x** faster than the 4-core implementation of [Zamir 2012], and **6x-10x** faster than the 2-core implementation of [Zhang 2012]. [Benfold 2011] achieves a real time performance on Towncenter using parallel CPU implementation, however the tracking accuracy is significantly lower than us.

All our reported timings exclude the time taken by detector and optical flow. Optical flow is a pre-requisite for many high level vision algorithms which use tracking. Since we do not need precise flow estimates for all points in the video, one can safely employ fast KLT based point trackers on GPU/CPU. KLT tracker on HD videos run real time and many open source implementations exist. The authors of [Zach 2008] achieve tracking speed of 200 fps using GPU, and real time speed for low-end mobile graphic processors (on videos of frame size similar to PETS 09).

Figure 7.10: **Consistent people crossing.** IDs 0, 11, 13, 14 (in the left part of image frames) are of interest in **top rows**. IDs 17 and 19's (right part of the image frames) consistency is shown in the **bottom row**.

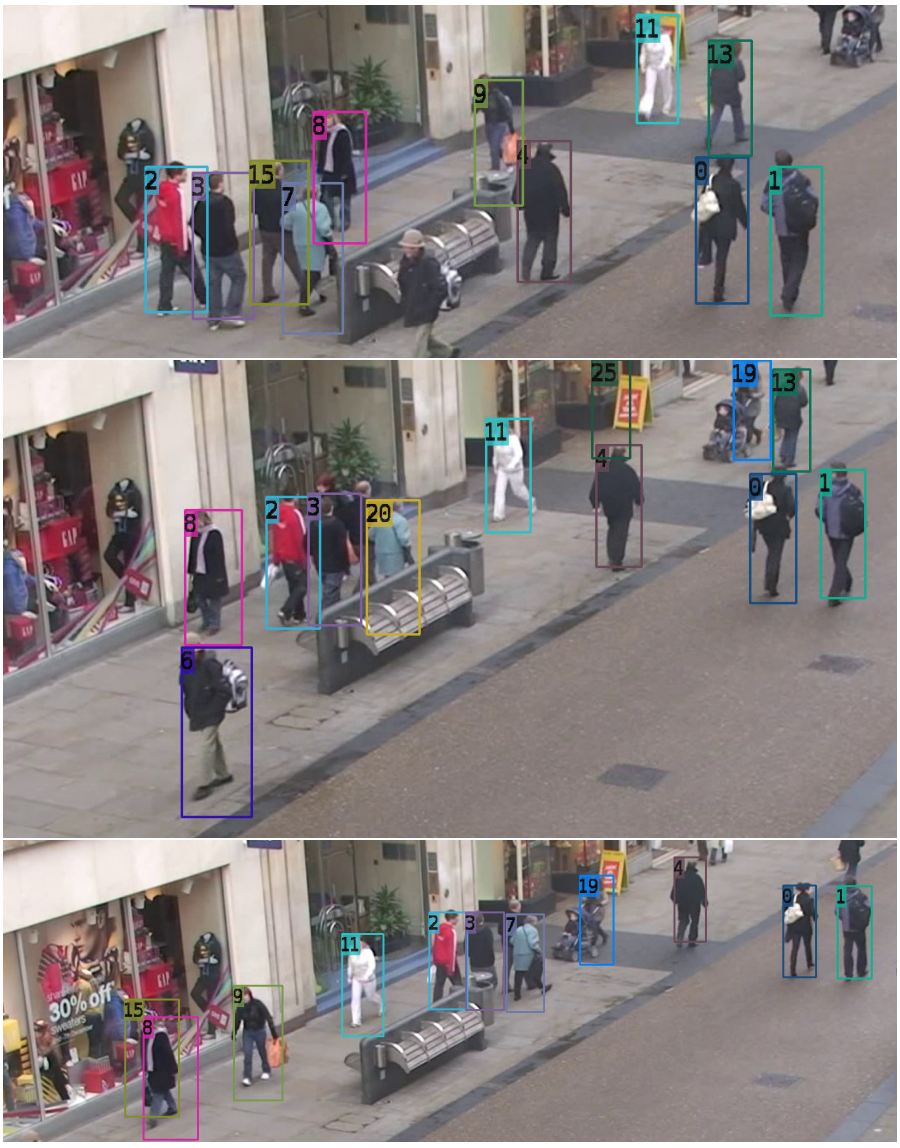

Figure 7.11: **Consistent people crossing in dense scenarios.** IDs 8, 9, 11, 2,3,7 are intact even after occlusion and false negatives. Images in the first row and last rows are 121 frames apart.

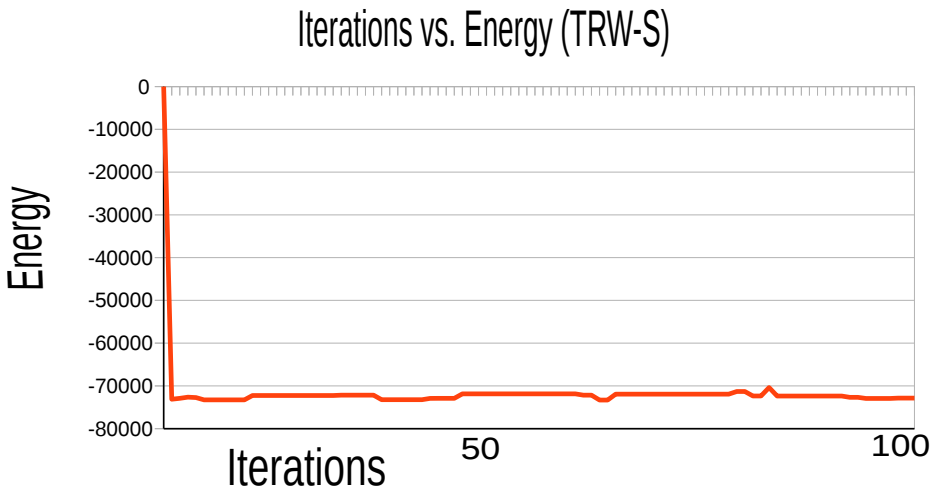

Figure 7.12: Iterations vs. Energy, on a problem size of 4000 variables. TRW-S is able to find usable solutions from first iteration (*cf.* section 7.6).

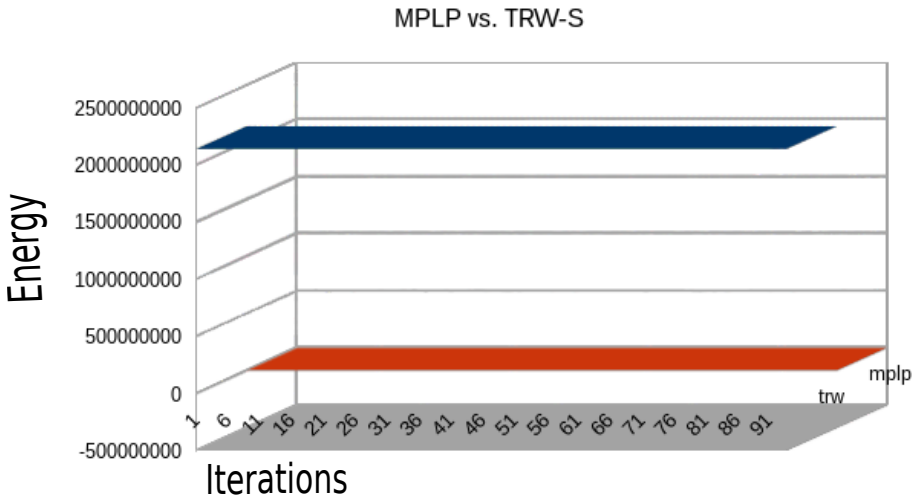

Figure 7.13: Iterations vs. Energy. The RED line corresponds to TRW-S, while the BLUE line corresponds to MPLP [Sontag 2012]. We observe that TRW-S is fast to minimize energy, and MPLP is unable to find one usable solution. Similar is the issue with another polyhedral method AD3 (*cf.* section 7.6).

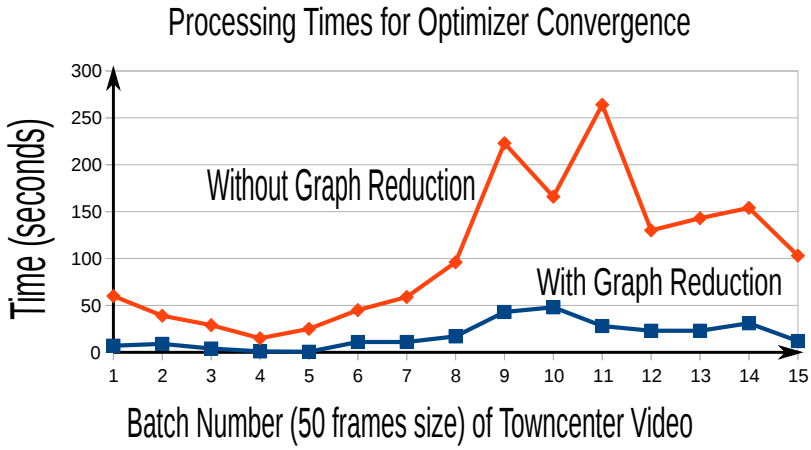

Figure 7.14: Processing time improvement by using graph reduction (similar MOTA in both cases), for the first 15 batches. High processing times from batch 8 to 12 are due to high detection rate/frame. In the above cases we let the optimizer run until convergence.

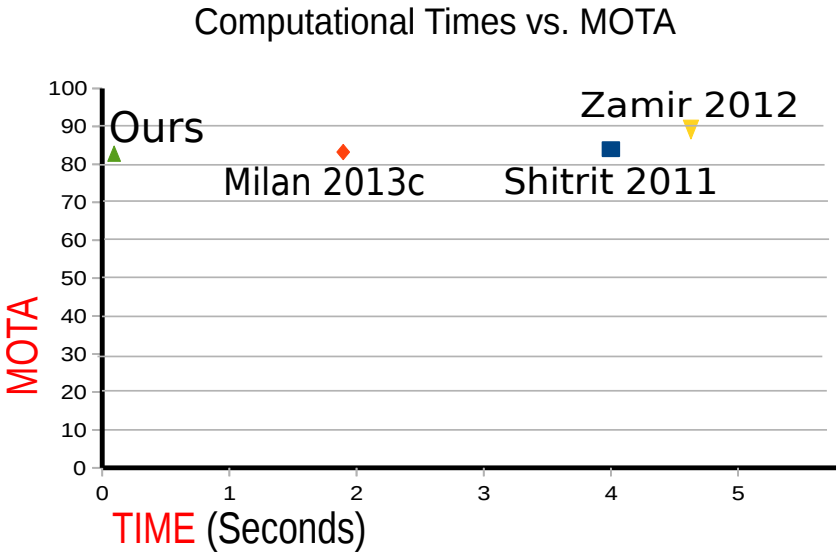

Figure 7.15: Average optimization processing times for the state-of-the-art tracking approaches. Note that this comparison is approximate due to the reasons mentioned in section 7.7. Nevertheless there is a significant gap indicating good computational efficiency of our approach. Note that we have not added the time required by the external tracker for [Milan 2013c].

7.8 Conclusion

Method	Optimization	TF	GAF
Network Flow ([Pirsiavash 2011])	K-Shortest Paths	×	×
Maximal Cliques ([Zamir 2012])	Tabu Search	×	✓
Trajectory Refinement ([Milan 2013c])	α-Expansion, TRW-S	✓	×
Minimum Description Length ([Benfold 2011])	Markov Monte-Carlo	×	×
Occupancy Maps ([Shitrit 2011])	Integer Programming	×	✓
Modified Network Flow ([Butt 2013])	Lagrange Relaxation	✓	×
CRF ([Heili 2014b])	Hungarian and ICM	×	✓
Ours	TRW-S, Flipper	✓	✓

Table 7.7: State-of-the-art approaches and their incorporation of local and global clues. TF stands for the curvature penalization factor (triplet). GAF denotes global appearance factor.

We proposed a novel tracking algorithm which exploits local and global cues in the most unified manner as compared to the state-of-the-art (*cf.* Table 7.7). Furthermore an apt combination of optimizers and suitable graph reduction lends stability and efficiency (5-10 fps on HD video of busy town street, on a single core CPU) to the model. The results obtained are competitive or better than the state-of-the-art at this speed. The proposed model has two degrees of freedom, and hence would be highly amenable to parameter learning using scene context and other relevant inputs. Also the algorithm is easy to implement and reproduce. Future prospects include tracking object parts and providing feedback to the point-trackers about incorrect tracks.

In the next chapter we visit the appearance matching problem and elaborate on the third contribution of the thesis, along with the utility of having a state-of-the-art global appearance feature for occlusion management in our tracking model.

CHAPTER 8

Distance Correlation for Appearance Matching

(This work was done in collaboration with Dr Slawomir Bak, INRIA. The experimental sections 8.4 and 8.5 were undertaken by him.)

Contents

8.1 Appearance Matching and People Re-Identification 128

8.2 Classical Covariance 128

8.3 Distance Correlation $\mathcal{V}^2$ 130

8.4 Qualitative Results in Visual Tracking 132

8.5 Quantitative Results on Person Re-identification datasets 133

8.5.1 i-LIDS-MA 136

8.5.2 i-LIDS-AA 136

8.5.3 i-LIDS 136

8.5.4 Descriptor Efficiency 138

8.6 Perspectives on Single Camera Multi-Object Tracking 138

8.7 Conclusion 141

In this chapter, we visit the problem of appearance matching of persons across multiple cameras, and subsequently provide details about our proposed descriptor for appearance matching. We conclude this chapter with experiments relating to appearance matching and multiple object tracking.

In the previous chapters we have studied a multiple object tracker, model and its implementation. One of the main challenges of a multiple object tracker lies in handling occlusions. Quite often a person gets occluded and appears again in the scene. In our proposed model we handle occlusions by connecting detections having similar global appearances. Hence the *occlusion management* for our tracker requires good appearance features. This leads to a question as to whether the state-of-the-art appearance features can aid in computing better object trajectories ? Referring to this inquest we experimented with existing state-of-the-art appearance matching descriptors; and also proposed a novel descriptor for person matching across multiple cameras.

8.1 Appearance Matching and People Re-Identification

Appearance matching is one of the key building blocks in any computer vision system *e.g.* object recognition, tracking, image retrieval. This problem is challenging due to significant appearance changes, viewpoint shifts, lighting variations and varied object poses (*cf.* Figure 8.1). These challenges have led to the development of several features and their representations.

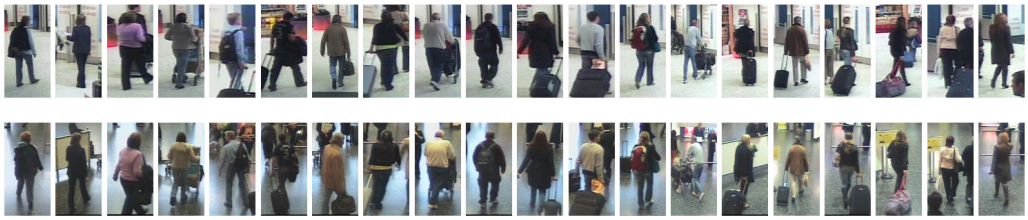

Figure 8.1: Top and Bottom Rows correspond to images from different cameras.

A good feature descriptor should be invariant to illumination, scale and viewpoint changes. Typically this involves a high dimensional floating point vector that encodes description for a local region. Some of the prominent examples are SIFT [Lowe 2004] and SURF [Bay 2006]. Another way of representing a local region is to employ statistics of features extracted by different kinds of image filters. Typical examples for this category include Histogram of Oriented Gradients [Dalal 2005] and Local Binary Patterns [Wang 1990].

Instead of concatenating filter outputs or statistics over the filter-outputs, a viable way of describing an image would be to encode the *inter-correlations* among filter outputs. This led to the emergence of *covariance descriptor* [Tuzel 2006]. This descriptor encodes information on feature variance inside an image region, their linear correlations with each other and the spatial layout. Over the years, correlation-based descriptors have shown a consistent invariance to many aspects (scale, illumination, rotation). Typical application areas are person detection, re-identification, object tracking ([Ma 2012], [Hirzer 2011], [Tuzel 2008], [Porikli 2006]).

The focus of this chapter is in exploiting recent advances in statistics to design feature descriptors based on inter-correlations among filter outputs. In the following sections, we will describe a covariance matrix and its limitations, followed by an introduction to *distance covariance*. Using this new covariance measure, we design a descriptor for identifying persons across multiple cameras. Finally, we describe the experimental results which demonstrate the descriptive power of our descriptor.

8.2 Classical Covariance

Let I be an image and F be the feature image extracted from I:

$$F(x, y) = \phi(I, x, y) \tag{8.1}$$

The function ϕ in equation (8.1) can correspond to any mapping such as intensity values, gradients and other relevant filter responses. Let us suppose that we have n such mappings (*i.e.* n feature images F), and we wish to encode the covariance of these mappings for a reliable description of the image. For a given rectangular region $R \subset I$ containing p pixels, let $\{f_k\}_{k=1...p}$ be the n dimensional feature points inside R. Each feature point f_k is characterized by a function ϕ. The covariance based description of R is the given by a nxn covariance matrix, where the (i, j)-th element is expressed as:

$$C_R(i,j) = \frac{1}{p-1}\sum_{k=1}^{p}(f_k(i) - \mu(i))(f_k(j) - \mu(j))^T, \tag{8.2}$$

where $\mu(i)$ is the mean of all feature points for a pixel.

The *Pearson Coefficient* is computed as follows:

$$\rho = \hat{C}_R(i,j) = \frac{C_R(i,j)}{\sqrt{C_R(i,i)C_R(j,j)}}, \tag{8.3}$$

Limitations of Sample Covariance

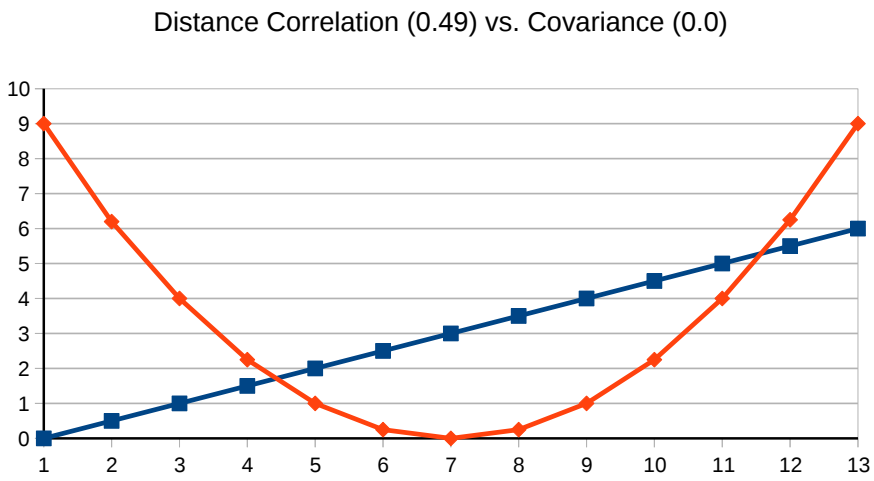

Figure 8.2: Correlation between a Parabola and a Line. We show that the classical covariance fails to compute any correlation, while the recently proposed *distance correlation* by [Szekely 2009] computes good correlation.

The covariance matrix (and ρ) computes linear correlation among variables (*cf.* Figure 8.2), and is due to its mean-dependent computations (*cf.* equations (8.2) and (8.3)). This is indeed a serious limitation as the data in feature spaces are, in general, highly nonlinear.

Furthermore, the covariance matrix does not lie on an Euclidean space. Hence to express distance between two covariance descriptors, we need to assume a manifold embedding to compute geodesic distance. Alternatively we can also map covariance matrices

onto a tangent space to approximate geodesic distance. Both of these solutions are computationally expensive, and are sometimes unaffordable in practice. Moreover, several machine learning techniques are not adequate for learning on complex manifolds, often producing over-fitted classifiers.

To summarize, following are the limitations of covariance matrices.

- It can compute only linear correlation.
- Expensive computations are required to compute the distance between two covariance matrices.

Recently the authors of [Szekely 2009] have proposed a novel covariance measure termed *Distance Correlation*. This theory inherits ideas stemming from the Brownian motion and measures dependence between random vectors in arbitrary dimensions. With a sufficient number of samples, the distance correlation can compute all kinds of possible relationships between random variables. In the following sections, we will describe the basics about computation of distance correlation, and the readers are encouraged to refer to [Szekely 2009] for more details and proofs. For the sake of simplicity we follow the mathematical notations from [Szekely 2009].

8.3 Distance Correlation $\mathcal{V}^2$

Let $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}^q$ be random vectors, where p and q are positive integers; f_X and f_Y denote characteristic functions of X and Y respectively, and their joint characteristic function is denoted by f_{XY}. In terms of characteristic functions, X and Y are independent iff $f_{XY} = f_X f_Y$. Thus a natural way of measuring the independence between X and Y is to find a suitable distance between f_{XY} and $f_X f_Y$.

Distance covariance $\mathcal{V}^2$ [Szekely 2009] is a new measure of dependence between random vectors and can be defined by

$$\mathcal{V}^2(X,Y) = ||f_{X,Y}(t,s) - f_X(t)f_Y(s)||^2 \tag{8.4}$$

$$= \frac{1}{c_p c_q} \int_{\mathbb{R}^{p+q}} \frac{|f_{X,Y}(t,s) - f_X(t)f_Y(s)|^2}{|t|_p^{1+p}|s|_q^{1+q}} dtds, \tag{8.5}$$

where c_p and c_q are constants determining norm function in $\mathbb{R}^p \times \mathbb{R}^q$, $t \in X$, $s \in Y$.

This measure is analogous to the classical covariance, but with an important property that $\mathcal{V}^2(X,Y) = 0$ if and only if X and Y are independent.

Sample distance covariance $\mathcal{V}_n^2$ between random vectors X and Y is defined as

$$\mathcal{V}_n^2(X,Y) = \frac{1}{n^2} \sum_{k,l=1}^{n} A_{kl} B_{kl}, \tag{8.6}$$

where A_{kl} and B_{kl} are simple linear functions of the pairwise distances between sample elements. These functions are defined in the following. For a random sample $(\mathbf{X}, \mathbf{Y}) = \{(X_k, Y_k) : k = 1 \dots n\}$ of n i.i.d random vectors (X, Y) from their joint distribution,

compute the Euclidean distance matrices $(a_{kl}) = (|X_k - X_l|_p)$ and $(b_{kl}) = (|Y_k - Y_l|_q)$. Define

$$A_{kl} = a_{kl} - \bar{a}_{k\cdot} - \bar{a}_{\cdot l} + \bar{a}_{\cdot\cdot}, \quad k, l = 1, \ldots, n, \tag{8.7}$$

where

$$\bar{a}_{k\cdot} = \frac{1}{n}\sum_{l=1}^{n} a_{kl}, \;\; \bar{a}_{\cdot l} = \frac{1}{n}\sum_{k=1}^{n} a_{kl}, \;\; \bar{a}_{\cdot\cdot} = \frac{1}{n^2}\sum_{k,l=1}^{n} a_{kl}. \tag{8.8}$$

Similarly, we define $B_{kl} = b_{kl} - \bar{b}_{k\cdot} - \bar{b}_{\cdot l} + \bar{b}_{\cdot\cdot}$. Although, the relation of equations (8.4) and (8.6) is not straightforward, in [Szekely 2009] we can find a theorem stating: *If* $E|X|_p < \infty$ *and* $E|Y|_q < \infty$*, then almost surely*

$$\lim_{n\to\infty} \mathcal{V}_n(X, Y) = \mathcal{V}(X, Y). \tag{8.9}$$

Standardization: Similarly to covariance which has its standardized counterpart ρ, $\mathcal{V}_n^2$ has its standardized version referred to as *distance correlation* $\mathcal{R}_n^2$ and defined by

$$\mathcal{R}_n^2(X, Y) = \begin{cases} \frac{\mathcal{V}_n^2(X,Y)}{\sqrt{\mathcal{V}_n^2(X)\mathcal{V}_n^2(Y)}}, & \mathcal{V}_n^2(X)\mathcal{V}_n^2(Y) > 0; \\ 0, & \mathcal{V}_n^2(X)\mathcal{V}_n^2(Y) = 0, \end{cases} \tag{8.10}$$

where

$$\mathcal{V}_n^2(X) = \mathcal{V}_n^2(X, X) = \frac{1}{n^2}\sum_{k,l=1}^{n} A_{kl}^2. \tag{8.11}$$

Summarizing Computation of Sample Distance covariance

This section summarizes the computation of sample distance covariance for two random variables.

The distance correlation for a random sample, $(\mathbf{X}, \mathbf{Y}) = \{(X_k, Y_k) : k = 1, \ldots n\}$ of n i.i.d. random vectors (X, Y) from the joint distribution of random vectors X in $\mathbb{R}^p$ and Y in $\mathbb{R}^q$, is computed as follows :

- Compute the euclidean distance matrices $(a_{kl} = (|X_k - X_l|_p)$ and $(b_{kl} = (|Y_k - Y_l|_q)))$.
- Compute doubly centered matrices $A_{kl} = a_{kl} - \bar{a_{k\cdot}} - \bar{a_{\cdot l}} + \bar{a_{\cdot\cdot}}$.
- Similarly to A_{kl}, compute B_{kl}.
- Distance Correlation : $\frac{1}{n^2}\sum_{j,k=1}^{n} A_{j,k}\, B_{j,k}$.

Computing descriptor based on Distance Correlation

We term our proposed descriptor based on distance correlation *Brownian Descriptor*. Let I be an image and $\mathcal{L} = \{L_1, L_2, \ldots, L_n\}$ be a set of feature layers ($\mathcal{L}$ is defined by the mapping ϕ, *cf.* section 8.2). The Brownian descriptor is obtained by computing $\mathcal{R}_n^2(\mathcal{L}, \mathcal{L})$, while keeping distance coefficients in the form of matrix (*cf.* Algorithm 3). Similarly

to the classical covariance matrix, the Brownian descriptor is represented by a positive definite and symmetric matrix and it provides a natural way of fusing multiple features. This descriptor does not contain any information regarding the ordering and the number of points (pixels). This implies a certain scale and rotation invariance over the image regions in different images as long as layers L_i are invariant (similarly to [Tuzel 2006]).

Algorithm 3: Brownian descriptor algorithm

Data: Layers $\mathcal{L} = \{L_1, L_2, \ldots, L_n\}$, $L_i \in \mathbb{R}^p$
Result: Brownian descriptor $\mathcal{R}^2$
begin
Compute the Euclidean distance matrix (a_{kl})
$(a_{kl}) = (|L_k - L_l|_p)$
Let $A_{kl} = a_{kl} - \bar{a}_{k\cdot} - \bar{a}_{\cdot l} + \bar{a}_{\cdot\cdot}$, where
$\bar{a}_{k\cdot} = \frac{1}{n}\sum_{l=1}^{n} a_{kl}$ *(mean of the k-th row)*
$\bar{a}_{\cdot l} = \frac{1}{n}\sum_{k=1}^{n} a_{kl}$ *(mean of the l-th column)*
$\bar{a}_{\cdot\cdot} = \frac{1}{n^2}\sum_{k,l=1}^{n} a_{kl}$ *(mean of distances)*
Let $\mathcal{V}_n^2(\mathcal{L}) = \mathcal{V}_n^2(\mathcal{L}, \mathcal{L}) = \frac{1}{n^2}\sum_{k,l=1}^{n} A_{kl}^2$
$\mathcal{R}_{kl}^2 = \frac{A_{kl}^2}{\mathcal{V}_n^2(\mathcal{L})}$
end

Extraction Complexity: The computational and memory complexity of the proposed descriptor (*cf.* Algorithm 3) is O(n^2p), where n is the number of layers in the matrix and p is the number of points inside the region to be encoded. Notice that this extraction complexity is exactly the same as the classical covariance matrix.
Matching Complexity: For computing the distance between the Brownian descriptors (matrices), we use the euclidean distance between the vectorized matrices. *Vectorization* of a matrix is a linear transform which converts the matrix into a column vector.

8.4 Qualitative Results in Visual Tracking

Sample qualitative tracking results are given in figures 8.3 and 8.4. In each case, a tracker is initialized manually by selecting a target window (green rectangle). From this target window we extract either the covariance descriptor or the Brownian descriptor using state of the art feature mapping ($\phi = \left[x, y, I_{xy}, \nabla_{xy}^I, \theta_{xy}^I\right]$) [Tuzel 2006, Tuzel 2008, Yao 2011]. $\nabla_{xy}^I, \theta_{xy}^I$ represents gradient magnitude and orientation respectively. The candidate region in the next frame is selected by a dense scanning with 2×2 pixel step and by minimizing *geodesic distance* [Forstner 2003] between the target and the candidate matrix (Brownian or covariance). Thus we select the bounding box in a successive frame which has the least distance (in the descriptor space) w.r.t. the current target location (*i.e.* bounding box in the current frame). We follow the scheme of [Porikli 2006], in which the target description is updated by averaging the descriptor on a Riemannian manifold. We show the tracking results for both, covariance and Brownian descriptor, using data related to different applications.

Car tracking: This video shows a moving car in a scene. Figure 8.3 shows a comparison for the Brownian and the traditional covariance descriptor, for the case of visual tracking. In this Figure 8.3 we can observe a drift of both descriptors due to texture changes (shadows on the car modify the appearance). However, the results show that the Brownian descriptor is able to track the vehicle through the whole sequence while the covariance descriptor loses the object in the middle of the sequence.
Tennis player tracking [Brox 2010]: This video is of a tennis player, from a widely accepted motion segmentation dataset, which contains moving and stationary camera recordings. In Figure 8.4 we observe that the Browning descriptor keeps track of the object for a longer period of time than the traditional covariance descriptor.

The drift of the bounding boxes (in Figures 8.3 and 8.4) from the main object to the background is commonly referred to as the *template drift problem*, wherein a template correspond to the bounding box of the main object. This template drift problem occurs due to the accumulation of small errors while successfully matching the template in time. The prime cause of this drift error is the change in appearance of object while moving across frames [Kaneko 2002, Matthews 2004]. The results in Figures 8.3 and 8.4 demonstrate that we are able to track the template for longer time duration with the Brownian descriptor as opposed to the classical covariance matrix. This is due to the fact that matching in successive frames is more reliable with the Brownian descriptor, and the template position is better tracked w.r.t. time. Therefore we obtain a better drift immunity in our case as compared to the covariance-based tracking.

8.5 Quantitative Results on Person Re-identification datasets

For illustrating quantitative results, we select the person re-identification problem. This problem is a challenging case of appearance matching, and hence we believe that it is a good choice for evaluating our proposed Brownian descriptor.

We carry out experiments on three i-LIDS datasets[1]: i-LIDS [Zheng 2009], i-LIDS-MA and i-LIDS-AA [Bak 2011b]. These datasets have been extracted from the 2008 i-LIDS Multiple-Camera Tracking Scenario (MCTS) dataset with multiple non-overlapping camera views. The results are analyzed in terms of recognition rate, using the *cumulative matching characteristic* (CMC) curve [Gray 2007]. The CMC curve represents the expectation of finding the correct match in the top n *ranks*. Additionally, we report a quantitative scalar from the CMC curve obtained by the normalized area under CMC curve (nAUC), which is reported in brackets.

For *baseline* comparisons, we select the MRCG [Bak 2011a] descriptor. This descriptor is formed by a dense grid of covariance matrices which are averaged on the Riemannian manifold. Using the same appearance representation we replace the classical covariance descriptor with our Brownian descriptor. We refer to this modified descriptor as **MRCG+B**. For a fair comparison, we evaluate both descriptors without applying discriminative analysis proposed in [Bak 2011a].

[1]The Image Library for Intelligent Detection Systems (i-LIDS) is the UK government's benchmark for Video Analytics (VA) systems

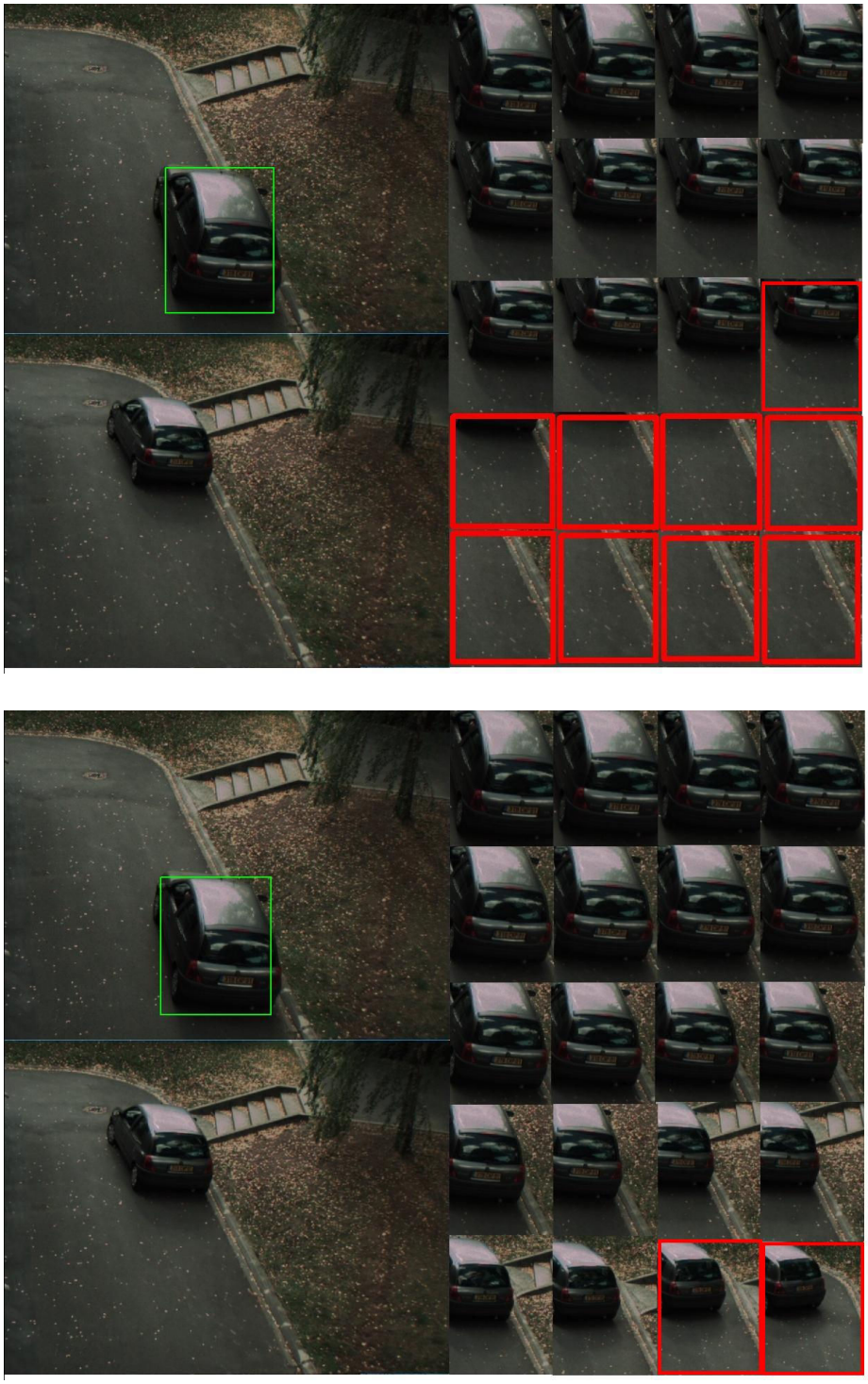

Figure 8.3: Template Tracking using Covariance. **Top** image shows tracking with the covariance based descriptor. **Bottom** image shows better drift immunity by using the proposed Brownian descriptor.

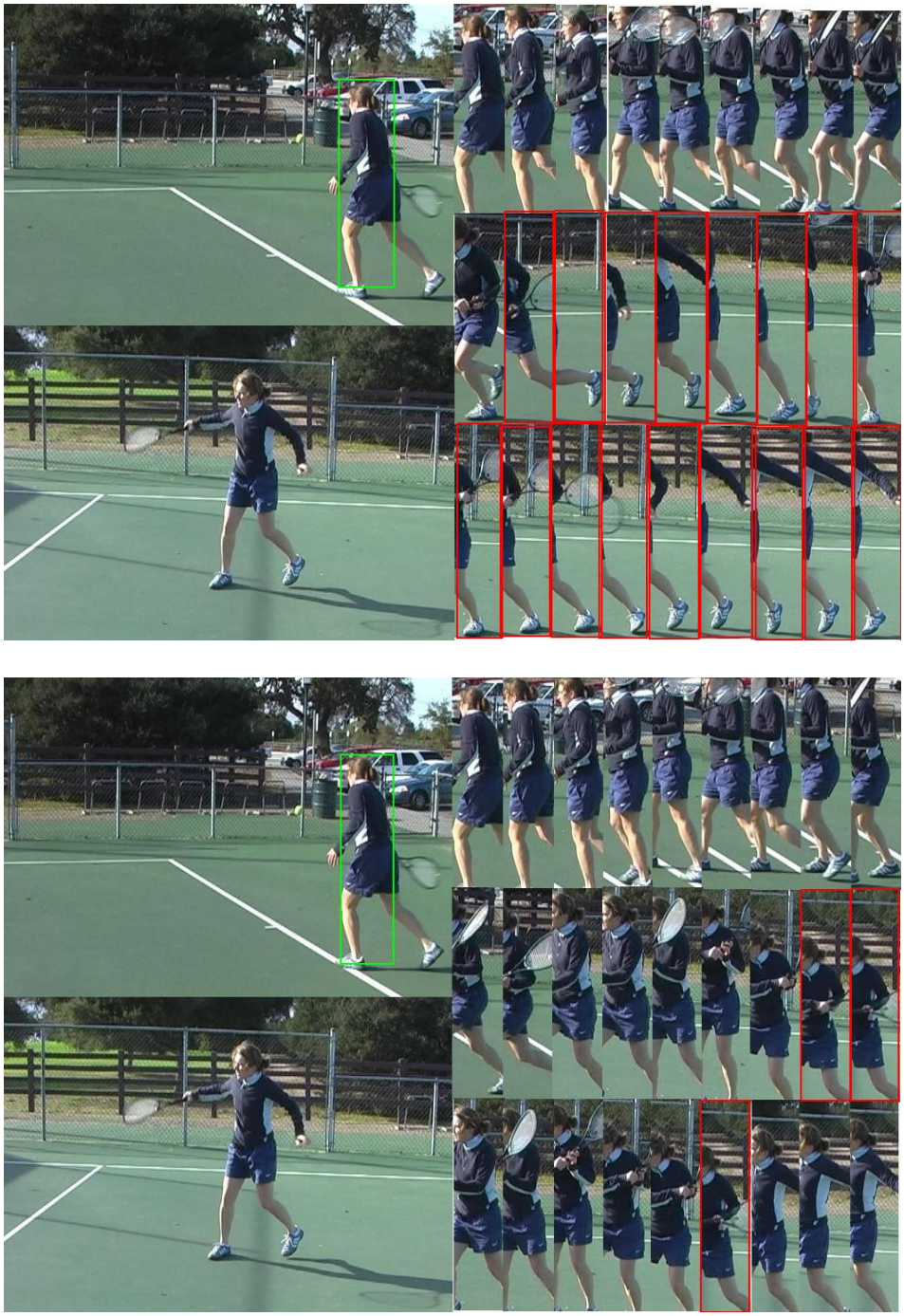

Figure 8.4: Template Tracking using Covariance. **Top** image shows tracking with the covariance based descriptor. **Bottom** image shows better drift immunity by using the proposed Brownian descriptor.

8.5.1 i-LIDS-MA

This dataset [Bak 2011b] consists of 40 individuals extracted from two non-overlapping camera views. For each individual a set of 46 images is given. Hence, we have in total $40 \times 2 \times 46 = 3680$ images. Figure 8.1 illustrates sample images from i-LIDS-MA.

For each pedestrian we create its human signatures using $N = 1, 3, 5, 10$ randomly selected images. Then, every signature is used as a query to the gallery set of signatures from different cameras. The procedures were repeated 10 times and the average CMC curves are displayed in Figure 8.5.

The results indicate that the larger are the number of frames, the better performance is achieved by both descriptors. Noticeably the Brownian descriptor outperforms the classical covariance matrix in all experiments consistently. This confirms the expected increase in performance of the new descriptor and shows that the Brownian descriptor is a richer meta-feature than the classical covariance for the appearance matching.

8.5.2 i-LIDS-AA

This dataset [Bak 2011b] contains 100 individuals automatically detected and tracked in two cameras. Cropped images are noisy, which makes the dataset more challenging (*e.g.* detected bounding boxes are not accurately centered around the people, only part of the people is detected due to occlusion). For minimizing misalignment issues, we employ *discriminatively trained deformable part models* (DPM, [Felzenszwalb 2010]), which slightly improves the detection accuracy.

The covariance descriptor as a positive definite and symmetric matrix is usually assumed to lie on a Riemannian manifold. Hence, computing distance between two matrices, we need to solve the generalized eigenvalues problem (*geodesic distance* [Forstner 2003]), which brings a computational burden. The geodesic distance can be also approximated by mapping matrices to a tangent plane [Tosato 2010] and then computing the L_1 norm. Finally we can also vectorize the descriptor matrix as a simple one dimensional vector and apply directly the L_1 or L_2 norm for computing distance between descriptor-matrices.

Figure 8.6 illustrates the performance w.r.t. the selected metric. The Brownian descriptor not only outperforms the original covariance in every case, but also achieves good performance even while using the L_1 or L_2 norm on vectorized descriptor matrices. This is an interesting prospect for further applications of this proposed Brownian descriptor to other vision areas. Until now, the covariance descriptor was often avoided for employing to the retrieval and recognition tasks due to the following limitations. First, the geodesic metric is computationally heavy. Second, learning on the manifold is difficult and usually produces models which are over-fitted to training data. But now we can utilize this Brownian descriptor in combination with common classifiers employed in an Euclidean space.

8.5.3 i-LIDS

This evaluation dataset [Zheng 2009] has been extracted automatically. It contains 476 images with 119 individuals registered by two different cameras. This dataset is very chal-

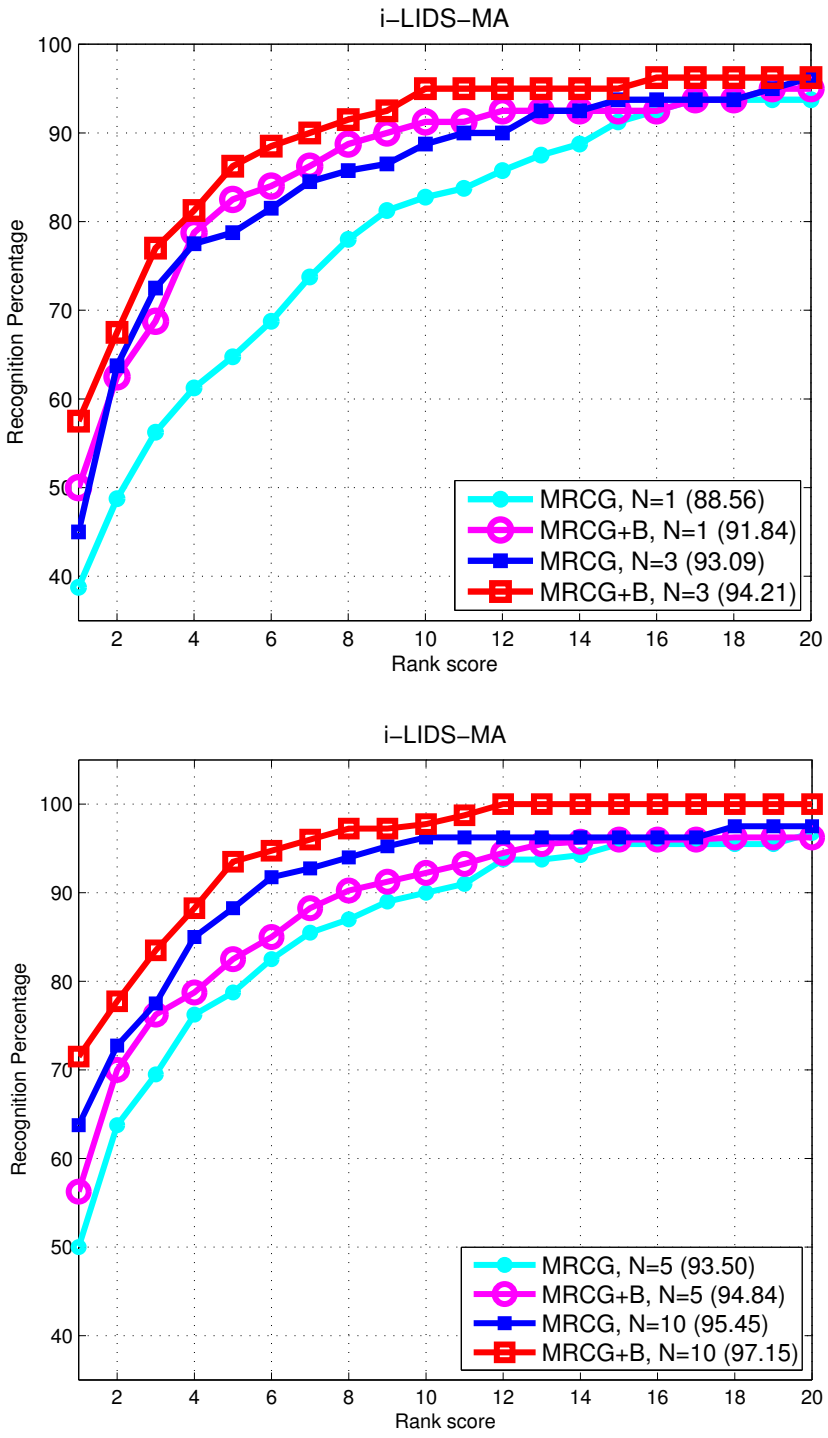

Figure 8.5: Performance comparison on i-LIDS-MA [Bak 2011b].Top figure corresponds to $N = 1, 3$, while bottom figure has $N = 5, 10$ (*cf.* section 8.5 for details).

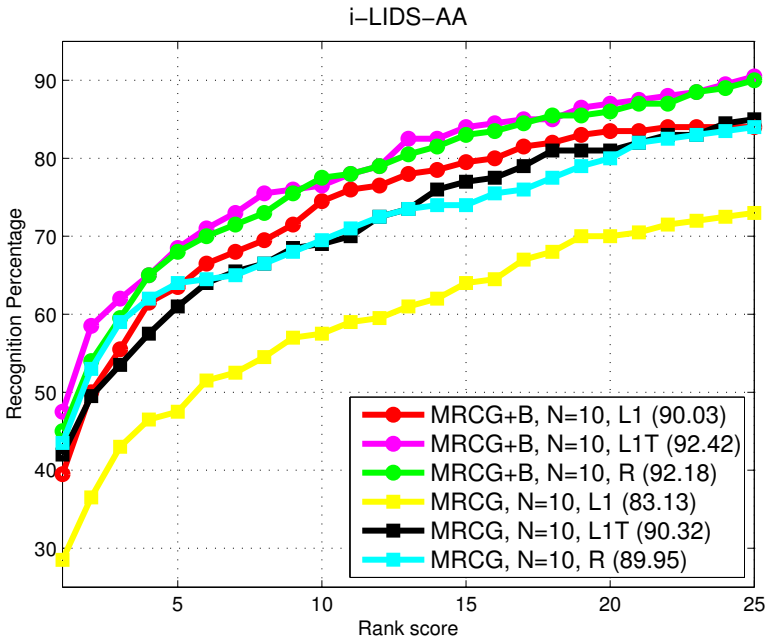

Figure 8.6: Performance comparison on i-LIDS-AA [Bak 2011b] using different metrics: $L1$ corresponds to the L_1 norm metric between vectorized Brownian descriptors, $L1T$ refers to the L_1 norm on a tangent plane, R - geodesic distance [Forstner 2003]

.

lenging due to many occlusions (often only the top part of the person is visible). We reproduce the same experimental settings as [Bak 2011a] and the results are shown in Figure 8.7. We can notice that our descriptor again outperforms the regular covariance descriptor. It clearly shows the advantage of our Brownian descriptor over the classical covariance descriptor.

8.5.4 Descriptor Efficiency

The above experiments show that the Brownian descriptor matrices need not be projected onto a manifold for expressing distances between them. This avoids intensive computational overhead. By simply replacing the classical covariance descriptor by the Brownian descriptor, on the same set of features, we achieve a significant computational speedup of **150x** over the classical covariance. Note that the computational complexity of extracting the Brownian descriptor and the classical covariance descriptor is the same.

8.6 Perspectives on Single Camera Multi-Object Tracking

This section provides an insight on the importance of having state-of-the-art appearance feature-descriptors to aid in obtaining good trajectories for multiple object tracking.

For a single camera tracking, it is of interest to see if sophisticated re-identification

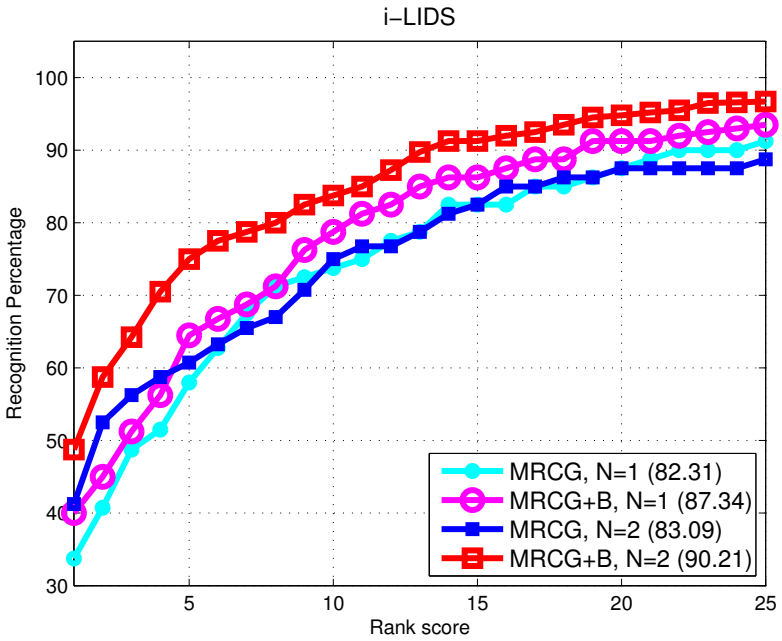

Figure 8.7: Performance comparison on i-LIDS [Zheng 2009].

features can bring improvements in solutions. The intuition is that the occlusion management for the proposed tracker in this thesis will be easier with these descriptors, as better descriptors will provide better clustering of bounding boxes (*cf.* section 6.2.6).

Comparing appearance descriptors for tracking

We take the same batch as in Figure 7.10 of the Towncenter video, and evaluate three advanced state-of-the-art descriptors employed in Re-Identification. Quantitative results are shown in Table 8.1. Features for the Covariance and Brownian descriptors are in sync with [Porikli 2006] ($\left[x, y, I_{xy}, \nabla^I_{xy}, \theta^I_{xy}\right]$).

Towncenter			
Appearance Features	MOTA	IDS	Detector
Covariance	63.13	115	[Benfold 2011]
Brownian	63.02	137	[Benfold 2011]
Color	**63.45**	**76**	[Benfold 2011]

Table 8.1: Quantitative Results for the Towncenter Video for a 390 frame batch. The first column identifies the appearance feature descriptors.

Observations

Simple color histogram performs better than other descriptors. For the other descriptors to work, a structure in intensity patterns is needed (*e.g.* textures). Hence uniform color clothing is detrimental for the covariance based descriptions. Moreover this failure also comes from the overlapping of the detection bounding boxes. Overlapped bounding boxes impair good feature extraction, and this in-turn results in a poor clustering for occlusion management of our tracker. This issue of poor feature extraction due to overlapped bounding boxes is indeed also with the color descriptors too. Hence a tracker should incorporate a reliability factor for the appearance descriptors based on the overlap ratio of the bounding boxes.

For a multiple camera scenario, covariance based appearance descriptors will help in matching different bounding boxes of same persons. However the problems of overlapping of bounding boxes, and intensity structure will remain a common issue. Hence in this scenario, similarly to above, a reliability factor based on percentage overlap of a bounding box w.r.t. the nearby boxes will have to be incorporated. Another way to resolve this (in cases of dense crowd) would be to incorporate part based features, and matching should be performed when there is no occlusion in parts location.

To conclude, it is important to use motion cues for resolving issues of overlapped bounding boxes and hence obtaining reliable trajectories. Appearance (Re-Identification) features seem to be of interest in matching appearances across different cameras, only when there are no spatial occlusions. In cases of persistent occlusions leading to overlapped bounding boxes, the motion cues will be of higher importance.

Depending upon the intensity structural information in the bounding boxes, appropriate appearance features can be selected. For example, in cases of color-homogeneous clothing, color based features *e.g.* histogram of colors will encode better description than structural information such as edges or gradients. However in the presence of sufficient texture information, covariance (color, gradient) based descriptors will be useful. A presence of intensity structure can be detected from the *structure tensor* matrix of an image I, $\nabla I \times \nabla I$, averaged over the detection area (*cf.* Appendix C for details on the structure tensor).

8.7 Conclusion

In this chapter we discussed the limitations of the classical covariance and a new statistical measure namely *distance correlation* [Szekely 2009]. Based on the distance correlation, we designed an appearance descriptor which we refer to as *Brownian descriptor* owing to its theoretical similarity with the Brownian motion. We extensively experimented with the proposed descriptor on the re-identification dataset, and demonstrated its usefulness. Finally we discussed on the applicability of sophisticated appearance descriptor to a single-camera-multi-object tracking, and from the experiments we observe that for a single camera multi-object tracker, it seems important to consider cues from overlapping ratio of bounding boxes to set relative importance for appearance and motion terms in a tracking-model. Furthermore, the development of appearance descriptors based on the presence of intensity structure inside bounding boxes would aid in better clustering of bounding boxes, and hence managing occlusions in a better way. For a textured region, it is reasonable to encode covariance of several texture based filter outputs, while for a texture-free region, simple color histogram based encoding should be sufficient.

This calls for setting adaptive factors based on the scene characteristics in our proposed tracker, wherein cues from overlapped ratio bounding boxes, presence of textural information have to be incorporated in future.

Concluding the thesis

CHAPTER 9

Conclusions and Future Work

Contents

9.1 Contributions 145
9.1.1 Video segmentation 145
9.1.2 Multiple object tracking 146
9.1.3 Distance Correlation for appearance matching 147
9.2 Future Perspectives 147
9.2.1 Short term future work 147
9.2.2 Long term future work 148

We conclude our work by pointing out the key contributions and their limitations. Finally, we discuss perspectives indicating interesting directions for future research in the realm of video segmentation and multiple object tracking.

9.1 Contributions

In this thesis we investigated key building blocks that form the core of a video understanding system, namely video segmentation and multiple object tracking. The contributions are outlined in the following sections:

9.1.1 Video segmentation

In this part of the thesis work, we joined both spatial and temporal aspect of a video into a single unified notion: *Fiber*. A Fiber is a set of trajectories spatially connected with a triangular mesh. Compared to the state-of-the-art, a fiber based segmentation presents advantages such as a spatio-temporal neighborhood accessor using mesh, long-term temporal correspondences for most pixels in a video. We evaluated this approach on the motion segmentation dataset by [Brox 2010], and also provided qualitative results w.r.t. [Grundmann 2010]. The results demonstrate the usefulness of dense long term motion incorporation by fibers.

We also demonstrated the utility of fibers by a simple video inpainting demo wherein only a few mouse clicks on a single frame were used to remove an unwanted object in a scene.

There is no underlying assumption about the camera motion, hence this proposed approach can be applied to both moving and static camera scenes.

Limitations

- Owing to the requirements of spatio-temporal color and motion coherency, this fiber based video segmentation is difficult to apply to videos having low frame rate (5-10 fps) and poor spatial resolution. A poor video frame rate impairs good correspondences measurement, while a poor spatial resolution prohibits computation of color and motion statistics.

- Trajectories comprising a fiber have same temporal span. For videos where points frequently get occluded or dis-occluded, it will be of interest to build fibers with trajectories of different temporal spans.

- Piecewise constant fiber-extension to a pixel by copying the closest fiber trajectory is not reliable in cases wherein the motion is varying rapidly in space, *e.g.* extending fibers from fingers of a person to the elbow during movements such as *punching*, *rotating arm*.

9.1.2 Multiple object tracking

Our proposed multi-object tracking computes trajectories in a unified manner incorporating local and global cues on a Conditional Random Field. A combination of message passing and of an iterated conditional modes variant is used to optimize this model. From a vision system's perspective, this approach has two important advantages. Firstly, the approach is computationally efficient and operates at a speed of 10 fps on HD videos of busy Towncenter street (single core). Secondly, the system can be stopped to adhere to time critical requirements, yet providing a high quality usable solution. We demonstrated the usefulness of our approach on two widely used videos, namely Towncenter and PETS S2L1. Our results outperform earlier recent works along with obtaining better computational efficiency.

Limitations

- The triplet factor to ensure smooth trajectories requires smooth or no camera motion for its reliable computation. More precisely the camera movement should not be fast and persistent jitter.

- The triplet factor is modulated by the height of the central bounding box. The bounding box height is an important cue for the depth of an object, and is a necessary factor for comparing apparent speed across videos. Hence this modulation helps in obtaining video invariancy for the triplet factors. This modulation makes an inherent assumption that the heights of persons are similar across the video. It is indeed a reasonable assumption for many videos. However this assumption is no longer valid in videos where the depth of the person changes significantly in a space of few frames. An example video for this is from the ETH-MS dataset. To mitigate this, we need to compute triplet factors by considering statistics on relative depth changes across frames.

9.1.3 Distance Correlation for appearance matching

We proposed a novel descriptor (Brownian descriptor) for appearance matching which utilizes a recently developed theory based on Brownian statistics to compute the covariance between random variables. This new measure is known as *distance correlation*, and is zero if and only if the random variables are independent, or the samples are identical; whereas the classical covariance computes only linear correlations.

For evaluations we considered one of the challenging areas of appearance matching namely Re-identification, and our proposed descriptor outperforms the descriptors based on classical covariance.

Limitation

Re-identification features are relevant for textured patches and are not suitable for homogeneous patches with a single color.

9.2 Future Perspectives

The solutions provided in this thesis open up many interesting future directions. In the following sections we provide some future perspectives. We categorize these perspectives into *short-term* and *long-term*.

9.2.1 Short term future work

We first summarize the short term perspectives in the following sections:

Video segmentation

- **Different time spans for different trajectories inside a fiber**: Trajectories for unextended fibers are of same temporal length as that of the fiber. Furthermore, the *long term* temporal assessment is also based on using trajectories which are of fixed length. For points appearing or disappearing in a scene, it is of interest to consider splitting of trajectories into shorter color coherent segments. This can be formulated as a variant of the standard *Dynamic Programming* algorithm for a *rod cutting problem*.

Multiple object tracking

- **Adaptive search for triplets**: In our proposed tracker we search for triplets comprising bounding boxes which are separated at symmetrical distances. Another way is to compute triplets in an adaptive manner based on scene characteristics. It is clear that multiple overlapped bounding boxes will impair good appearance features extraction. Hence in these situations ideally we should incorporate exhaustive (or close to exhaustive) set of triplets. On the other hand if the appearance features are discriminative, then we can reduce the exhaustiveness in the search for triplets.

- **Combining multiple object/part detectors**: Detectors often make mistakes leading to a lot of false positives and negatives. Also in cases of high occlusion density and multiple overlapping people, a person detector is bound to fail. In these scenarios, a head or an upper body detector can provide good information to localize a person. A viable tracking model would be to incorporate information from multiple detectors.

Distance Correlation for appearance matching

- **Investigate the relationship to kernel methods**: Distance correlation keeps non-linear relationship among variables. Hence it makes sense to have a comparative study of non-linear mapping kernels and distance correlation. An example of a non linear mapping would be a kernel-PCA.

9.2.2 Long term future work

The following sections will elaborate on long term perspectives on video segmentation and multi-object tracking.

Video segmentation

- **Interpolating flow from sparse fibers**: An interesting feature of the fiber based video segmentation is the computational efficiency in computing trajectories. In the current work we copy the nearest fiber-trajectory to a pixel. However if the initial set of un-extended fibers is sparse, this way of copying the nearest trajectory based on the assumption of piece-wise constant flow in space, will not be reliable (*cf.* section 4.5, Figure 4.14).

  A viable approach for computing continuous and long term flow at a particular pixel would be to interpolate trajectories from nearby fibers. The challenges here lie in knowing which fibers to choose for interpolation at a point/region, and keeping track of points which repetitively appear and disappear because of occlusions. To this end it will be of interest to consider object proposal algorithms such as [Ziming 2014], [Endres 2013] to aid in knowing relevant fibers for extension at a particular pixel.

- **Action Recognition and Localization**: Very recently there has been an increase in exploration towards utilizing dense spatio-temporal video segmentation for application to activity analysis and action recognition and localization. Some of the notable approaches are [Jain 2014], [Trichet 2014], [Trichet 2013b].

  Fibers facilitate the expression of long term movement details for all pixels, and hence are of particular interest for applications such as action recognition. Fine motion details are preserved near interest points which in turn will facilitate the design of good descriptors for the long term motion. On a higher level, one has to design the co-occurrence of motion of object/parts to infer an action. The challenges lie in handling different viewpoints, scales.

Multiple Object Tracking

- **Automatic parameter estimation**: We have two degrees of freedom in our tracking model. The parameters modulate the weights for triplet factors, point tracks and appearance factors. The relative importance of parameters can be adjusted depending on the scene characteristics. For example, in a scene with high crowd density (many overlapping bounding boxes), it is intuitive to set a very high weight on triplet factors as compared to others. Other scene characteristic of interest is the discriminative power of appearance features (*i.e.* textural regions), which will enable setting relative importance of triplets and appearance factor. This can be formulated as a graphical model for which parameter estimation can be performed using techniques such as *Expectation-Maximization*.

APPENDIX A

Computing geodesics

Contents

A.1 Dijkstra and Fast Marching Method 151

A.1 Dijkstra and Fast Marching Method

Dijkstra and Fast Marching Methods (FMM) are algorithms for solving the *shortest path problem* on a graph. While in Dijkstra's algorithm the movement from a node to another is restricted to follow graph edges, FMM allows for a continuous movement as shown in Figure A.1. The Dijkstra algorithm suffers from a strong metrization problem, and for an image-grid with 4-system neighborhood, it actually computes the l^1 distance.

The Fast Marching algorithm replace the graph update by a local resolution of the Eikonal equation. This reduces significantly the grid bias, and can be shown to converge to the underlying geodesic distance when the grid step size tends to zero.

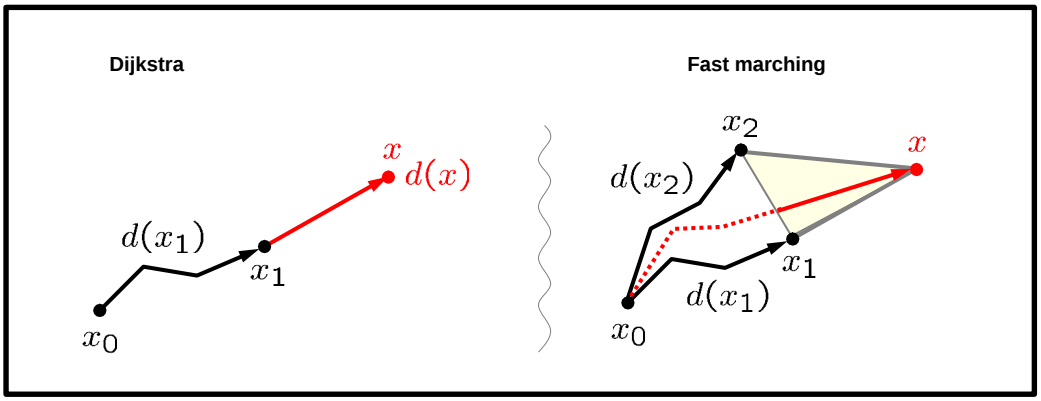

Figure A.1: Difference between Dijkstra and Fast Marching Method. **Image Source** : http://tosca.cs.technion.ac.il/book/course_siam10.html

In terms of implementation, the graph update steps are the only difference between Dijkstra and FMM (*cf.* Figures A.2 & A.3).

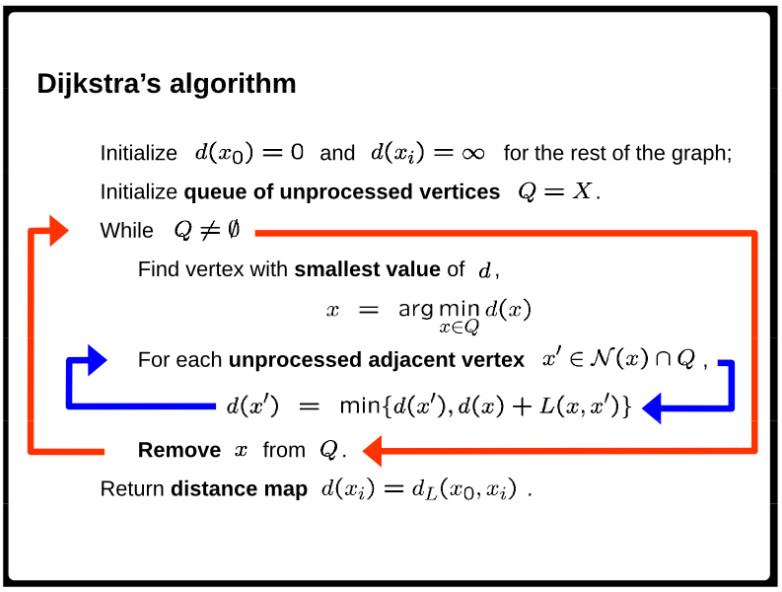

Figure A.2: Dijkstra Algorithm. x_0 is the source and hence $d(x_0)$ is zero; Set X represents all nodes on a grid. **Image Source** : http://tosca.cs.technion.ac.il/book/course_siam10.html.

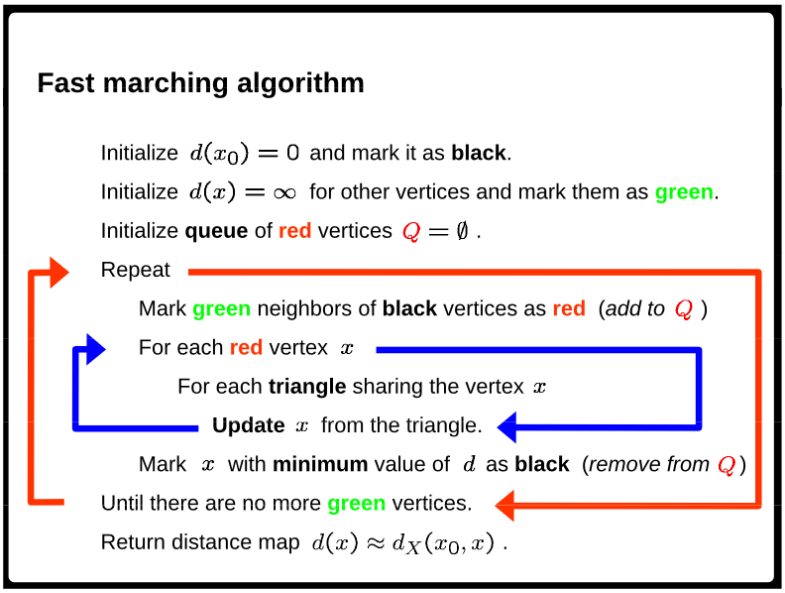

Figure A.3: Algorithm Fast Marching Method. x_0 is the source and hence $d(x_0)$ is zero; Set X represents all nodes on a grid. Notice the difference in the update step w.r.t. Dijkstra Algorithm in Figure A.2. **Image Source** : http://tosca.cs.technion.ac.il/book/course_siam10.html.

APPENDIX B

Markov Random Fields

Contents

B.1 Markov Random Fields: Connect locally and optimize globally 153
B.2 Optimization for MRF model 154

B.1 Markov Random Fields: Connect locally and optimize globally

Markov (and conditional) random fields have successfully been employed in an impressive variety of applications, including computer vision. We review the basics of MRF in this section. A description of basic terminology is described first, followed by some remarks on designing a model and an optimization. For an introductory treatment of graphical models and MRF, the reader is encouraged to refer to Chapter 8 of Christopher Bishop's Pattern Recognition and Machine Learning book [1].

A stochastic process is a collection of random variables $\{X_t : t \in \tau\}$ indexed by some subset τ of the real line $\mathbb{R}$. The elements of τ are often interpreted as times, in which case X_t represents the state at time t of the random process in consideration. The term *random field* refers to a generalization of the notion of a stochastic process : a random field $\{X_s : s \in G\}$ is still a collection of random variables, but now the index set G need not be a subset of $\mathbb{R}$. For example, G could be a subset of the plane $\mathbb{R}^2$; such random fields are naturally of interest in certain image processing problems [Chang 2007].

Markov Random Fields (MRF) : Given a graph G having set of nodes $\{1 \cdots n\}$ and a given neighborhood $N(t) = \{neighbors\ of\ node\ t\}$, the collection of random variables $(X_1 \cdots X_n)$ is a MRF on G iff

$$P\{X_t = x_t | X_s = x_s \text{ for } s \neq t\} = P\{X_t = x_t | X_s = x_s \text{ for } s \in N(t)\} \tag{B.1}$$

for all nodes $t \in \{1 \cdots n\}$.

Gibbs Random Fields (GRF) : A set of random variables $\{X_1 \cdots X_n\}$ is said to be a GRF on G if and only if its configurations obey a Gibbs distribution. A Gibbs distribution is defined as

$$P(X) = \frac{1}{Z} exp\{U(X)/T\}, \tag{B.2}$$

[1] http://research.microsoft.com/en-us/um/people/cmbishop/prml/

where Z is a normalizing constant called the *partition function*, T is a constant called the *temperature* and U is the energy function. The energy $U(X)$ is the sum of neighborhood potentials over all possible neighborhoods.
Hammersley-Clifford Theorem: Given X, G and P as a probability distribution over X such that P(x) > 0 for any realization x of X, then the following conditions are equivalent [Hammersley 1971] :

- $P(X)$ is a GRF which factorizes according to maximal cliques in G.
- If $N(X_i)$ is the set of neighbors of X_i in G, then the *local markov property* holds: $P(X_i|X \backslash X_i) = P(X_i|N(X_i))$.
- If A, B and S are three disjoint subsets of X, and S separates A from B, then the *global markov* property holds, *i.e.* $P(A|B \cup S) = P(A|S)$.

Remarks

- The *Hammersley-Clifford Theorem* entails that the following assumptions concerning a given set of variables are exactly equivalent [2]:
  - The random vector can be modeled by a Gibbs distribution.
  - The random vector satisfies the local or the global Markov property.

  Therefore, using a Markov network to model a certain distribution is justified exactly when any one of the three conditions specified in the theorem is satisfied by the given random variables.
- The positivity condition on the probability distribution is *strictly necessary* in order for the theorem to hold.

B.2 Optimization for MRF model

Optimization for an MRF model corresponds to finding a configuration ($x \in X$) such that the joint probability is maximized. There are a host of optimization techniques which can be applied to an MRF. The algorithms are sometimes referred to as *inference algorithms* in the literature.

For specific graphs such as trees (graphs with no loops), exact inference can be computed in polynomial time using algorithms such as *Viterbi*, or *message passing*. Otherwise for general graphs, popular techniques are *Graph cuts* (exact for binary labels, and approximate otherwise), *alpha expansion* (approximate), *TRW-S* (approximate). A detailed survey of inference methods applicable to a wide range of problems can be seen in [Kappes 2013a].

[2] http://researchers.lille.inria.fr/~freno/MOCAD_PGM_2011-2012.html

APPENDIX C

Notes on Structure Tensor

Contents

C.1 Structure Tensor 155
C.2 Properties of $\mathcal{T}$ 155
C.3 Geometric interpretation of eigenvalues of $\mathcal{T}$ 156

This short appendix outlines the structure tensor. A detailed overview about the structure tensor and the state of the art on *low level image analysis* can be found in [Köthe 2007]. The details below are in sync with the short course on structure tensor by Dr Bastian Goldlocke, HCI, Germany [1].

C.1 Structure Tensor

The structure tensor $\mathcal{T}$ of an image $\mathcal{I}$ is a symmetric 2x2 matrix as following:

$$\mathcal{T} = G_\tau * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{C.1}$$

where

$$I_x = \partial_x I_\sigma,\ I_y = \partial_y I_\sigma, I_\sigma = G_\sigma * I \tag{C.2}$$

The parameter $\sigma \geqslant 0$ is referred as inner scale, whereas $\tau \geqslant 0$ is called the outer scale. G_σ is the Gaussian kernel with standard deviation of σ. The operator "*" indicates convolution operation in above equations.

C.2 Properties of $\mathcal{T}$

- $\mathcal{T}$ characterizes the joint distribution of the two partial derivatives (I_x and I_y). Assuming that the mean of the partial derivatives are zero, *i.e.* $E[I_x] = E[I_y] = 0$, $\mathcal{T}$ can also be expressed as *covariance matrix* of I_x and I_y as following:

$$\mathcal{T} = \begin{bmatrix} E[I_x^2] & E[I_x I_y] \\ E[I_x I_y] & E[I_y^2] \end{bmatrix} \tag{C.3}$$

[1] http://hci.iwr.uni-heidelberg.de/Staff/bgoldlue/cuda_ss_2012.php

- $\mathcal{T}$ is positive definite and hence has two non-negative eigenvalues λ_1, λ_2.
- The larger eigenvalue indicates the strength of the local image edge, and the corresponding eigenvector is perpendicular to the edge.
- The popular *Harris cornerness* measure [Harris 1988] is based on the relationship between the eigenvalues of $\mathcal{T}$, and is given by:

$$det(\mathcal{T}) - 0.04 * trace^2(\mathcal{T}) = \lambda_1\lambda_2 - 0.04(\lambda_1 + \lambda_2)^2 \tag{C.4}$$

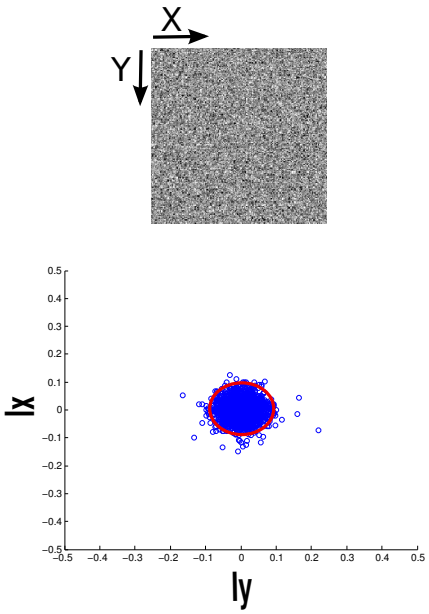

Figure C.1: Top: A noisy image with no structure such as edges and corners. Bottom: The distribution of I_x and I_y. Both eigenvalues of $\mathcal{T}$ are of very small magnitude, and hence the ellipse is of small size.

C.3 Geometric interpretation of eigenvalues of $\mathcal{T}$

As the $\mathcal{T}$ is positive definite, it is diagonalizable and its eigenvectors correspond to an orthogonal co-ordinate system. In this system the eigenvalues correspond to the length of an ellipsoid fitted to the distribution of image gradient.

- For a homogeneous intensity image (*cf.* Figure C.1), the magnitudes of both eigenvalues are small.
- Figure C.2 shows the fitted ellipsoid for an image with a distinct edge. In this case one eigenvalue is of large magnitude, while the second is very small in magnitude.
- Figure C.3 shows an image which has one corner, and in this case, both eigenvalues are of large magnitude indicated by the larger fitted ellipsoid in comparison with the earlier cases.

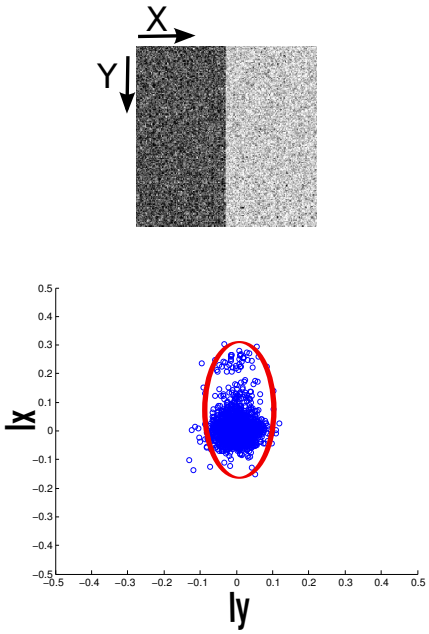

Figure C.2: Top: An image with an **edge**. Bottom: The distribution of I_x and I_y. One eigenvalue is of very large magnitude stretching one axis of ellipsoid along the corresponding eigenvector.

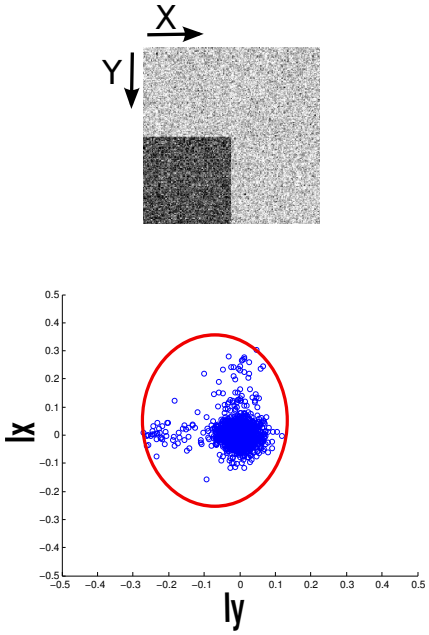

Figure C.3: Top: An image with a **corner**. Bottom: The distribution of I_x and I_y. Both eigenvalues are of very large value and hence the fitted ellipsoid is of larger size than in Figures C.1 and C.2

Bibliography

[Achanta 2012] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua and Sabine Süsstrunk. *SLIC superpixels compared to state-of-the-art superpixel methods.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 11, pages 2274–82, November 2012. (Cited on pages vii and 23.)

[Aggarwal 2011] J K Aggarwal and M S Ryoo. *Human Activity Analysis : A Review*. ACM Computing Surveys, vol. 43, no. 3, pages 16:1–16:43, 2011. (Cited on pages 37 and 39.)

[Alexe 2012] Bogdan Alexe, Thomas Deselaers and Vittorio Ferrari. *Measuring the objectness of image windows*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 11, pages 2189–202, November 2012. (Cited on page 25.)

[Andres 2012] Bjoern Andres, Jorg H. Kappes, Ullrich Kothe and Fred A. Hamprecht. *The Lazy Flipper: Efficient Depth-limited Exhaustive Search in Discrete Graphical Models*. In European Conference on Computer Vision, 2012. (Cited on pages 104, 109 and 120.)

[Andrieu 2003] Christophe Andrieu, N De Freitas, A Doucet and MI Jordan. *An introduction to MCMC for machine learning*. Machine learning, pages 5–43, 2003. (Cited on page 87.)

[Andriyenko 2012] A. Andriyenko, K. Schindler and S. Roth. *Discrete-continuous optimization for multi-target tracking*. In IEEE Conference on Computer Vision and Pattern Recognition, 2012. (Cited on pages 88, 91, 94 and 114.)

[Apostoloff 2006] Nicholas Apostoloff and Andrew Fitzgibbon. *Automatic video segmentation using spatiotemporal T-junctions*. In British Machine Vision Conference, 2006. (Cited on page 24.)

[Ayer 1995] Serge Ayer and HS Sawhney. *Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding*. In International Conference on Computer Vision, 1995. (Cited on page 36.)

[Bak 2011a] Slawomir Bak, Etiene Corvee, Francois Bremond and Monique Thonnat. *Multiple-shot Human Re-Identification by Mean Riemannian Covariance Grid*. In Conference on Advanced Video and Signal-Based Surveillance, pages 179–184, 2011. (Cited on pages 133 and 138.)

[Bak 2011b] Slawomir Bak, Etienne Corvee, Francois Bremond and Monique Thonnat. *Boosted human re-identification using Riemannian manifolds*. Image and Vision Computing, vol. 30, no. 6-7, pages 443–452, June 2011. (Cited on pages xv, xvi, 133, 136, 137 and 138.)

[Baker 2010] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black and Richard Szeliski. *A Database and Evaluation Methodology for Optical Flow*. International Journal of Computer Vision, vol. 92, no. 1, pages 1–31, November 2010. (Cited on page 72.)

[Bay 2006] Herbert Bay, Tinne Tuytelaars and Luc Van Gool. *SURF: Speeded up robust features*. In European Conference on Computer Vision, 2006. (Cited on page 128.)

[Benfold 2011] Ben Benfold and Ian Reid. *Stable multi-target tracking in real-time surveillance video*. In IEEE Conference on Computer Vision and Pattern Recognition, June 2011. (Cited on pages xvii, 85, 87, 91, 108, 111, 117, 118, 121, 126 and 139.)

[Berclaz 2011] J. Berclaz, F. Fleuret, E. Türetken and P. Fua. *Multiple Object Tracking using K-Shortest Paths Optimization*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 9, pages 1806–1819, 2011. (Cited on pages xiii, 89, 90, 116 and 117.)

[Bernardin 2008] Keni Bernardin and Rainer Stiefelhagen. *Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics*. EURASIP Journal of Image and Video Processing, vol. 2008, pages 1:1–1:10, 2008. (Cited on page 114.)

[Besag 1986] Julian Besag and Julian Besag. *On the statistical analysis of dirty pictures*. Journal of the Royal Statistical Society B, pages 48–259, 1986. (Cited on pages 104 and 120.)

[Boykov 2001] Yuri Boykov, Olga Veksler and Ramin Zabih. *Fast approximate energy minimization via graph cuts*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 11, pages 1–18, 2001. (Cited on page 88.)

[Boykov 2006] Yuri Boykov and Gareth Funka-Lea. *Graph Cuts and Efficient N-D Image Segmentation*. International Journal of Computer Vision, vol. 70, no. 2, pages 109–131, 2006. (Cited on pages 21 and 47.)

[Breitenstein 2009] MD Breitenstein and Fabian Reichlin. *Robust tracking-by-detection using a detector confidence particle filter*. In International Conference on Computer Vision, 2009. (Cited on page 81.)

[Brendel 2011] William Brendel, Mohamed Amer and Sinisa Todorovic. *Multiobject tracking as maximum weight independent set*. IEEE Conference on Computer Vision and Pattern Recognition, 2011. (Cited on pages 83, 84 and 85.)

[Brostow 2006] G.J. Brostow and R. Cipolla. *Unsupervised bayesian detection of independent motion in crowds*. In IEEE Conference on Computer Vision and Pattern Recognition, volume 1, pages 594–601. Ieee, 2006. (Cited on page 31.)

[Brox 2010] Thomas Brox and Jitendra Malik. *Object segmentation by long term analysis of point trajectories*. In European Conference on Computer Vision. Springer, 2010.

(Cited on pages vii, viii, x, xi, xvii, 10, 13, 31, 32, 33, 38, 52, 57, 58, 63, 64, 65, 66, 67, 70, 71, 74, 133 and 145.)

[Brox 2011] Thomas Brox and Jitendra Malik. *Large displacement optical flow: descriptor matching in variational motion estimation.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 3, pages 500–13, March 2011. (Cited on pages 31 and 108.)

[Butler 2012] Daniel J Butler, Jonas Wulff, Garrett B Stanley and Michael J Black. *A naturalistic open source movie for optical flow evaluation*. In European Conference on Computer Vision, 2012. (Cited on pages xii, 72, 73 and 112.)

[Butt 2013] AA Butt and RT Collins. *Multi-target tracking by Lagrangian relaxation to min-cost network flow*. In IEEE Conference on Computer Vision and Pattern Recognition, 2013. (Cited on pages xiii, 85, 86, 91 and 126.)

[Chang 2007] J. Chang. Stochastic Processes. 2007. (Cited on page 153.)

[Chau 2012] Duc Phu Chau. *Dynamic and robust object tracking for activity recognition*. PhD thesis, University of Nice, Sophia Antipolis, 2012. (Cited on page 82.)

[Choi 2010] Wongun Choi and Silvio Savarese. *Multiple target tracking in world coordinate with single, minimally calibrated camera*. In European Conference on Computer Vision, pages 553–567, 2010. (Cited on page 87.)

[Collins 2012] R. T. Collins. *Multitarget data association with higher-order motion models*. In IEEE Conference on Computer Vision and Pattern Recognition, June 2012. (Cited on pages 85, 91 and 104.)

[Cremers 2004] Daniel Cremers and Stefano Soatto. *Motion Competition: A Variational Approach to Piecewise Parametric Motion Segmentation*. International Journal of Computer Vision, vol. 62, no. 3, pages 249–265, November 2004. (Cited on pages 36 and 37.)

[Dalal 2005] N. Dalal and B. Triggs. *Histograms of Oriented Gradients for Human Detection*. In Conference on IEEE Conference on Computer Vision and Pattern Recognition, volume 1, pages 886–893. Ieee, 2005. (Cited on pages 108 and 128.)

[Deriche 1987] R Deriche. *Using Canny's criteria to derive a recursively implemented optimal edge detector*. International Journal of Computer Vision, vol. 1, no. 2, pages 167–187, 1987. (Cited on page 10.)

[Dijkstra 1959] EW Dijkstra. *A note on two problems in connexion with graphs*. Numerische Mathematik, no. 1 959, pages 269–271, 1959. (Cited on page 52.)

[Dubout 2012] Charles Dubout and F Fleuret. *Exact acceleration of linear object detectors*. In European Conference on Computer Vision, 2012. (Cited on page 108.)

[Ellis 2009] A. Ellis, A. Shahrokni and J.M. Ferryman. *PETS2009 and Winter-PETS 2009 results : A combined evaluation*. In PETS Workshop, 2009. (Cited on page 111.)

[Elqursh 2012] Ali Elqursh and Ahmed Elgammal. *Online Moving Camera Background Subtraction*. In European Conference on Computer Vision, volume 7577. Springer, 2012. (Cited on page 33.)

[Endres 2010] Ian Endres and Derek Hoiem. *Category Independent Object Proposals*. In European Conference on Computer Vision, 2010. (Cited on pages viii, 25, 26, 27 and 28.)

[Endres 2013] Ian Endres and Derek Hoiem. *Category-independent object proposals with diverse ranking*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 2, pages 222–34, February 2013. (Cited on pages viii, 25, 26 and 148.)

[Felzenszwalb 2004] P.F. Felzenszwalb and D.P. Huttenlocher. *Efficient graph-based image segmentation*. International Journal of Computer Vision, vol. 59, no. 2, pages 167–181, 2004. (Cited on pages vii, 22, 23, 28, 29, 30 and 34.)

[Felzenszwalb 2010] Pedro F Felzenszwalb, Ross B Girshick, David McAllester and Deva Ramanan. *Object detection with discriminatively trained part-based models*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 9, pages 1627–1645, September 2010. (Cited on pages 108, 118, 119 and 136.)

[Fischler 1981] MA Fischler and RC Bolles. *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*. Communications of the ACM, vol. 24, no. 6, 1981. (Cited on page 83.)

[Forstner 2003] Wolfgang Forstner and Boudewijn Moonen. *A metric for covariance matrices*. Geodesy-The Challenge of the third Millennium, pages 113–128, 2003. (Cited on pages xvi, 132, 136 and 138.)

[Fradet 2009] Matthieu Fradet, Philippe Robert and Patrick Pérez. *Clustering Point Trajectories with Various Life-Spans*. In Conference for Visual Media Production. Ieee, November 2009. (Cited on page 31.)

[Fragkiadaki 2011] Katerina Fragkiadaki and Jianbo Shi. *Detection free tracking: Exploiting motion and topology for segmenting and tracking under entanglement*. In IEEE Conference on Computer Vision and Pattern Recognition, 2011. (Cited on pages 31, 57 and 90.)

[Galasso 2012] Fabio Galasso, Roberto Cipolla and Bernt Schiele. *Video Segmentation with Superpixels*. In Asian Conference of Computer Vision, 2012. (Cited on pages 69 and 72.)

[Gauvrit 1997] H. Gauvrit and J. P L E Cadre. *A formulation of multitarget tracking as an incomplete data problem*. IEEE Transactions on Aerospace and Electronic Systems, vol. 33, no. 4, pages 1242–1257, 1997. (Cited on page 87.)

[Ge 2008] W. Ge and R.T. Collins. *Multi-target Data Association by Tracklets with Unsupervised Parameter Estimation*. In Procedings of the British Machine Vision Conference 2008, pages 93.1–93.10, 2008. (Cited on page 87.)

[Gelgon 2000] Marc Gelgon and Patrick Bouthemy. *A region-level motion-based graph representation and labeling for tracking a spatial image partition*. Pattern Recognition, vol. 33, pages 725–740, 2000. (Cited on page 36.)

[Gelgon 2005] Marc Gelgon, Patrick Bouthemy and Jean-pierre Le Cadre. *Recovery of the trajectories of multiple moving objects in an image sequence with a PMHT approach*. Image and Vision Computing, vol. 23, no. 1, pages 19–31, 2005. (Cited on pages 36 and 87.)

[Glover 1997] Fred Glover, Manuel Laguna and Rafael MartÃ. *Tabu Search*, 1997. (Cited on page 83.)

[Granados 2012] Miguel Granados, KI Kim and James Tompkin. *Background inpainting for videos with dynamic objects and a free-moving camera*. In European Conference on Computer Vision, 2012. (Cited on page 76.)

[Gray 2007] Doug Gray, S Brennan and H Tao. *Evaluating appearance models for recognition, reacquisition, and tracking*. In PETS Workshop, 2007. (Cited on page 133.)

[Grundmann 2010] Matthias Grundmann, V. Kwatra, Mei Han and Irfan Essa. *Efficient hierarchical graph-based video segmentation*. In IEEE Conference on Computer Vision and Pattern Recognition, 2010. (Cited on pages viii, x, xii, 23, 28, 30, 34, 35, 38, 62, 63, 69, 72 and 145.)

[Hammersley 1971] JM Hammersley and P Clifford. *Markov fields on finite graphs and lattices*. Rapport technique, 1971. (Cited on page 154.)

[Harris 1988] Chris Harris and Mike Step. *A combined corner and edge detector*. In Alvey Vision Conference, 1988. (Cited on page 156.)

[Heili 2013] A. Heili and JM Odobez. *Parameter estimation and contextual adaptation for a multi-object tracking CRF model*. In IEEE Workshop on Performance Evaluation of Tracking and Surveillance (PETS), January 2013. (Cited on page 88.)

[Heili 2014a] Alexandre Heili. *Human Tracking and Pose Estimation in Open Spaces*. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2014. (Cited on page 89.)

[Heili 2014b] Alexandre Heili, A Lopez-Mendez and J Odobez. *Exploiting Long-Term Connectivity and Visual Motion in CRF-based Multi-Person Tracking*. IEEE Transactions in Image Processing, vol. 23, no. 7, pages 3040–3056, 2014. (Cited on pages xvii, 88, 91, 116, 117, 118 and 126.)

[Hirzer 2011] Martin Hirzer, Csaba Beleznai, PM Roth and Horst Bischof. *Person re-identification by descriptive and discriminative classification*. In Scandinivian Conference on Image Analysis, pages 91–102, 2011. (Cited on page 128.)

[Hoiem 2007a] Derek Hoiem, Alexei A. Efros and Martial Hebert. *Recovering Surface Layout from an Image*. International Journal of Computer Vision, vol. 75, no. 1, pages 151–172, February 2007. (Cited on page 25.)

[Hoiem 2007b] Derek Hoiem, Andrew N. Stein, Alexei A. Efros and Martial Hebert. *Recovering Occlusion Boundaries from a Single Image*. In International Conference on Computer Vision, 2007. (Cited on page 25.)

[Horn 1981] Berthold K.P. Horn and Brian G. Schunck. *Determining optical flow*. Artificial Intelligence, vol. 17, no. 1-3, pages 185–203, August 1981. (Cited on page 6.)

[Jain 2014] M. Jain, J. C. van Gemert, H. Jégou, P. Bouthemy and C. G. M. Snoek. *Action Localization by Tubelets from Motion*. In IEEE Conference on Computer Vision and Pattern Recognition, 2014. (Cited on pages 35 and 148.)

[Jiang 2007] Hao Jiang, Sidney Fels and James J Little. *A Linear Programming Approach for Multiple Object Tracking*. In IEEE Conference on Computer Vision and Pattern Recognition, 2007. (Cited on page 84.)

[Kaneko 2002] T Kaneko and O Hori. *Template update criterion for template matching of image sequences*. International Conference on Pattern Recognition, pages 1–5, 2002. (Cited on page 133.)

[Kappes 2013a] Jorg H. Kappes, Bjoern Andres, Fred. Hamprecht, Christoph Schnorr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard X. Kausler, Jan Lellmann, Nikos Komodakis and Carsten Rother. *A Comparative Study of Modern Inference Techniques for Discrete Energy Minimization Problems*. In IEEE Conference on Computer Vision and Pattern Recognition, 2013. (Cited on pages 109 and 154.)

[Kappes 2013b] Jorg Hendrik Kappes, Markus Speth, Gerhard Reinelt and Christoph Schnorr. *Towards Efficient and Exact MAP-Inference for Large Scale Discrete Computer Vision Problems via Combinatorial Optimization*. In IEEE Conference on Computer Vision and Pattern Recognition, June 2013. (Cited on pages 90 and 104.)

[Khan 2005] Zia Khan, Tucker Balch and Frank Dellaert. *MCMC-based particle filtering for tracking a variable number of interacting targets.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 11, pages 1805–19, November 2005. (Cited on page 87.)

[Kolmogorov 2006] Vladimir Kolmogorov. *Convergent tree-reweighted message passing for energy minimization.* IEEE Transactions on Pattern Analysis and Machine

Intelligence, vol. 28, no. 10, pages 1568–1583, 2006. (Cited on pages 88, 103 and 109.)

[Komodakis 2007] Nikos Komodakis and Georgios Tziritas. *Approximate labeling via graph cuts based on linear programming.* IEEE transactions on pattern analysis and machine intelligence, vol. 29, no. 8, pages 1436–53, August 2007. (Cited on page 27.)

[Köthe 1995] Ullrich Köthe. *Primary image segmentation*. In Deutsche Arbeitsgemeinschaft fur Mustererkennung, 1995. (Cited on pages 21, 34 and 41.)

[Köthe 2007] Ullrich Köthe. *Reliable low-level image analysis*. PhD thesis, University of Hamburg, 2007. (Cited on page 155.)

[Lee 2011] Yong Jae Lee, Jaechul Kim and Kristen Grauman. *Key-Segments for Video Object Segmentation*. In International Conference on Computer Vision, 2011. (Cited on pages 27, 28 and 33.)

[Lezama 2011] J. Lezama, K. Alahari, Josef Sivic and I. Laptev. *Track to the Future: Spatio-temporal Video Segmentation with Long-range Motion Cues*. In IEEE Conference on Computer Vision and Pattern Recognition, 2011. (Cited on pages viii, 34 and 35.)

[Li 2009] Yuan. Li, Chang. Huang and R. Nevatia. *Learning to associate: HybridBoosted multi-target tracker for crowded scene*. In IEEE Conference on Computer Vision and Pattern Recognition, 2009. (Cited on page 114.)

[Liu 2008] Ce Liu, William T. Freeman, Edward H. Adelson and Yair Weiss. *Human-assisted motion annotation*. IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, June 2008. (Cited on pages xii, xvii, 72, 73 and 74.)

[Lowe 2004] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints*. International Journal of Computer Vision, vol. 60, no. 2, pages 91–110, November 2004. (Cited on page 128.)

[Luxburg 2007] U Von Luxburg. *A tutorial on spectral clustering*. Statistics and computing, vol. 17, no. 4, pages 395–416, August 2007. (Cited on page 31.)

[Ma 2012] Bingpeng Ma, Yu Su and Frederic Jurie. *BiCov: a novel image representation for person re-identification and face verification*. In British Machine Vision Conference, 2012. (Cited on page 128.)

[Marc 2002] C Marc, P Stéphane and N Henri. *Segmentation of non-rigid video objects using long term temporal consistency*. In IEEE International Conference in Image Processing, pages 1–4, 2002. (Cited on page 35.)

[Martins 2011] Andre F. T. Martins, Mario A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith and Eric P. Xing. *An augmented Lagrangian approach to constrained MAP*

inference. In International Conference on Machine Learning, 2011. (Cited on page 104.)

[Matthews 2004] Iain Matthews, Takahiro Ishikawa and Simon Baker. *The template update problem.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 6, pages 810–5, June 2004. (Cited on page 133.)

[Metropolis 1953] 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, vol. 21, no. 6, page 1087, 1953. (Cited on page 87.)

[Milan 2013a] Anton Milan. *Energy Minimization for Multiple Object Tracking*. PhD thesis, Darmstadt University of Technology, Germany, 2013. (Cited on pages xiv, 111 and 113.)

[Milan 2013b] Anton Milan, K Schindler and Stefan Roth. *Challenges of Ground Truth Evaluation of Multi-Target Tracking*. In IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2013. (Cited on page 112.)

[Milan 2013c] Anton Milan, K Schindler and Stefan Roth. *Detection-and Trajectory-Level Exclusion in Multiple Object Tracking*. In IEEE Conference on Computer Vision and Pattern Recognition, 2013. (Cited on pages xv, 88, 90, 91, 94, 96, 116, 117, 121, 125 and 126.)

[Mumford 1989] David Mumford and J Shah. *Optimal approximations by piecewise smooth functions and associated variational problems*. Communications on pure and applied mathematics, vol. XLII, no. 5, pages 577–684, 1989. (Cited on page 36.)

[Niebles 2010] J.C. Niebles, Bohyung Han and L. Fei-Fei. *Efficient extraction of human motion volumes by tracking*. In IEEE Conference on Computer Vision and Pattern Recognition, 2010. (Cited on page 24.)

[Niyogi, S.A. and Adelson 1994] E.H. Niyogi, S.A. and Adelson. *Analyzing and recognizing walking figures in XYT*. In IEEE Conference on Computer Vision and Pattern Recognition, 1994. (Cited on pages 23 and 24.)

[Ochs 2011] Peter Ochs and Thomas Brox. *Object Segmentation in Video : A Hierarchical Variational Approach for Turning Point Trajectories into Dense Regions*. In International Conference on Computer Vision, 2011. (Cited on pages viii, 32, 33, 38 and 72.)

[Odobez 1998] Jean-Marc Odobez and Patrick Bouthemy. *Direct incremental model-based image motion segmentation for video analysis*. Signal Processing, vol. 66, no. 2, pages 143–155, April 1998. (Cited on page 36.)

[Okuma 2004] Kenji Okuma, Ali Taleghani, Nando De Freitas, James J Little and David G Lowe. *A Boosted Particle Filter : Multitarget Detection and Tracking*. In European Conference on Computer Vision, 2004. (Cited on page 81.)

[Palou 2013] Guillem Palou and Philippe Salembier. *Hierarchical Video Representation with Trajectory Binary Partition Tree*. In IEEE Conference on Computer Vision and Pattern Recognition, 2013. (Cited on pages 34, 35 and 38.)

[Papazoglou 2013] Anestis Papazoglou and Vittorio Ferrari. *Fast object segmentation in unconstrained video*. In International Conference on Computer Vision, 2013. (Cited on pages viii, 28 and 29.)

[Pirsiavash 2011] Hamed Pirsiavash, Deva Ramanan and Charless C. Fowlkes. *Globally-optimal greedy algorithms for tracking a variable number of objects*. IEEE Conference on Computer Vision and Pattern Recognition, 2011. (Cited on pages 85, 88, 91 and 126.)

[Porikli 2006] F. Porikli, O. Tuzel and P. Meer. *Covariance Tracking using Model Update Based on Lie Algebra*. In IEEE Conference on Computer Vision and Pattern Recognition, 2006. (Cited on pages 128, 132 and 139.)

[Ravichandran 2012] Avinash Ravichandran, Chaohui Wang, Michalis Raptis and Stefano Soatto. *SuperFloxels: A Mid-level Representation for Video Sequences*. In European Conference on Computer Vision, Workshops, 2012. (Cited on page 58.)

[Reid 1979] D. Reid. *An algorithm for tracking multiple targets*. IEEE Transactions on Automatic Control, vol. 24, no. 6, pages 843–854, December 1979. (Cited on page 87.)

[Rissanen 1978] J Rissanen. *Modeling by shortest data description*. Automatica, vol. 14, no. 5, pages 465–471, 1978. (Cited on page 85.)

[Ristivojević 2006] Mirko Ristivojević and Janusz Konrad. *Space-time image sequence analysis: object tunnels and occlusion volumes.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, no. 2, pages 364–76, February 2006. (Cited on page 24.)

[Rother 2004] Carsten Rother, V Kolmogorov and Andrew Blake. *Grabcut: Interactive foreground extraction using iterated graph cuts*. ACM Transactions on Graphics (TOG), vol. 23, no. 3, pages 309–314, 2004. (Cited on page 28.)

[Rubner 1999] Yossi Rubner. *Perceptual Metrics for Image Database Navigation*. PhD thesis, Stanford University, 1999. (Cited on page 54.)

[Ryoo 2007] MS Ryoo and JK Aggarwal. *Hierarchical recognition of human activities interacting with objects*. In IEEE Conference on Computer Vision and Pattern Recognition, 2007. (Cited on pages vii and 6.)

[Sand 2008] Peter Sand and Seth Teller. *Particle Video: Long-Range Motion Estimation Using Point Trajectories*. International Journal of Computer Vision, vol. 80, no. 1, pages 72–91, 2008. (Cited on pages xi, 31, 63, 64, 66 and 67.)

[Schoenemann 2006] Thomas Schoenemann and Daniel Cremers. *Near real-time motion segmentation using graph cuts*. Pattern Recognition, 2006. (Cited on page 37.)

[Senst 2012] Tobias Senst, Volker Eiselein and T Sikora. *Robust local optical flow for feature tracking*. IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 9, pages 1377–1387, 2012. (Cited on page 120.)

[Sheikh 2009] Yaser Sheikh, Omar Javed and Takeo Kanade. *Background Subtraction for Freely Moving Cameras*. In International Conference on Computer Vision, pages 1219–1225. Ieee, September 2009. (Cited on pages viii and 33.)

[Shi 2000] Jianbo Shi and Jitendra Malik. *Normalized cuts and image segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pages 888–905, 2000. (Cited on page 22.)

[Shitrit 2011] Horesh Ben Shitrit, J Berclaz, Francois Fleuret and Pascal Fua. *Tracking multiple people under global appearance constraints*. In International Conference on Computer Vision, 2011. (Cited on pages 90, 91, 116, 117, 121 and 126.)

[Shu Guang 2012] Shu Guang, A Dehghan, O Oreifej, E Hand and M Shah. *Part-based Multiple-Person Tracking with Partial Occlusion Handling*. In IEEE Conference on Computer Vision and Pattern Recognition, June 2012. (Cited on pages 112 and 119.)

[Smith 2004] Paul Smith, T Drummond and R Cipolla. *Layered motion segmentation and depth ordering by tracking edges*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. XX, no. April, pages 1–35, 2004. (Cited on page 36.)

[Sontag 2012] David Sontag, DK Choe and Yitao Li. *Efficiently searching for frustrated cycles in MAP inference*. In Uncertainty in Artificial Intelligence, 2012. (Cited on pages xv, 104, 120 and 124.)

[Sun 2010] Deqing Sun, EB Sudderth and MJ Black. *Layered image motion with explicit occlusions, temporal consistency, and depth ordering*. In Advances in Neural Information Processing Systems, 2010. (Cited on pages viii, 36 and 37.)

[Sundaram 2010] Narayanan Sundaram, Thomas Brox and Kurt Keutzer. *Dense point trajectories by GPU-accelerated large displacement optical flow*. In European Conference on Computer Vision, 2010. (Cited on pages 31 and 108.)

[Szekely 2009] Gabor Szekely and Maria Rizzo. *Brownian Distance Covariance*. The Annals of Applied Statistics, vol. 3, no. 4, pages 1236–1265, January 2009. (Cited on pages xv, 14, 129, 130, 131 and 141.)

[Tianyang 2012] Ma Tianyang and L. J. Latecki. *Maximum weight cliques with mutex constraints for video object segmentation*. In IEEE Conference on Computer Vision and Pattern Recognition, June 2012. (Cited on pages viii, 26, 27, 28 and 33.)

[Tomasi 1991] Carlo Tomasi and Takeo Kanade. *Detection and Tracking of Point Features*. Rapport technique, 1991. (Cited on page 31.)

[Tosato 2010] Diego Tosato, Michela Farenzena, Mauro Spera, Murino Vittorio and Marco Cristani. *Multi-class classification on riemannian manifolds for video surveillance*. In European Conference on Computer Vision, 2010. (Cited on page 136.)

[Trichet 2013a] Remi Trichet and Ram Nevatia. *Video Segmentation with Spatio - Temporal Tubes*. In AVSS, 2013. (Cited on page 35.)

[Trichet 2013b] Rémi Trichet and Ramakant Nevatia. *Video segmentation with spatio-temporal tubes*. In Conference on Advanced Video and Signal-Based Surveillance, 2013. (Cited on page 148.)

[Trichet 2014] Rémi Trichet and Ramakant Nevatia. *Video segmentation and feature co-occurrences for activity classification*. In Winter Conference on Applications of Computer Vision, 2014. (Cited on page 148.)

[Tsai 2011] David Tsai, Matthew Flagg, Atsushi Nakazawa and James M. Rehg. *Motion Coherent Tracking Using Multi-label MRF Optimization*. International Journal of Computer Vision, vol. 100, no. 2, pages 190–202, December 2011. (Cited on pages viii, 25, 27, 28 and 38.)

[Tsitsiklis 1995] JN Tsitsiklis. *Efficient algorithms for globally optimal trajectories*. IEEE Transactions on Automatic Control, vol. 40, no. 9, pages 1528–1538, 1995. (Cited on page 52.)

[Tsochantaridis 2004] I Tsochantaridis and Thomas Hofmann. *Support vector machine learning for interdependent and structured output spaces*. In International Conference on Machine Learning, 2004. (Cited on page 25.)

[Tuzel 2006] Oncel Tuzel, Fatih Porikli and Peter Meer. *Region covariance: A fast descriptor for detection and classification*. In European Conference on Computer Vision, 2006. (Cited on pages 128 and 132.)

[Tuzel 2008] Oncel Tuzel, Fatih Porikli and Peter Meer. *Pedestrian detection via classification on Riemannian manifolds.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 10, pages 1713–27, October 2008. (Cited on pages 128 and 132.)

[Unger 2012] M Unger and M Werlberger. *Joint motion estimation and segmentation of complex scenes with label costs and occlusion modeling*. In IEEE Conference on Computer Vision and Pattern Recognition, 2012. (Cited on pages 36 and 37.)

[Urvoy 2009] Matthieu Urvoy and Nathalie Cammas. *Motion tubes for the representation of images sequences*. In International Conference on Multimedia and Expo, 2009. (Cited on pages viii and 35.)

[Vazquez 2006] C Vazquez. *Joint multiregion segmentation and parametric estimation of image motion by basis function representation and level set evolution*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 5, pages 782–793, 2006. (Cited on page 36.)

[Vermaak 2005] J Vermaak, SJ Godsill and P Perez. *Monte carlo filtering for multi target tracking and data association*. IEEE Transactions on Aerospace and Electronic Systems, vol. 41, no. 1, pages 309–332, 2005. (Cited on page 87.)

[Wang 1990] L I Wang. *Texture Unit, Texture Spectrum, And Texture Analysis*. IEEE Transactions on Geoscience and Remote Sensing, vol. 28, no. 4, pages 509–512, July 1990. (Cited on page 128.)

[Wang 1994] JYA Wang and EH Adelson. *Representing moving images with layers*. IEEE Transactions on Image Processing, vol. 3, no. 5, pages 625–638, 1994. (Cited on page 36.)

[Werlberger 2009] Manuel Werlberger, Werner Trobin, Thomas Pock, Andreas Wedel, Daniel Cremers and Horst Bischof. *Anisotropic Huber-L1 Optical Flow*. British Machine Vision Conference, 2009. (Cited on pages xii, 70, 74 and 108.)

[Willett 2002] P Willett, Y Ruan and R Streit. *PMHT: problems and some solutions*. IEEE Transactions on Aerospace and Electronic Systems, vol. 38, no. 3, pages 738–754, 2002. (Cited on page 87.)

[Xu 2012] Chenliang Xu and Jason J. Corso. *Evaluation of super-voxel methods for early video processing*. In IEEE Conference on Computer Vision and Pattern Recognition, 2012. (Cited on pages 30, 63 and 69.)

[Yang 2011] Bo Yang, Chang Huang and Ram Nevatia. *Learning affinities and dependencies for multi-target tracking using a CRF model*. In IEEE Conference on Computer Vision and Pattern Recognition, volume 2, June 2011. (Cited on page 88.)

[Yang 2012] Bo Yang and R. Nevatia. *An online learned CRF model for multi-target tracking*. In IEEE Conference on Computer Vision and Pattern Recognition, 2012. (Cited on page 88.)

[Yao 2011] Jian Yao and Jean-Marc Odobez. *Fast human detection from joint appearance and foreground feature subset covariances*. Computer Vision and Image Understanding, vol. 115, no. 10, pages 1414–1426, October 2011. (Cited on page 132.)

[Yu 2001] SX Yu and J Shi. *Understanding popout through repulsion*. In IEEE Conference on Computer Vision and Pattern Recognition, 2001. (Cited on page 32.)

[Zach 2008] Christopher Zach, David Gallup and JM Frahm. *Fast gain-adaptive KLT tracking on the GPU*. In IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2008. (Cited on pages 31, 120 and 121.)

[Zamir 2012] AR Zamir, Afshin Dehghan and Mubarak Shah. *GMCP-Tracker: Global Multi-Object Tracking Using Generalized Minimum Clique Graphs*. In European Conference on Computer Vision, 2012. (Cited on pages xiii, 12, 82, 83, 88, 90, 91, 116, 117, 119, 121 and 126.)

[Zhang 2008] L Zhang, Y Li and Ramakant Nevatia. *Global data association for multi-object tracking using network flows*. In IEEE Conference on Computer Vision and Pattern Recognition, 2008. (Cited on pages xiii, 84, 85 and 91.)

[Zhang 2012] Jianming Zhang, LL Presti and Stan Sclaroff. *Online multi-person tracking by tracker hierarchy*. In Conference on Advanced Video and Signal-Based Surveillance, 2012. (Cited on pages 117, 118 and 121.)

[Zheng 2009] Wei-Shi Zheng, Shaogang Gong and Tao Xiang. *Associating Groups of People*. In Procedings of the British Machine Vision Conference 2009, numéro 1, pages 23.1–23.11, 2009. (Cited on pages xvi, 133, 136 and 139.)

[Ziming 2014] Ming-ming Cheng Ziming, Zhang Wen-yan Lin and Philip Torr. *BING : Binarized Normed Gradients for Objectness Estimation at 300fps*. In IEEE Conference on Computer Vision and Pattern Recognition, 2014. (Cited on pages 25 and 148.)

[Zuckerman 2007] David Zuckerman. *Linear degree extractors and the inapproximability of max clique and chromatic number*. ACM Symposium on Theory of Computing, vol. 3, pages 103–128, 2007. (Cited on page 28.)

List of Publications

The following publications were made (or submitted) as a result of the work carried out during this thesis term.

- Ratnesh Kumar, Guillaume Charpiat, Monique Thonnat. "*Hierarchical Representation of Videos with Spatio-Temporal Fibers*", IEEE WACV '2014.
- Ratnesh Kumar, Guillaume Charpiat, Monique Thonnat. "*Multiple Object Tracking using Graph Partitioning*", ACCV '2014.
- Slawomir Bak, Ratnesh Kumar, Francois Bremond. "*Brownian descriptor: A Rich Meta-Feature for Appearance Matching*", IEEE WACV '2014.
- Slawomir Bak, Marco S.B., Ratnesh Kumar, Francois Bremond. "*Exploiting feature correlations for appearance matching*", Submitted to Journal '2014.

Video Segmentation and Multiple Object Tracking

Abstract: In this thesis we propose novel algorithms for video analysis. The first contribution of this thesis is in the domain of video segmentation wherein the objective is to obtain a dense and coherent spatio-temporal segmentation. We propose joining both spatial and temporal aspects of a video into a single notion Fiber. A fiber is a set of trajectories which are spatially connected by a mesh. Fibers are built by jointly assessing spatial and temporal aspects of the video. Compared to the state-of-the-art, a fiber based video segmentation presents advantages such as a natural spatio-temporal neighborhood accessor by a mesh, and temporal correspondences for most pixels in the video. Furthermore, this fiber-based segmentation is of quasi-linear complexity w.r.t. the number of pixels. The second contribution is in the realm of multiple object tracking. We proposed a tracking approach which utilizes cues from point tracks, kinematics of moving objects and global appearance of detections. Unification of all these cues is performed on a Conditional Random Field. Subsequently this model is optimized by a combination of message passing and an Iterated Conditional Modes (ICM) variant to infer object-trajectories. A third, minor, contribution relates to the development of suitable feature descriptor for appearance matching of persons. All of our proposed approaches achieve competitive and better results (both qualitatively and quantitatively) than state-of-the-art on open source datasets.

All of our proposed approaches achieve competitive and better results (both qualitatively and quantitatively) than state-of-the-art on open source datasets.

Keywords: Video Segmentation, Meshes, Point tracks, Partitioning, Labeling, Multiple Object Tracking, Conditional Random Field, Message Passing, ICM.