DUBLIN CITY UNIVERSITY

DOCTORAL THESIS

---

# Model-based Human Upper Body Tracking using Interest Points in Real-time Video

---

*Author:*
Alireza Dehghani, M.Sc.

*Supervisor:*
Dr. Alistair Sutherland

*A dissertation submitted in fulfilment of the requirements*
*for the degree of Doctor of Philosophy*

*in the*

School of Computing

June 2015

# Declaration of Authorship

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:

ID No.: 10113665

Date: June 2015

*”Although my senses were searching the desert fatiguelessly; discovering nothing although finding a lot; my soul was illuminated by a thousand suns; but could never ever touch the perfection of a single atom”*

Avicenna, 10th century

*To my wife Neda, whose patience and support has sustained me through this effort.*

# *Acknowledgements*

I would like to express my sincere gratitude to my supervisor, Dr. Alistair Sutherland for his never ending support and advice. Alistair, you have been more than a supervisor for me. Your constant academic and spiritual supports accompanied me throughout the years.

My gratitude also goes towards Dr. David Moloney at Movidius plc, our partner in this project. His priceless technical and scientific advices have been inspiring for me all the times. Also, I would like to thank him for giving me the opportunity to work at Movidius as an intern in partnership with HiPEAC. Thanks also goes to all people at Movidius because of their support during my intern-ship and the monthly meetings we have had at Movidius or DCU, without whom all the industry inside information would have been impossible to obtain.

Thanks to the members of my research qualifier committee, Prof. Heather Ruskin, Prof. Thomas Baltzer Moeslund, and Dr. Robert Sadleir for having accepted this significant task and also for their feedbacks on my research.

Thanks to Patricia Lacey for her great administrative support. She used to sort everything out for people like me at School of Computing, so I didn't need to spend my time on paperwork. Many thanks Patricia.

I would like to thank my friends at DCU and 3D Vision Group with whom I have had the pleasure of working and living over the years. These include Fattah Alizadeh, Fiona Dermody, Dexmont Pena, Mohammed Farouk, Marlon Oliveira, and Pooyan Jamshidi.

My endless thanks go towards my family for their never ending love and encouragement throughout my whole life and my study. I am very lucky to have such a fantastic family. My gratitude also goes towards my mother's and brother's soul which have been with me all the time from when they started their spiritual journey. I hope my work makes them proud.

Finally, I would like to thank my wife Neda for providing a welcome distraction from school, for bringing me happiness, and for her support, encouragement and love. She bore being far from her parents and her home-town to be beside me and support me during this long journey.

This thesis was funded by the Irish Research Council (IRC) under their Enterprise Partnership Scheme in partnership with Movidius plc. Their support is gratefully acknowledged.

# Contents

**Declaration of Authorship** i

**Acknowledgements** iv

**Contents** v

**List of Figures** ix

**List of Tables** xii

**List of Publications** xiv

**Abbreviations** xv

**Abstract** xvii

**1 Introduction** 1
- 1.1 Research Review . . . . . 1
- 1.2 Motivation . . . . . 5
- 1.3 Significance and Contributions . . . . . 8
- 1.4 Thesis Outline . . . . . 10

**2 Literature Review** 12
- 2.1 Modelling . . . . . 14
  - 2.1.1 Human Body Representation . . . . . 14
  - 2.1.2 Human Body Models . . . . . 24
- 2.2 Background Subtraction . . . . . 27
- 2.3 Human Body Pose Estimation and Tracking . . . . . 30
  - 2.3.1 Top-down Versus Bottom-up . . . . . 30
  - 2.3.2 Single-hypothesis Versus Multiple-hypothesis . . . . . 32
  - 2.3.3 Stochastic versus Heuristic . . . . . 34

2.3.4 IP-Based Object Tracking . . . . . 36
2.4 Conclusions . . . . . 39

**3 IP-Based Background Subtraction 41**
3.1 Motivation . . . . . 43
3.2 IP-Based BGS Algorithm . . . . . 47
3.2.1 Technical Description of the Algorithm . . . . . 47
3.2.2 Post-Processing . . . . . 52
3.3 Experimental Results . . . . . 54
3.3.1 Evaluation Factors . . . . . 55
3.3.2 Experiment 1: Effect of Block Sizes on the Precision and Recall . . . 56
3.3.3 Experiment 2: Robustness Under Low-Light conditions . . . . . 62
3.3.4 Experiment 3: Comparison of the IP-Based BGS with Different BGS Algorithms . . . . . 62
3.3.5 Experiment 4: Effect of Threshold on the Results . . . . . 68
3.4 Concluding Remarks and Future Work . . . . . 69

**4 Interest Point Matching 71**
4.1 Motivation . . . . . 72
4.2 Interest Point Matching Algorithms . . . . . 74
4.2.1 Notations and Definitions . . . . . 75
4.2.2 Stage 1: Local-Based IP Matching . . . . . 76
4.2.2.1 Cross-checking . . . . . 77
4.2.2.2 Displacement-checking . . . . . 77
4.2.3 Stage 2: Spatial-Based IP Matching . . . . . 80
4.2.3.1 Graph-Based Spatial IP Matching . . . . . 82
4.2.3.2 Shape-Context-Based Spatial IP Matching . . . . . 87
4.3 LSR Strategy . . . . . 91
4.4 Experimental Results . . . . . 96
4.4.1 Results of different stages of the Algorithm . . . . . 97
4.4.2 Effect of LSR on the Algorithm . . . . . 97
4.4.3 Performance in terms of Evaluation Factors . . . . . 99
4.5 Concluding Remarks and Future Work . . . . . 105

**5 Two-Stage Hierarchical-Global Model-Based Human Upper Body Pose Estimation and Tracking using PSO 107**
5.1 Motivation . . . . . 108
5.2 Particle Swarm Optimisation . . . . . 111
5.2.1 PSO Notations and Optimisation Process . . . . . 112
5.2.2 PSO Parameters . . . . . 114
5.2.2.1 Velocity Update Terms . . . . . 116
5.2.2.2 Constraint Coefficients . . . . . 118
5.2.2.3 Velocity Clamping . . . . . 118

5.3 PSO-Based HUB Pose Estimation Formulation and Prerequisites . . . . . . . 119
5.3.1 Problem Formulation . . . . . . . . . . . . . . . . . . . . . . . . . . . 119
5.3.2 Modelling the Human Upper Body . . . . . . . . . . . . . . . . . . . 121
5.3.2.1 Pose Synthesis and Projection . . . . . . . . . . . . . . . . 123
5.3.2.2 Degrees of Freedom . . . . . . . . . . . . . . . . . . . . . . 125
5.3.3 IP Rendering for the Pose Hypotheses . . . . . . . . . . . . . . . . . 127
5.3.3.1 Prototyping . . . . . . . . . . . . . . . . . . . . . . . . . . 128
5.3.3.2 Refinement . . . . . . . . . . . . . . . . . . . . . . . . . . 131
5.3.4 Scaling Factor Estimation . . . . . . . . . . . . . . . . . . . . . . . . 135
5.3.4.1 Scaling of the Individual Limbs . . . . . . . . . . . . . . . 136
5.3.5 Cost (Evaluation) Function . . . . . . . . . . . . . . . . . . . . . . . 137
5.4 Implementation and Experimental Results . . . . . . . . . . . . . . . . . . 139
5.4.1 Hierarchical PSO-Based HUB Pose Estimation (HPSO) Algorithm . 139
5.4.1.1 Hierarchy in Estimation . . . . . . . . . . . . . . . . . . . 140
5.4.1.2 HPSO Upper Body Pose Estimation Results . . . . . . . . 144
5.4.1.3 Virtual Matched IP Pair Rendering . . . . . . . . . . . . . 147
5.4.1.4 Error Propagation in Pose Estimation . . . . . . . . . . . . 149
5.4.2 Global-PSO for Consistency Check and Pose Refinement . . . . . . . 150
5.4.3 Experimental Results . . . . . . . . . . . . . . . . . . . . . . . . . . 153
5.4.3.1 Performance Evaluation . . . . . . . . . . . . . . . . . . . 156
5.5 Conclusions and Future Work . . . . . . . . . . . . . . . . . . . . . . . . . 157

**6 Overall Conclusions and Future Work** **159**
6.1 Summary of Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . 160
6.1.1 IP-based Background Subtraction (Chapter 3) . . . . . . . . . . . . . 160
6.1.2 IP Matching (Chapter 4) . . . . . . . . . . . . . . . . . . . . . . . . . 161
6.1.3 Two-Stage Hierarchical-Global Model-based Human Upper Body Pose Estimation and Tracking by Means of PSO (Chapter 5) . . . . . . . . 162
6.2 Directions for Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . 164
6.2.1 Developing Sparse Depth Extraction (SDE) . . . . . . . . . . . . . . 164
6.2.2 Extending the BSG algorithm to Support Moving Camera Situations. 165
6.2.3 Automatic Pose Initialization . . . . . . . . . . . . . . . . . . . . . . 166
6.2.4 Occlusion Handling . . . . . . . . . . . . . . . . . . . . . . . . . . . 167
6.2.5 Extending the Whole System to 3D . . . . . . . . . . . . . . . . . . . 167

**Bibliography** **168**

**A Kinematic Human Body Model** **A1**

**B Particle Swarm Optimisation Considerations** **B1**
B.1 PSO versus Bayesian Filtering . . . . . . . . . . . . . . . . . . . . . . . . B1

B.2 Convergence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B2

**C Homography Estimation** **C1**

**D Implementation and Programming** **D1**

# List of Figures

1.1 A human motion tracking example and its different stages, adapted from (Manzanera, 2009). . . . . 5
1.2 Framework of our human upper body pose estimation and tracking system. . 8

2.1 Object representations. (a) Centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) control points on object contour, (h) complete object contour, (i) object silhouette, copied from (Yilmaz et al., 2006). . . . . 15
2.2 Animation of a Motion Capture Sequence taken for a single body scan subject. The muscle deformations are synthesized automatically from the space of pose and body shape deformations, copied from (Anguelov et al., 2005). . . . . . 27
2.3 detector-descriptor-based visual tracking system (Gauglitz et al., 2011). . . . 37

3.1 The block and its different types. . . . . 47
3.2 A sample Event $\{16,35,67,132,153\}$ for IPs $(2,2),(3,6),(5,8),(9,13),(11,4)$ in a block with size $15\times15$ pixels. . . . . 49
3.3 A binary search tree: the larger inputs feed into the right sub-trees and the smaller ones feed into the left sub-trees. . . . . 49
3.4 Repetition Index for Events of a block after 1000 frames. . . . . 50
3.5 The real image with its IPs, red: foreground IPs, blue: background IPs. . . . . 53
3.6 The 2D kernel of $3\times3$ filter. . . . . 53
3.7 Left: foreground blocks, Right from bottom to top: the real image, foreground IPs before, and after post-processing. . . . . 54
3.8 Evaluation Factors versus block sizes for resolution $256\times512$. . . . . 57
3.9 6 frames vs different block sizes of $8\times8$, $16\times16$, and $32\times32$. . . . . 59
3.10 Run-time versus block size . . . . . 60
3.11 Comparison between IP-based and pixel-based BGS methods under low-light conditions over 5 random frames of a video with 450 frames. . . . . 63
3.12 F-measures (Eq. 3.4) and total number of errors versus threshold. . . . . 69

4.1 IP-matching algorithm block diagram. . . . . 74
4.2 Local IPM block diagram. . . . . 76
4.3 Why cross-checking? Left to right: green to red matching, red to green matching, and Cross-checked matching. . . . . 77

4.4 Estimated PDF as well as the number of repetitions for different displacement length throughout several frames. . . . . . . . . . . . . . . . . . . . . . . . . 79
4.5 Displacement Threshold estimation over 1200 frames. . . . . . . . . . . . . . 80
4.6 The Graph $g_r$, confidently matched points $cr_1, cr_2, \ldots, cr_n$, and unmatched point $ur_i$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
4.7 Graph formation: (a) the clustered confidently matched IPs, (b) the unmatched target IPs, (c) search areas, (d) a reference graph, and (e-i) five target graphs for unmatched target IPs in the search area. . . . . . . . . . . . . . . . . . . . 83
4.8 An example edit graph G with $n = 5$ and $m = 4$, the edit operations, the shortest path from **s** to **t**, and its corresponding edit sequence, adapted from (Wu and Wang, 1999). Weights are calculated using the edit cost function of Eq. 4.7. . 86
4.9 The edit graph $H$ for the graph $G$ of Fig. 4.8 adapted from (Wu and Wang, 1999). Weights are calculated using the edit cost function of Eq. 4.7 . . . . . 87
4.10 Shape Context feature descriptors (Mori et al., 2005). . . . . . . . . . . . . . 89
4.11 SC matching process: (a) the matched and unmatched Reference IPs, red and green respectively (b) the matched and unmatched Target IPs, red and green respectively, (c) a highlighted unmatched Reference IP, yellow (d) some highlighted candidate unmatched Target IPs, cyan colour, inside the search area, red colour, (e) SC histogram of unmatched Reference IP, and (f-l) SC histogram of seven unmatched Target IPs inside the search area. . . . . . . . . . 91
4.12 The LSR for the first two rounds of matching. . . . . . . . . . . . . . . . . . 96
4.13 Preprocessing for IP matching; Left to right: image, FAST IPs, foreground IPs. 97
4.14 Results of local-based stage of the proposed IP matching algorithms: (a) "Brute-Force" matching, (b) cross-checking, (c) the confidently matched IPs. . . . . 98
4.15 Results of spatial-based stage of the proposed IP matching algorithms: (a) Graph-based method; (b) SC-based method. . . . . . . . . . . . . . . . . . . . 99
4.16 Final result of proposed IP matching algorithm for some consecutive frames ”without” (a) and ”with” (b) the LSR strategy. . . . . . . . . . . . . . . . . . . 100
4.17 Matched IPs of some consecutive and non-consecutive frames, ”without” and ”with” LSR. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 101
4.18 Precision and Recall curves of the algorithm. . . . . . . . . . . . . . . . . . . 103
4.19 Precision-Recall curve of the algorithm. . . . . . . . . . . . . . . . . . . . . . 104

5.1 Position constraint process over 4 iterations for particle $i$. . . . . . . . . . . . 115
5.2 PSO optimisation process in iteration $t$. . . . . . . . . . . . . . . . . . . . . . 116
5.3 The skeletal HB model and its kinematic tree and paths. . . . . . . . . . . . . 123
5.4 Skeletal HB model and its anthropometric information (Han, 2010). . . . . . 124
5.5 Six DoFs of a rigid object adapted from Wikipedia (Wikipedia). . . . . . . . 125
5.6 Human body, the joints, the DoFs of joints. . . . . . . . . . . . . . . . . . . . 126
5.7 Shortest distance from a point to a line. . . . . . . . . . . . . . . . . . . . . . 128
5.8 IP labelling with limb distance thresholds $\{\Theta_j\}_{j=0}^{7}$. . . . . . . . . . . . . . . 130
5.9 Limb distance thresholds $\{\Theta_j\}_{j=0}^{7}$ in pictorial model. . . . . . . . . . . . . 131
5.10 IP prototyping stage. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132

5.11 The process of rendering IPs $\Pi_k^h$ for the pose hypothesis $P_k^h$. . . . . . . . . . . . 135
5.12 The Hamming distance between two strings of equal length. . . . . . . . . . 138
5.13 The hierarchical PSO-based HUB pose estimation algorithm block diagram. . 140
5.14 Levels in the kinematic tree. . . . . . . . . . . . . . . . . . . . . . . . . . 143
5.15 The initial pose $\mathbf{X} = [175, 150, 0, 0, 65, 70, 125, 120]$, which we use in our experiments. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 144
5.16 6 rounds of pose estimation from frame 710 to 715. . . . . . . . . . . . . . . 146
5.17 Generating the random virtual IPs for a limb. . . . . . . . . . . . . . . . . . 148
5.18 Spatial and temporal propagation errors over *5* rounds of pose estimation from frame 716 to 720. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 149
5.19 6 non-consecutive rounds of pose estimation from frame 726 to 784. . . . . . 152
5.20 8 non-consecutive rounds of pose estimation from frame 716 to 784, where GPSO amend the result of HPSO in Fig. 5.16 and 5.19. . . . . . . . . . . . . . 155
5.21 Euclidean distances between the estimated pose and the GT pose over *several* frames of the test video for the HPSO and GPSO stages of the proposed approach. 157

# List of Tables

1.1 The typical assumptions made by motion capture systems listed in ranked order according to frequency, adapted from (Moeslund and Granum, 2001). . . 4

2.1 Existing feature-based visual tracking systems (Gauglitz et al., 2011). . . . . 18

3.1 The stored Events of block 102 of image for 1000 frames. . . . . . . . . . . . 51
3.2 Evaluation factors in classification context . . . . . . . . . . . . . . . . . . . 55
3.3 The Wallflower dataset scenarios. . . . . . . . . . . . . . . . . . . . . . . . . 64
3.4 Comparison on the Wallflower dataset for different BGS algorithms. . . . . . 65
3.5 Results on the Wallflower dataset for different BGS algorithms. . . . . . . . . 66

4.1 Performance comparison on the image pairs in Figs. 4.14 and 4.15. . . . . . . 104

5.1 Summary of results of experiment on inertia weight over 5 optimization test problems (Bansal et al., 2011). . . . . . . . . . . . . . . . . . . . . . . . . . . 117
5.2 Kinematic HB model; the joints and the links. . . . . . . . . . . . . . . . . . . 122
5.3 Kinematic HB model joints DoFs for both upper and lower bodies, though only the upper body part is used in this thesis. . . . . . . . . . . . . . . . . . . 127
5.4 The 7 hierarchical steps of our HPSO upper body pose optimisation. . . . . . 141
5.5 Average of MSE over all the frames for HPSO and GPSO. . . . . . . . . . . 157

# List of Algorithms

3.1 IP-Based BGS Algorithm . . . 52
4.1 Local-Based IP Matching Algorithm . . . 81
4.2 Graph-Based Spatial IP Matching Algorithm . . . 88
4.3 SC-Based Spatial-based IP Matching Algorithm . . . 90
4.4 LSR strategy: IP Scoring Algorithm . . . 94
4.5 LSR strategy: List Refinement Algorithm . . . 95
5.1 IP Labelling Procedure . . . 129
5.2 IP Rendering Algorithm . . . 133
5.3 Hierarchical PSO-Based HUB Pose Estimation Algorithm . . . 142
5.4 Virtual-Matched-IP-Pair-Rendering Algorithm . . . 148

# *List of Publications*

- A. Dehghani and A. Sutherland. A combined two-stage local-spatial interest point matching algorithm. In *8th Iranian Conference on Machine Vision and Image Processing (MVIP)*, pages 105–109. IEEE, 2013
- A. Dehghani and A. Sutherland. A novel interest-point-based background subtraction algorithm. In *ELCVIA: Electronic Letters on Computer Vision and Image Analysis*, volume 13, pages 0050–67, 2014b
- A. Dehghani and A. Sutherland. A dynamic hybrid local-spatial interest point matching algorithm for articulated human body tracking. In *International Conference on Pattern Recognition Applications and Methods (ICPRAM 2014)*, pages 536–543. SCITEPRESS, 2014a
- A. Dehghani, A. Sutherland, D. Moloney, and D. Pena. An improved interest point matching algorithm for human body tracking. In *1st International Image Processing, Applications and Systems Conference (IPAS 2014).*, pages 1–6. IEEE, 2014
- A. Dehghani and A. Sutherland. A two-stage hierarchical-global model-based human upper body pose estimation and tracking using particle swarm optimisation. In *to be submitted to 26th British Machine Vision Conference (BMVC 2015)*, 2015. Manuscript submitted for publication
- F. Alizadeh, A. Sutherland, and A. Dehghani. A simple and efficient approach for 3d model decomposition. In *8th Iranian Conference on Machine Vision and Image Processing (MVIP 2013)*, pages 1–4. IEEE, 2013

# Abbreviations

| | |
|---|---|
| **ACO** | **A**nt **C**olony **O**ptimization |
| **APF** | **A**nnealed **P**article **F**ilter |
| **BGS** | **B**ack**G**round **S**ubtraction |
| **BG** | **B**ack**G**round |
| **BG-IP** | **B**ack**G**round **I**nterest **P**oint |
| **CenSurE** | **Cen**tre **Sur**round **E**xtrema |
| **CT** | **C**ensus **T**ransform |
| **CV** | **C**omputer **V**ision |
| **DoF** | **D**egree of **F**reedom |
| **DoG** | **D**ifference **o**f **G**aussians |
| **DT** | **D**isplacement **T**hreshold |
| **FBB** | **F**oreground **B**lock **B**inary |
| **FG** | **F**ore**G**round |
| **FG-IP** | **F**ore**G**round **I**nterest **P**oint |
| **FP** | **F**alse **P**ositive |
| **FN** | **F**alse **N**egative |
| **GA** | **G**enetic **A**lgorithm |
| **GLOH** | **G**radient **L**ocation and **O**rientation **H**istogram |
| **H-G** | **H**ierarchical-**G**lobal |
| **GT** | **G**round **T**ruth |
| **HB** | **H**uman **B**ody |
| **HBT** | **H**uman **B**ody **T**racking |

| | |
|---|---|
| **HPSO** | **H**ierarchical **P**article **S**warm **O**ptimization |
| **HUB** | **H**uman **U**pper **B**ody |
| **HUBT** | **H**uman **U**pper **B**ody **T**racking |
| **IDE** | **I**ntegrated **D**evelopment **E**nvironment |
| **IP** | **I**nterest **P**oint |
| **IPM** | **I**nterest **P**oint **M**atching |
| **KDE** | **K**ernel **D**ensity **E**stimation |
| **LMeDS** | **L**east **MeD**ian of **S**quares |
| **LSR** | **L**ist **S**coring and **R**efinement |
| **MSE** | **M**ean **S**quare **E**rror |
| **MoCap** | **Mo**tion **Cap**ture |
| **OpenCV** | **O**pen **S**ource **C**omputer **V**ision |
| **ORB** | **O**riented FAST and **R**otated **B**RIEF |
| **PCL** | **P**oint **C**loud **L**ibrary |
| **PF** | **P**article **F**ilter (**P**article **F**iltering) |
| **RI** | **R**epetition **I**ndex |
| **PSO** | **P**article **S**warm **O**ptimization |
| **RANSAC** | **RAN**dom **SA**mple **C**onsensus |
| **RGB** | **R**ed-**G**reen-**B**lue |
| **RLL** | **R**eference **L**ist **L**eakage |
| **SC** | **S**hape **C**antext |
| **SIFT** | **S**cale **I**nvariant **F**eature **T**ransform |
| **SURF** | **S**peeded **U**p **R**obust **F**eatures |
| **SVD** | **S**ingular **V**alue **D**ecomposition |
| **SVM** | **S**upport **V**ector **M**achine |
| **TN** | **T**rue **N**egative |
| **TP** | **T**rue **P**ositive |

# *Abstract*

## Model-based Human Upper Body Tracking using Interest Points in Real-time Video

by Alireza Dehghani, M.Sc.

Vision-based human motion analysis has received huge attention from researchers because of the number of applications, such as automated surveillance, video indexing, human-machine interaction, traffic monitoring, and vehicle navigation. However, it contains several open problems. To date, despite very promising proposed approaches, no explicit solution has been found to solve these open problems efficiently. In this regard, this thesis presents a model-based human upper body pose estimation and tracking system using interest points (IPs) in real-time video.

In the first stage, we propose a novel IP-based background-subtraction algorithm to segment the foreground IPs of each frame from the background ones. Afterwards, the foreground IPs of any two consecutive frames are matched to each other using a dynamic hybrid local-spatial IP matching algorithm, proposed in this research.

The IP matching algorithm starts by using the local feature descriptors of the IPs to find an initial set of possible matches. Then two filtering steps are applied to the results to increase the precision by deleting the mismatched pairs. To improve the recall, a spatial matching process is applied to the remaining unmatched points.

Finally, a two-stage hierarchical-global model-based pose estimation and tracking algorithm based on Particle Swarm Optimiation (PSO) is proposed to track the human upper body through consecutive frames. Given the pose and the foreground IPs in the previous frame and the matched points in the current frame, the proposed PSO-based pose estimation and tracking algorithm estimates the current pose hierarchically by minimizing the discrepancy between the hypothesized pose and the real matched observed points in the first stage. Then a global PSO is applied to the pose estimated by the first stage to do a consistency check and pose refinement.

# Chapter 1

# Introduction

*This sky where we live is no place to lose your wings, so love, love, love.*

HAFEZ, 14TH CENTURY, PERSIAN POET.

This chapter introduces the central concept of the research and provides a high-level explanation of the motivation and approach for the research. In Section 1.1 this field of research is reviewed briefly. Then, the motivation behind this thesis is presented in Section 1.2. The significance and contributions presented in this work are summarized in Section 1.3. Finally, Section 1.4 outlines the thesis contents.

## 1.1 Research Review

Human motion analysis has been one of the most challenging and active research topics in Computer Vision (CV) for more than 30 years. This strongly growing interest has been motivated by its inherent complexity as well as its number of existing and forthcoming applications, such as automated surveillance, video indexing, human-machine interaction, traffic monitoring, vehicle navigation etc. (Yilmaz et al., 2006). Factors such as the speed and price of the existing technologies, and the eagerness of the market for security products have increased this attention towards making human motion capture and analysis automatic and as reliable as possible. The number of papers presented in conferences, journals, workshops as well as the size of budgets, which have been assigned to this research field by funding agencies, prove its importance (Moeslund et al., 2006).

This field of research deals with the analysis of images involving humans. Theoretically, it could involve the analysis of every detail in the movement of the Human Body (HB) from facial movements to changes in the skin due to tightening the muscles (Poppe, 2007). However, in practice only the larger limbs, which make up the articulated human body, are dealt with. Among others, face recognition, hand gesture recognition, whole/partial body tracking and activity recognition are some of the most active research topics of this field (Gavrila, 1999). Nevertheless, some other topics such as eye gaze tracking and recognition, head pose estimation, gait analysis are investigated also by researchers. The main purpose of this thesis is Human Upper Body (HUB) pose estimation and tracking. The methodology of research in this topic is affected by several parameters such as (Moeslund and Granum, 2001):

- The limb or limbs to be tracked.
- The type of sensors which capture the motion (visual light, infra-red, range data, etc.).
- The number and structure of sensors.
- The method of tracking, model-based versus model-free methods (which are also called appearance-based or view-based methods).
- The number of people to be tracked.
- The tracking environment (indoor versus outdoor).
- Tracking in 2D versus 3D.

Despite the huge amount of work which has been done in articulated pose estimation and tracking, it is still a largely unsolved problem due to the challenges it faces. These challenges can be divided as follows (Moeslund et al., 2011):

- The human body is an object with variable visual appearance in the image.

- The illumination condition is variable with the time and environment.
- The physics of the human body is not fixed and changes from person to person.
- Partial occlusions by the human body itself or the objects in the scene.
- Non-linear and complex structure of the human body skeleton.
- The pose space is fairly high-dimensional due to the high number of degrees of freedoms (DoFs).
- Image acquisition implies the loss of 3D information.

All the above factors are important challenges in this field. However, the lack of 3D information is more important than others. Observation from multiple cameras and sensors tackles this problem to an extent and makes the articulated human body pose estimation and tracking more feasible, particularly in practical use. Nonetheless, the single camera strategy is still the focus of research despite it substantial challenging nature (Moeslund et al., 2011), because:

- The depth information can be calculated using the single camera (Saxena et al., 2007).
- Monocular computer vision approaches requires less computational cost, because only one camera is processed at each frame.
- Stereo-based depth calculations imply a high-cost matching process, which is even more challenging in low light and/or low resolution situations and results in noisy and low-resolution depth maps. So, monocular techniques would be an affordable solution in these situations. As an instance, Crivellaro et al. (2014) proposed a monocular-based approach for tracking specular and poorly textured objects, which only requires a standard monocular camera and has no need for a depth sensor.
- In no-occlusion situations, which are the objective of this thesis, a single camera can be used to handle even the 3D movements.

In order to make tracking easier, and more feasible, and tackle some of the above mentioned challenges, different types of constraints can be imposed on the motion. The appearance of the target, the light condition of the environment, the length of time which the person is in scene, the number of people in the scene, the background colour and contents, are some of these constraints, which influence the direction of the approach (Yilmaz et al., 2006). According to Moeslund and Granum (2001), the typical assumptions, which apply constraints on the conditions of motion, are divided into movement and appearance assumptions. Table 1.1 summarizes the typical assumptions which apply constraints on the conditions of motion in ranked order according to their popularity.

TABLE 1.1: The typical assumptions made by motion capture systems listed in ranked order according to frequency, adapted from (Moeslund and Granum, 2001).

| **Assumptions related to movements** | **Assumptions related to appearance** | |
|---|---|---|
| The subject remains inside the workspace | **Environment** | Constant lighting |
| None or constant camera motion | | Static background |
| Only one person in the workspace at the time | | Uniform background |
| The subject faces the camera at all time | | Known camera parameters |
| Movements parallel to the camera-plane | | Special hardware |
| No occlusion | **Subject** | Known start pose |
| Slow and continuous movements | | Known subject |
| Only move one or a few limb | | Markers placed on the subject |
| The motion pattern of the subject is known | | Special coloured clothes |
| Subject moves on a flat ground plane | | Tight-fitting clothes |

According to Yilmaz et al. (2006), the task of human motion tracking is classified into three key steps: detection of the target in the scene (sometimes referred to as ”separating foreground from background”), tracking the detected object or objects, and analysing the tracked movements to understand them and recognize some pre-defined actions. Similarly, Moeslund and Granum (2001) categorize it into initialization, tracking, pose estimation, and recognition.

Fig. 1.1[1] shows a human motion tracking example and its different stages, adapted from (Manzanera, 2009). In comparison to most research work, which emphasises just one or some of these steps (Moeslund and Granum, 2001), industrial research is supposed to cover almost all of the steps. This thesis, which has been supported by Movidius[2], an Irish company active in the Mobile Vision Processor field, seeks to carry out a human upper body pose estimation and tracking in the presence of some constraints which will be described in the following sections.

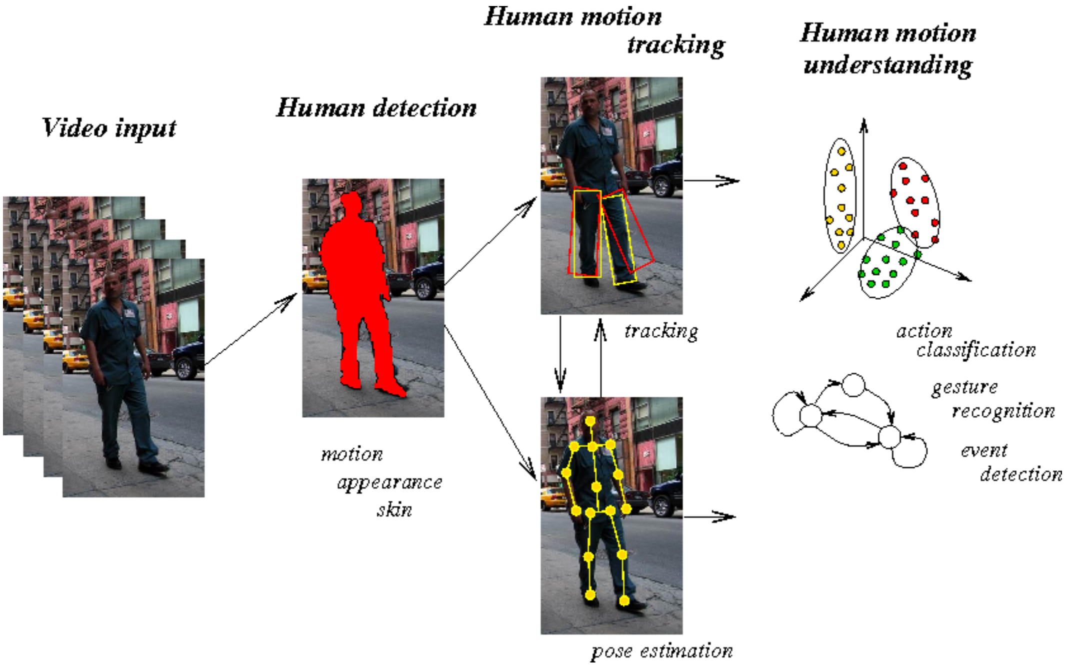


FIGURE 1.1: A human motion tracking example and its different stages, adapted from (Manzanera, 2009).

## 1.2 Motivation

Different types of commercial human motion-capture analysis systems (marker-based and marker-less) have been proposed in recent years. Some of them, such as Motion Capture

---

[1] Human motion analysis: Tools, models, algorithms and applications http://perso.ensta-paristech.fr/~manzaner/Download/Tutorials/Tutorial_HumanMotion_LACNEM09.pdf

[2] Movidius: http://www.movidius.com/

(MoCap)[3] and Vicon[4] can capture human movements at a very high frequency rate with an acceptable level of accuracy, which is good for industrial applications such as the film industry. However, mostly they do not have an affordable price to be used in general by people in mobile and wearable applications. Moreover, they force the user to wear some special clothing and even operate under multiple permanently installed cameras in specific locations like studio environments. In comparison, some others, like Kinect[5] by Microsoft, satisfy home application users, particularly in gaming, by their price. However, they use different types of sensors like laser cameras for capturing the 3D movement, which makes it infeasible to be used in mobile and wearable devices in terms of the size and energy consumption.

By contrast, the new trend in this context is using sensors with an affordable power consumption and price rate as well as with high frame rate capture capability. It is becoming more popular in a diverse range of applications from home computers to mobile phones and wearable devices. In this regard, a single sensor like most of the traditional vision based devices, multiple sensors of the same type such as Google's Project Tango[6], or a combination of different sensors like Amazon's fire phone[7] can be used. Embedded platforms like Movidius's Myriad1[8] and Myriad2[9] offer very low power consumption rates in a stereo-vision-based framework for realizing affordable and efficient vision-based intelligent devices and systems.

This new trend started a new era a few years ago and has tackled the dependency on markers and specific studio-like environments using expensive devices. However, motion tracking is still a challenging task, which has not been solved yet and there are a number of hard and open problems. The high dimensionality of the human body, its fast movements, self-occlusion, and occlusions by other objects or people, illumination changes of the environment are some of the problems which researchers are still trying to solve.

---

[3] MoCap: `https://www.xsens.com/tags/motion-capture/`

[4] vicon: `http://www.vicon.com/`

[5] Kinect for Xbox One: `http://www.microsoft.com/en-us/kinectforwindows/`

[6] Project Tango: `https://www.google.com/atap/projecttango/#project`

[7] Amazon Fire phone: `http://www.amazon.com/Fire_Phone_13MP-Camera_32GB/dp/B00EOE0WKQ`

[8] Movidius's Myriad1: `http://www.movidius.com/our-technology/myriad1/`

[9] Movidius's Myriad2: `http://www.movidius.com/our-technology/myriad-2-platform/`

In this work, we are seeking *a robust pose estimator and tracker in a* ***monocular*** *scenario based on interest points (IPs) to estimate and track the pose of the upper body of a person in an indoor environment under slight changes of illumination*. The underlying system aims to estimate and track 3D movements in a 2D pose space. As mentioned earlier in this chapter, this thesis is an industry-supported research and covers all the steps in the process that would be required for industrial implementation. The detection of a person as the foreground object – a process, which is named *background subtraction* (BGS) - will be done first. Model-based human upper body pose estimation and tracking (HUBT) by means of IP matching between the consecutive frames is the next goal of this thesis. We examine the efficiency of using the IPs for the task of human body tracking (HBT). Owing to the limited time we have had in this research, due to covering all the steps from BGS to pose estimation, the occlusion problem has not dealt with in this research. Similarly, we will not consider the pose initialization process in this work. Instead, we suppose the initial pose is available and we focus on estimating the pose of the subsequent frames using the IPs in a Particle-Swarm-Optimization(PSO)-based framework. Nevertheless, we have designed an automatic pose initialization algorithm as well as some other future works, which will be discussed in Chapter 6. Our proposed model-based human upper body pose estimation and tracking system is illustrated in Fig. 1.2.

Due to the lack of a suitable ground-truth dataset for comparing the efficiency of different stages of our proposed system (Chapters 3-5) with the other works, a test video containing a person acting in front of the camera with a range of pose changes captured from a static camera is used in this regard. However, the BGS algorithm of Chapter 3 will be compared with the popular Wallflower dataset (Toyama et al., 1999). In terms of the frame rate of the test video, we aimed to maintain the smoothness or small inter-frame motion assumptions, which are valid to assume in human body tacking (Herda et al., 2000). The test video has been captured with 15 frames per second. Increasing the frame rate would help the proposed system because it would increase the inter-frame motion smoothness, while decreasing the frame rate would violate the assumption which will cause failure in the system. Experimental results for the frame rate are presented on p. 159.

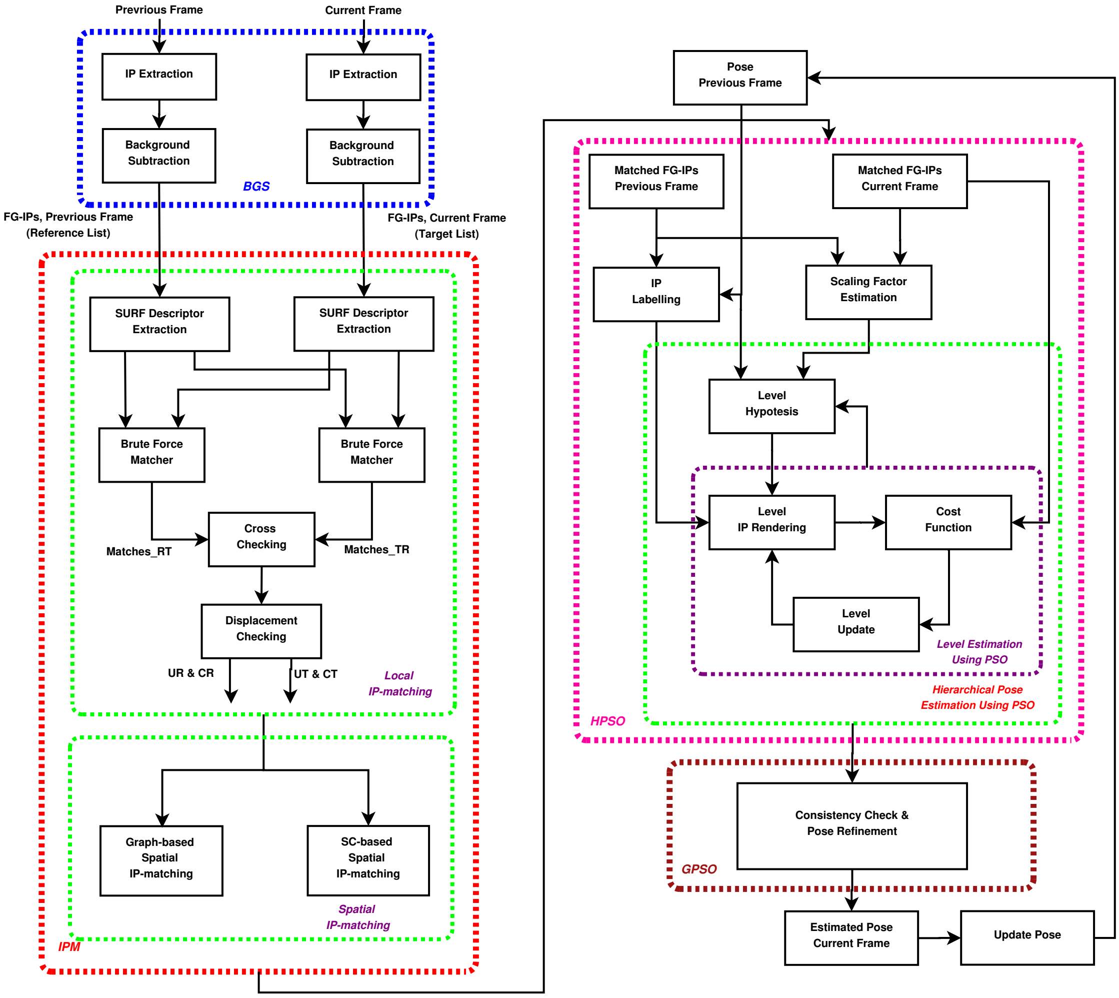

Prevrious Frame
Current Frame
IP Extraction
IP Extraction
Background Subtraction
Background Subtraction
BGS
FG-IPs, Prevrious Frame (Reference List)
FG-IPs, Current Frame (Target List)
SURF Descriptor Extraction
SURF Descriptor Extraction
Brute Force Matcher
Brute Force Matcher
Cross Checking
Matches_RT
Matches_TR
Displacement Checking
UR & CR
UT & CT
Local IP-matching
Graph-based Spatial IP-matching
SC-based Spatial IP-matching
Spatial IP-matching
IPM
Pose Previous Frame
Matched FG-IPs Previous Frame
Matched FG-IPs Current Frame
IP Labelling
Scaling Factor Estimation
Level Hypotesis
Level IP Rendering
Cost Function
Level Update
Level Estimation Using PSO
Hierarchical Pose Estimation Using PSO
HPSO
Consistency Check & Pose Refinement
GPSO
Estimated Pose Current Frame
Update Pose

FIGURE 1.2: Framework of our human upper body pose estimation and tracking system.

## 1.3 Significance and Contributions

This thesis makes three major contributions as well as a few minor contributions to the field of computer vision. These contributions, which constitute sequentially the proposed model-based Human Upper Body Tracking (HUBT) system, are:

- **IP-based Background Subtraction (Chapter 3)**: the first major contribution of this thesis is a novel IP-based background subtraction algorithm, which can be used in any

IP-based tracking application beyond our project. Current Background Subtraction algorithms are mostly pixel-based methods. We propose an IP-based BGS approach applicable in IP-based CV applications.

Briefly, based on a block-wise processing strategy, the frames are divided into blocks of the same size. The IPs inside each block are treated together as Events. Throughout the frame sequence, the algorithm stores the Events in each block as well as the numbers of their occurrences (Repetition Index (RI)) in a Binary Tree. The RI is used then to classify Events as either background or foreground. The background Events appear significantly more often than foreground Events. Events with an RI greater than a certain threshold are classified as background, the rest as foreground. This Event classification is used to label IPs of frames into the foreground and background IPs (BG-IPs).

- **IP-matching (Chapter 4)**: following the BGS algorithm, we propose a dynamic hybrid local-spatial IP matching algorithm, which is used to match the foreground IPs of any two consecutive frames. This algorithm finds an initial set of possible matched pairs using the local feature descriptors of the IPs. Then, two cross-checking and displacement-checking filtering steps are applied to the results to increase the precision by deleting the mismatched pairs. The spatial matching stage, which uses the spatial relation between IPs, is applied then to the remaining unmatched IPs to improve the recall, while holding the precision at the same level. Two different spatial matching strategies, i.e. graph-based and the Shape-Context-based, have been developed for the second stage of the IP matching algorithm.

- **Model-based Human Upper Body Tracking by means of PSO (Chapter 5)**: the final major contribution of this work is a two-stage hierarchical-global model-based articulated human upper body tracking approach based on Particle Swarm Optimisation (PSO). This algorithm, which is a combined bottom-up top-down approach, estimates the skeletal pose for the current frame given the pose in the previous frame and the matched foreground IPs between the previous and current frames.

  The two PSO-based pose estimators of the first and second stages in the proposed algorithm similarly hypothesize a number of pose candidates around the potential position for the pose in the current frame. Then, a set of IPs with the same number of points as IPs in the previous frame is rendered for any hypothesized pose. A discrepancy function, which calculates the similarity between the IPs of the current frame and the rendered IPs of a pose hypothesis, measures the estimation cost for that hypothesised pose. PSO, starts with a random generation of solutions in the first iteration and moves the solutions toward the optimum position iteratively to find the pose with lowest cost.

Taking advantage of the hierarchical nature of our tree-like kinematic model, the proposed two-stage hierarchical-global PSO approach tackles the intrinsic difficulty of solving the optimization problem in the entire high dimensional pose space at once. Our approach solves this problem through a divide-and-conquer strategy in the first stage, by decomposing the parameters of the pose vector and estimating them separately. It optimizes the corresponding parameters of the joints in the pose vector hierarchically in several levels, from the higher to lower order. Thereby, it reduces the complexity of the search and overcomes the mentioned drawbacks. Afterwards, the second stage operates on the pose estimated at the first stage globally at once to do a consistency check and refine the estimated pose. This stage acts as a post-processing stage on the result of the first stage to compensate for the inaccurate estimated parameters of the pose vector.

The proposed algorithms benefit from:

- A hierarchical model-based pose estimation method (HPSO) in the first stage to cut down the complexity and computational cost of a high-dimensional optimisation problem.
- A post-processing pose refinement method (GPSO) to compensate for the spatial and temporal propagation errors caused by the first stage.

## 1.4 Thesis Outline

Following on from the above, the thesis is organized as follows:

- **Chapter 2** discusses the concepts of different areas related to articulated human body tracking as well as the state-of-the-art approaches in this field.
- **Chapter 3** is dedicated to a comprehensive discussion of the proposed novel IP-based background subtraction algorithm.
- **Chapter 4** describes extensively the proposed two-stage IP-matching algorithms. Both the graph-based and shape-context-based approaches are explained and compared in this chapter.
- **Chapter 5** goes through the two-stage hierarchical-global model-based human upper body pose estimation and tracking using the PSO.
- **Chapter 6** summarizes the implications and contributions, and discusses the possibilities for future work.
- **Appendix A** introduces the deployed skeletal model in terms of its implementation.
- **Appendix B** presents the complementary discussion and considerations about the PSO.
- **Appendix C** discusses the geometric transformation we need in Chapter 5.
- **Appendix D** deals with the implementation considerations. We discuss the tools and the libraries we use for developing the proposed ideas.

# Chapter 2

# Literature Review

*To live is the rarest thing in the world. Most people exist, that is all.*

---

OSCAR WILDE, IRISH POET.

As stated earlier in Chapter 1, human motion analysis is a very hot topic in computer vision, which has been discussed for more than three decades. The importance and popularity of this field of research, as well as the number of published papers in conferences and journals has led to several surveys with different taxonomies based on how they deal with (Moeslund and Granum, 2001):

- the object representation.
- the image features.
- the modelling of the object motion.
- the appearance and shape.
- the type and number of sensors

Some of the most popular surveys are as follows:

- Moeslund and Granum (2001) and Moeslund et al. (2006) focus on the overall structure of the motion capture systems and review these based on the subsequent phases in the pose estimation process, initialization, tracking, pose estimation, and recognition. Moeslund et al. (2011) introduce a coherent text that gives a comprehensive review of progress and open-problems.

- Gavrila (1999) divides the research into 2D and 3D approaches, where 2D methods are divided into the methods with or without explicit use of shape models.

- Aggarwal and Cai (1997), Aggarwal and Ryoo (2011), and Wang et al. (2003b) present a taxonomy with three categories: human detection, tracking, and activity recognition. The tracking parts of these surveys are divided into model-based and model-free methods. Similarly, Ji and Liu (2010) have proposed the same taxonomy for only the view-invariant approaches.

- Ye et al. (2013) give an overview of approaches that use depth data to perform human motion analysis. Meantime, they discuss traditional image based approaches. Similarly, Chen et al. (2013) address human activity analysis using depth imagery by reviewing the main published research in this regard. Based on their claim, the review is a good guide in the selection and development of depth-based algorithms, not only for the researchers and practitioners familiar with the topic, but also for newcomers to this field.

- Saini et al. (2013) investigate the various stochastic tracking algorithms in 3D human articulated tracking. They mainly focus on stochastic filtering (PF and APF) and evolutionary optimization algorithms (PSO and QPSO).

- Nunes et al. (2013) present a detailed and broad review of the most cited motion simulation and/or analysis tools, developed both by the scientific communities and commercial entities.

It is out of the scope of this thesis to do a survey similar to the above surveys and review all the published works in this field. Instead, in this chapter we discuss the concepts of different areas related to this research as well as the state-of-the-art approaches in this field, which allow us to identify limitations and propose suitable approaches. In this regard, firstly a compact definition and review of object modelling methods with regard to human body models will be introduced in Section 2.1. Then background subtraction methods (as the first stage of our work) will be described and reviewed in Section 2.2. Finally, Section 2.3 addresses the human body pose estimation and tracking problems.

## 2.1 Modelling

As stated in Chapter 1, the following parameters specify what method should be selected for human motion tracking:

- Object representation.
- Image features.
- Model-based versus model-free.
- Number of viewpoints and cameras.
- Environment.
- 2D versus 3D.

Among these parameters, object representation is one of the most important ones in selecting a specific scenario for human body tracking. It also influences different stages of the tracking system. On this basis, we start this chapter by reviewing the methods of representing the object to be tracked, before dealing with different tracking approaches. Human body models are also discussed in this Section, owing to the fact that the focus of this thesis is on model-based tracking methods.

### 2.1.1 Human Body Representation

Vision-based human body tracking algorithms look for the position and pose of the human body across several successive images. In this way, the target human body, which is a part of the image, should be represented in an efficient way. This is the first step of human motion analysis. Moeslund and Granum (2001) call it initialization.

The image by itself has plenty of redundant information. If the target object shares this redundancy, it increases the computational cost of tracking. To overcome this inadequacy, the human body target object inside the acquired images is described only with meaningful information extracted from the image as follows (Fig. 2.1):

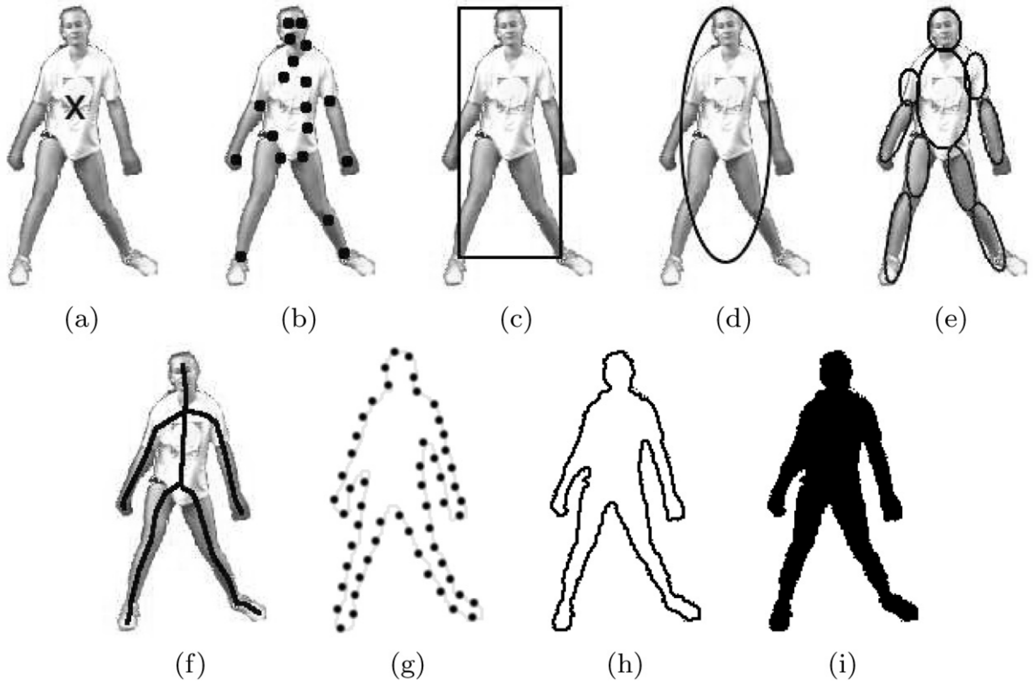


FIGURE 2.1: Object representations. (a) Centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) control points on object contour, (h) complete object contour, (i) object silhouette, copied from (Yilmaz et al., 2006).

- **Points**

  As the most lightweight form of object representation (Aanæs et al., 2012), IPs represent well-defined features of the image such as corners. They provide a high level of descriptive power and a robustness to illumination changes (Leutenegger et al., 2011). These strengths make them superior to other object representation methods, such as Kernel and

Silhouette (will be defined in the following pages), in terms of speed, accuracy and robustness (Yilmaz et al., 2006). Moreover, tracking the objects represented by IPs avoids tracking them as a whole, but instead requires only tracking interesting distinguishable points of the image (Saunier and Sayed, 2006). The object can be represented only by a single point, (e.g. the centroid) when the target occupies a small region of the image, or by a set of points, when it occupies a larger region of the image (Yilmaz et al., 2006).

Different IP detectors and feature descriptors have been proposed up to now. Although there is no clear definition about what should be considered as an interesting point in an image, the definition proposed by Shi and Tomasi (1994) states “the right features are exactly those that make the tracker work best”. This implicitly accepts any set of points which are consistent. In other words, the same points should be detected by point detection algorithm from different images that show the same scene. Some of the well-known interest point detectors are (Gauglitz et al., 2011):

- Harris corner detectors (Derpanis, 2004).
- Shi-Tomasi’s ”Good Features to Track” Difference of Gaussians (DoG) (Shi and Tomasi, 1994).
- Features from Accelerated Segment Test (FAST) (Rosten and Drummond, 2006).
- Centre-Surround Extrema (CenSurE) (Von Hirsch, 1993).
- Scale Invariant Feature Transform (SIFT) (Lowe, 2004).
- Speeded Up Robust Features (SURF) (Bay et al., 2008).
- Gradient Location and Orientation Histogram (GLOH) (Mikolajczyk and Schmid, 2005).
- Oriented FAST and Rotated BRIEF (ORB) (Rublee et al., 2011).

To use the IPs in the tasks such as the IP-matching and IP-based tracking, the local feature descriptors around the IPs should be calculated. As for the interest point detectors, some of the most-used descriptors (Mikolajczyk and Schmid, 2005), which capture the texture of the local neighbourhood and are invariant to changes in illumination, scale and rotation are:

  - Image Patch
  - SIFT.
  - SURF.
  - Key point Classification with Randomized Trees.
  - Key point Classification with Ferns.

  Table 2.1 summarizes and compares different feature points alongside descriptors (Gauglitz et al., 2011). Also, this table shows the existing IP-based visual object tracking systems along with the algorithms they use. The purpose of the table is not to go into detail on specific features and contributions of the listed systems. Instead, it gives an overview of the applications of visual tracking and the algorithms that have been employed based on different IP detectors and descriptors. It could be said that they perform well in the field. However, it is more complicated to compare these algorithms in terms of how well they do because their performance is measured according to different criteria which is depends on the method they use. Thus, sometimes it is not possible to do a comparison between them.

- **Silhouettes and contours**

  A contour defines the boundary curve of an object in the image, whereas a silhouette defines the region inside the extracted contour. These types of image descriptors are used when the objects to be tracked are complex non-rigid shapes particularly against static backgrounds (Poppe, 2007; Yilmaz et al., 2006). In the presence of a background with a different colour to the target object, silhouettes and contours can be extracted effectively, while backgrounds, which are cluttered or changeable with time, make the situation more challenging. These image descriptors have no information about the colour, texture, and depth of the object. Silhouettes and contours have been used for mostly 2D tracking. However, they have sometimes been used for recovering 3D poses (Avidan, 2004). The performance of this type of object representation is highly dependent on shadow and noise in the background as the result of low resolution and changes in illumination.

TABLE 2.1: Existing feature-based visual tracking systems (Gauglitz et al., 2011).

| Reference | Objective/prior knowledge or additional inputs[a] | Detector[b] | Descriptor[b] | Matching, outlier removal, pose estimation[c] |
|---|---|---|---|---|
| Bleser and Stricker (2008) | tracking/3D model, IMU | FAST | patch, warped | SSD, Kalman filter |
| Carrera et al. (2007) | tracking/known target | Harris | SURF | UKF |
| Chekhlov et al. (2007) | SLAM/– | Shi-Tomasi | SIFT-like | UKF |
| Cheng et al. (2006) | odom./stereo, wheel odom., IMU | Harris or sim. | patch | NCC, RANSAC, LSE |
| Davison et al. (2007) | SLAM/initialization target | Shi-Tomasi[d] | patch (11 × 11), warped | NCC, EKF |
| DiVerdi et al. (2008) | panorama creation/– | Shi-Tomasi | Optical flow & SURF[e] | RANSAC (Horn 1987) |
| Eade and Drummond (2006b) | SLAM/– | FAST[d] | patch, warped | NCC, particle filter |
| Klein and Murray (2007) | SLAM/– | FAST | patch (8 × 8), warped | SSD, Tukey M-estimator |
| Lee and Höllerer (2008) | tracking/init. with user's hand | DoG | Optical flow & SIFT[e] | RANSAC, Kalman filter |
| Lepetit and Fua (2006) | tracking-by-det./known target | cf. reference | Randomized Trees | RANSAC, LSE, P-$n$-P |
| Nistér et al. (2004) | odometry/– | Harris | patch (11 × 11) | NCC, RANSAC |
| Özuysal et al. (2007) | tracking-by-det./known target | cf. reference | Ferns | RANSAC |
| Park et al. (2008) | tracking-by-det./known targets | not specified | Ferns | RANSAC, P-$n$-P |
| Se et al. (2002) | SLAM/trinocular camera | DoG | [scale, orientation] | LSE, Kalman filter |
| Skrypnyk and Lowe (2004) | tracking/known scene | DoG | SIFT | RANSAC, non-lin. LSE |
| Taylor et al. (2009) | tracking-by-det./known target | FAST | trained histograms | PROSAC |
| Wagner et al. (2009) | tracking/known targets | FAST[d] | patch & reduced SIFT[e] | NCC, PROSAC, M-estim. |
| Wagner et al. (2010) | panorama creation/– | FAST[d] | patch (8 × 8), warped | NCC, M-estimator |
| Williams et al. (2007) | recovery for SLAM/– | FAST | Randomized lists | JCBB, EKF |

[a] Unless specified otherwise, the system aims to track or recover the full 6 degree-of-freedom pose and uses a single camera as the only sensor. Abbreviations: IMU = inertial measurement unit, SLAM = simultaneous localization and mapping

[b] Abbreviations used in these columns will be explained in Sects. 4.2 and 4.3, respectively

[c] Abbreviations: EKF = Extended Kalman Filter, JCBB = Joint Compatibility Branch-and-Bound (Neira and Tardos 2001), LSE = Least-squares estimation, NCC = normalized cross-correlation, PROSAC = progressive sample consensus (Chum and Matas 2005), P-$n$-P cf. (Moreno-Noguer et al. 2007), RANSAC = random sample consensus (Fischler and Bolles 1981), SSD = sum of squared distances, UKF = Unscented Kalman Filter (Julier and Uhlmann 1997)

[d] Detector is only used to discover new features. Matching is performed against all pixels around the predicted feature position

[e] Hybrid approach: patch/Lucas-Kanade optical flow for frame-to-frame tracking, SIFT/SURF for long-term matching/detection

- **Colours and textures**

  The human body can be seen in different poses and views. However, the appearance (colour and texture) of the human body parts individually remains unchanged. This suggests the idea that the human body can be represented using colour and texture (Poppe, 2007). Gaussian colour distributions (Wren et al., 1997) or colour histograms (Ramanan and Forsyth, 2003) can be used to describe the appearance of the human body parts. Nevertheless, the appearance depends on the clothing, rotation, and changes in the illumination. To make this type of object representation robust with regard to these factors, appearance models can be used (Roberts et al., 2006). Skin colour, as a good cue for finding the not-covered parts of the human body i.e. head and hands, has been a common way of using colours and textures in human motion analysis.

- **Primitive geometric shapes (Kernels)**

  Simple shape primitives such as rectangles and ellipses in 2D or cylinders, spheres, cubes and cones in 3D, or a surface such as polygonal mesh can be used as a generic humanoid model to approximate the subject's shape. Historically, these primitive geometric shapes delivered a coarser representation than silhouettes (Wang et al., 2012). Hence primitives are generally more suitable for representing rigid bodies than non-rigid bodies, which need more precise representation (Yilmaz et al., 2006). For example, Hilton et al. (1999) presented a generic mesh model created from the silhouettes of the front and side views.

  More recently, several approaches have been proposed to make this type of human body representation method more mature in approximating a specific person. In this regard, multiple sensors such as multiple calibrated cameras have been used to capture several simultaneous views of the human body and achieve more accurate shape and appearance (Carranza et al., 2003; Pascal and Plankers, 2003; Starck and Hilton, 2003). Model fitting approaches have been used to provide an accurate parametrized approximation of the human body using the pre-specified shapes for the generic model. However, they could not be general methods because of their dependency on the hair and tight clothing assumption. Nevertheless, some other efforts tried to create a highly detailed representation of the human body shape using 3D scanners (Allen et al., 2002; Starck et al., 2003). As will be seen, kernels can be used in object representation for both the model-based and model-free tracking scenarios.

  Blob representation is another member of the kernel family, which represents the object to be tracked as a blob or a number of blobs each having some similar characteristics. It typically follows some of the figure–ground segmentation approaches (Moeslund and Granum, 2001). Similarity criteria such as coherent flow (Ju et al., 1996), similar colors (Heisele and Woehler, 1998), or both (Bregler, 1997) can be used for blob detection. Blob representation is mostly applicable in model-free human body tracking methods.

- **Depth representation**

Most of the above mentioned object representation methods represent the object of tracking based on the pixel information inside an intensity image. Although intensity images deliver rich information, they have some drawbacks and present object representation methods with some difficulties. They are sensitive to illumination changes, and therefore tasks such as background subtraction are difficult to perform robustly. Moreover, the interest point detectors may be attracted more to the object textures rather than its geometry. In contrast, object representation using depth images has some superiorities over intensity images. Depth images are robust against illumination changes and provide the 3D information of the scene. These advantages can make tasks such as background subtraction, segmentation, and motion estimation easier (Chen et al., 2013).

The occlusion problem, which is one of the recent challenges in this field of research, is hard to solve using the above-mentioned object representation methods. However, depth representation has shown itself to be a good solution to the occlusion problem (Gavrila, 1999). Recovering 3D depth information from images has important applications such as scene understanding and 3D reconstruction (Saxena et al., 2008b). In the case of the silhouette representation, a 3D representation of the scene is constructed from different views of the camera using two common techniques: volume intersection (Bottino and Laurentini, 2001) and voxel-based approaches (Cheung et al., 2003; Mikić et al., 2003).

In terms of depth cues, depth perception methods can be divided into monocular and binocular approaches, which provide the depth information from a single view or two (or more) views respectively, as follows:

- **Binocular perception**: in this category, the disparity and depth information can be extracted from cues such as binocular parallax (Stereopsis[1]), Convergence, and Shadow Stereopsis. In binocular methods, stereo-matching is a common way to compute the depth information from the images of two calibrated cameras. Using triangulation the depth is calculated either in a dense fashion (depth for the whole image), or sparse fashion (depth for some patches of the image, or only for some interesting points). Nearly all stereo correspondence algorithms use a cost function

---

[1] Stereopsis `http://en.wikipedia.org/wiki/Stereopsis`

to measure the disparity of the image locations. Different matching costs, stereo-methods and their evolution have been discussed by Hirschmuller and Scharstein (2007). In spite of the advantages of calculating the depth using stereo-cameras, they have the following drawbacks for depth-perception:

  - Since most existing approaches rely on triangulation for depth estimation, their performance is highly affected by any changes in the baseline distance between the two camera-positions (Olson and Abi-Rached, 2010; Saxena et al., 2007).
  - Texture-less areas, repetition, occlusion and discontinuity can cause false correspondences (Domínguez-Morales et al., 2012).
  - The accuracy of the disparity calculation and hence the resolution of the depth map are highly dependent on the camera resolution (Munro and Gerdelan, 2009). High resolution images cannot be acquired using cheap USB cameras because of bandwidth limitations.

- **Monocular perception**: numerous monocular cues such as texture variations and gradients, defocus, and colour/haze (Saxena et al., 2008a) make depth estimation possible from a single still image. On this basis, monocular-based depth perception methods have produced promising results. They need no calibration and rectification.

An important issue in the case of both binocular and monocular depth extraction strategies is the regions of image, for which the depth information is calculated. This depends on the object representation method. In the case of kernels and silhouettes, since the whole region of the target object is detected and tracked in each frame, the depth information of the whole image should be calculated in a pixel-wise sense. It is performed either locally by searching in a window with a small number of pixels around the pixel under study or globally in the entire image itself (Domínguez-Morales et al., 2012). This is not efficient, particularly in a real-time system. In contrast, point-wise or patch-wise depth calculation decreases the computational cost.

In terms of 3D modelling and 3D imaging sensors, three dimensional image reconstruction techniques can be categorized as follows (Gomes et al., 2014; Rocha et al., 2014; Sansoni et al., 2009):

- **Structured light sensor:** In this field, a laser combined with camera triangulation are used to extract 3D information. Ferreira et al. (2010); Pinto et al. (2010) present 3D modelling approaches for industrial applications, in which the precision directly depends on the thickness of the laser line and the camera. Furthermore, they need the object or robot to move to reconstruct the 3D. A well-structured light environment is also required. Klimentjew et al. (2010) use a three-dimensional Laser Range Finder (LRF) and a camera. Similarly, the Microsoft Kinect[2] (which has received most of attention recently particularly due to its affordability.) uses a single infra-red pattern to acquire 3D data at a rate of 30 range images per second with low precision and noisy data. Kinect sensor has a high potential due to its capability to extract 3D points cloud plus colour features. Kinect 2 also has a very high resolution at the moment. Kinect has been used extensively for different depth-based applications from background subtraction to segmentation and human body tracking and pose estimation.
- **Stereo Vision:** As we discussed earlier, a category of depth extraction methods use two or more cameras to calculate the disparity and depth using binocular cues. For example, Brandou et al. (2007) presents a 3D reconstruction stereo-based approach that it could be used in a vast range of applications such as visual navigation of robots or 3D games. The main aspects of these approaches are: low cost (can be built using common web-cams), portability, low resolution and low accuracy in an uncontrolled environment, usually real-time acquisition.
- **Time of flight:** This group of 3D modelling approaches emits pulses of light and calculates the round-trip time determine the distances over the surface of the scene. Callieri et al. (2009) propose a system based on time-of-flight laser scanners, mounted beneath a balloon to obtain data of large historical buildings from

---

[2] Kinect: `http://www.microsoft.com/en-us/kinectforwindows/`

above. This system operates mainly over large distances which makes it suitable for scanning large structures under noisy conditions.

- **Why interest points rather than kernels and silhouettes**

  The way in which the target object is represented affects all the steps in object detection and tracking. Different object representation methods (Yilmaz et al., 2006) are briefly compared as follows:

  - Since interest points are only related to the most important and distinguishable information of the object (Saunier and Sayed, 2006) in comparison to kernels and silhouettes, they reduce the computational cost highly.
  - Background subtraction, which carries out segmentation of foreground objects from a stationary background (Paragios and Deriche, 2000), faces a significant difficulty caused by environmental changes in illumination, noise, and shadow. This is more serious in the case of silhouettes and kernels where the whole region of the foreground object is segmented from the background. Although the temporal information computed from a sequence of frames (Elgammal et al., 2002) compensates for the weakness of using only a single frame, it increases computation cost. In contrast, interest points have demonstrated a great improvement in robustness to environmental changes (Gevrekci and Gunturk, 2009), which makes background segmentation more robust to changes in background objects. This addresses one of the crucial needs of gaming applications.
  - In point-based tracking methods the same algorithm can be used for tracking in daylight, twilight or night-time conditions, as well as different scene conditions (Saunier and Sayed, 2006). It is automatic because it selects the most prominent features under the given conditions. In contrast, in the cases of kernel and silhouette-based tracking methods, different types of constraints such as the appearance of the target, the background colour and contents, and the light conditions are imposed on the scene, in order to make tracking easier and more feasible (Yilmaz et al., 2006).

  - The scale, rotation and affine invariant properties of most interest points are another of their superiorities.
  - In terms of the occlusion problem, particularly partial occlusion, interest points behave more efficiently than kernels and silhouettes, because in partial occlusion some of the feature points of the moving object can be observed (Saunier and Sayed, 2006). So, it may overcome the problem and reduce the need for reinitialisation in the case of losing the object during tracking. Although some parts of the kernel and silhouette also would be visible, they are not as individually meaningful as interest points.

### 2.1.2 Human Body Models

In the case of the model-based (or generative) human motion analysis, prior information about the human body is used in tracking (Poppe, 2007). This information helps to solve some ambiguities in movement tracking and makes the model-based approaches more accurate than the model-free methods particularly in occlusion situations. However, it increases their computational cost due to the large number of parameters that have to be estimated (Sigalas et al., 2009). In terms of the model, typically it would describe the kinematic properties of the body along with its shape and appearance. We will discuss these in this section.

- **Kinematic human body models**

  The human body, as an articulated body composed of some rigid limbs connected together by joints, is governed by a mathematical model of mechanical systems, namely the kinematic chain (Reuleaux, 1963). In theory, each joint of the kinematic chain can move in 6 different directions (DOFs), 3 translations in the x, y, and z directions and 3 rotations of roll, yaw and pitch. Owing to the number of body parts, the study of human body movement would be complicated by the large number of DOFs in the body ( 244) (Kuo, 1994). However, in practice kinematic constraints hinder the motion of these independent rigid bodies. They reduce the number of rigid bodies as well as the feasible movements of each joint and so reduce the DOFs to about 34 for a full body (Ning et al., 2004). A pose of the body model is represented by the all DOFs of the body together. Generally, the two types of kinematic models, which have been used, are 2D and 3D.

Although 2D models are more lightweight than the 3D ones from the computational cost point of view, they are more suitable for the case of motion parallel to the image plane and are used mostly for gait analysis (Poppe, 2007). Ning et al. (Ning et al., 2004) use a pictorial model of whole body with 12 DOFs in which each body part is represented by a truncated cone except for the head represented by a sphere. Oh et al (Oh et al., 2011) represent the upper body model by a cardboard cut-out of 6 parts for torso, head, left and right arm, left and right forearm. Huo et al. (Huo et al., 2009b) and (Huo et al., 2009a) use a 2D silhouette model with only three parameters: the x and y coordinates of the model in 2D space and its scale.

In the case of 3D a maximum of three orthogonal rotations (kinematic constraints may reduce them to two or one) can be considered for each joint (Poppe, 2007), whereas in 2D only one rotation per joint is possible. Various kinematic models have been proposed, depending on the application at hand. Gavrila and Davis (Gavrila and Davis, 1996) have proposed a 30 DOF model for the whole body and arms. In comparison, a model of 30 DOFs consisting of 15 cylinders, with a mesh of $4 \times 5$ points has been proposed by Lehment et al. (2010). Li and Kulic (2010) use a 25 DOF skeletal model and an outer shape model of truncated cones for the description of the model surface. In a simpler way, Sigalas et al. (2010) exploit a 4 DOF kinematic model for each arm and one additional DOF for the orientation of the body around the vertical axis. The pose of the user is tracked in a 9 DOF model space.

As can be seen, different numbers of DOFs have been used in different studies. Sometimes around 10 DOFs are used for upper body pose estimation. In contrast, for the estimation of the full-body poses, no less than 50 DOFs (Agarwal and Triggs, 2006) have been utilized. The number of possible poses is very high even in the case of few DOFs. Elimination of the infeasible poses by applying kinematic constraints such as joint angle limits (Deutscher and Reid, 2005), limits on angular velocity and acceleration (Yamamoto and Yagishita, 2000), and non-penetrability of human body are an effective solution to prune the pose space (Sminchisescu and Triggs, 2003).

- **Human body shape models**

  Looking at the shape of the parts of the human body model without considering its kinematics is called a ”human body shape model”. 2D shape models usually consist of 2D primitive geometric shapes. For example, Da Xu and Kemp (2009) propose a 2D connected ellipse model, consisting of 6 ellipses and 3 joints, which is used for human upper body tracking. On the other hand, the 3D models are presented in two different forms:

  - **Volumetric-based:** segments of the model are represented commonly by spheres (O’Rourke and Badler, 1980), cylinders (Sidenbladh et al., 2000) or tapered superquadrics (Kehl and Gool, 2006).
  - **Surface-based:** the whole body is represented by a single surface which consists of a deformed mesh of polygons under the kinematic structure (Barrón and Kakadiaris, 2001).

  Such human body representations have been promising. However, they are too crude and unrealistic, to recover both shape and motion precisely due to using oversimplified primitives such as cylinders or ellipsoids (Deutscher and Reid, 2005). In contrast, new 3D surface estimation methods make it possible to capture time-varying geometry in detail. The model introduced by Plankers and Fua models surfaces based on metaballs (according to Wikipedia ”Metaballs are, in computer graphics, organic-looking n-dimensional objects”.) (Plankers and Fua, 2003), whereas Balan et al. (2007) present a learned SCAPE (Shape Completion and Animation of PEople) which models non-rigid deformation of the human body through combining both pose and shape deformation models (Anguelov et al., 2005) (Fig. 2.2)). These approaches have not involved clothing in modelling the human body, whereas Bălan and Black (2008) have estimated clothes from images using SCAPE.

  The increasing attention on user-centered design (UCD) in the last few years (Abras et al., 2004) has made consideration of the human factors significant and compelling.

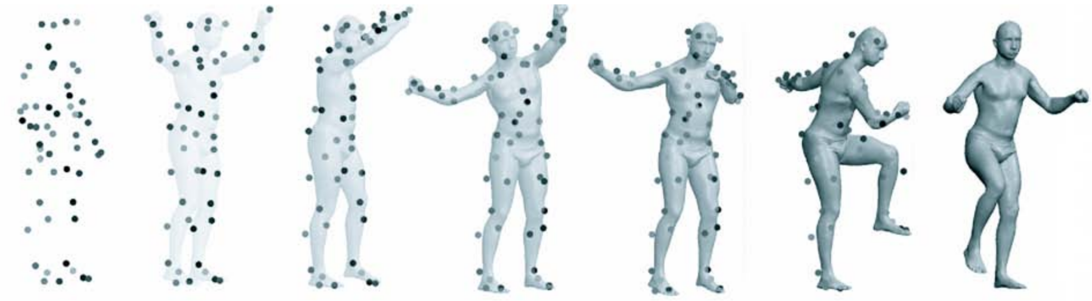


FIGURE 2.2: Animation of a Motion Capture Sequence taken for a single body scan subject. The muscle deformations are synthesized automatically from the space of pose and body shape deformations, copied from (Anguelov et al., 2005).

Consequently, lots of effort has been made to generate accurate and reliable digital human models. The body shape modelling methods which have been widely discussed in the literature can be categorized into Direct model acquisition/creation, Template model-based scaling, Image-based reconstruction, Statistics-based model synthesis (Baek and Lee, 2012).

## 2.2 Background Subtraction

The segmentation of areas of an image related to moving objects (foreground) from the areas related to static objects in the scene (background) is called Background Subtraction when the processed image is captured by static cameras (Herrero and Bescós, 2009). This is the earliest stage of many computer vision applications such as human motion analysis, automated surveillance, video indexing, and vehicle navigation. Therefore, it exerts a strong influence on further processing (Moeslund et al., 2006). Accordingly, a great deal of research has been conducted on BGS over the past few years and many algorithms have been proposed, most of them pixel-based. They rely on the difference between pixels through one of the following ways to model and update the background (Varcheie et al., 2008):

- **Individually**: which uses the pixels' illumination (Stauffer and Grimson, 1999).
- **Regionally**: which uses the texture of a group of pixels in the form of blocks (Fang et al., 2006) or clusters (Bhaskar et al., 2010).

Owing to the importance of BGS (Varona et al., 2008) as the earliest step of algorithms such as tracking, recognition, and behaviour analysis approaches, a huge amount of research has been conducted in this field. Several surveys can be found in the literature which have studied and classified the proposed algorithms from different points of view (Bouwmans, 2011, 2012; Cristani et al., 2010). In terms of image measurement and segmentation methods, background segmentation is divided into *motion-based*, *appearance-based*, *shape-based,* and *depth-data-based* (Moeslund et al., 2006). Motion-based BGS methods (the scope of this chapter) are classified, based on the way they model the background (Bouwmans et al., 2014), into:

- **Traditional models**: which are basic, simple to implement, but have limitations. They are briefly classified into:
  - **Basic Models**: which model the background using techniques such as average (Lee and Hedley, 2002), median (McFarlane and Schofield, 1995) or histogram analysis over time (Zheng et al., 2006).
  - **Statistical Models**: which, for instance, take into account statistically the history of pixel brightness (Gaussian methods (Bouwmans et al., 2008)), or model the background using supervised learning methods such as SVM (Lin et al., 2002) (Support Vector Models).
  - **Cluster Models**: which cluster pixels in each frame using K-means (Butler et al., 1900), Codebooks (Kim et al., 2004), or basic sequential clustering approaches (Xiao et al., 2006).
  - **Neural Networks**: where the background is modelled by a trained neural network.
  - **Estimation Models**: which estimate the background using filters such as Wiener (Toyama et al., 1999), Kalman, and Chebychev (Chang et al., 2004).
- **Recent models**: which are more sophisticated, capable of addressing more complicated challenges, and require improvements to become real-time, are classified into:
  - **Advanced Statistical Background Models**, which for example use new distributions in Mixture Models, or fuse different distributions in Hybrid Models.

  - **Fuzzy Background Models**, which use fuzzy concepts to deal with imprecisions and uncertainties in BGS (Bouwmans, 2012).
  - **Discriminative Subspace Learning Models**, where discriminative methods are used to provide supervised modelling of the background in contrast with the earlier re-constructive subspace learning models (Skocaj et al., 2006).
  - **Robust Subspace Models**, which separate the background and foreground using a robust subspace model based on a low-rank and sparse decomposition.
  - **Sparse Models**, where sparse models such as structure sparsity models (Huang et al., 2011), dynamic group sparsity models (Tibshirani, 1996), and dictionary models (Tropp and Gilbert, 2007) are used.
  - **Transform Domain Models**, where the background and foreground are discriminated in a different domain using different transformation such as Fast Fourier, Discrete Cosine (Porikli and Wren, 2005), Walsh (Tezuka and Nishitani, 2008), Wavelet (Gao et al., 2009), and Hadamard (Baltieri et al., 2010).

Beside the above, the size of the image element in background modelling, which could be a pixel (Stauffer and Grimson, 1999), a block of pixels (Fang et al., 2006), or a cluster of pixels (Bhaskar et al., 2010), is another important issue and determines the precision and robustness to noise. The bigger the size of the element, the higher the precision of the algorithm and the lower its robustness to noise. The type of feature, moreover, is another important factor in BGS, which is classified into (Baltieri et al., 2010):

- spectral features (colour features).
- spatial features (edge features, texture features).
- temporal features (motion features)

## 2.3 Human Body Pose Estimation and Tracking

**Human body tracking** is defined as a way to find the temporal trajectory over the frame sequence of a human body, partially or totally, in a state-space, where the state-space could be 3D Cartesian space or 2D image space. There are several different ways to classify human body tracking methods. Based on whether they use explicitly a prior model or not, human body tracking methods are classified into model-based and model-free methods. Model-based (generative) approaches use an explicit model to perform tracking directly by minimizing the error between the real data and the human body model. In contrast, model-free (discriminative) methods commonly optimize the error between the observations and a projection function (in the learning-based case) or example set (in the example-based case) (Poppe, 2007; Sigalas et al., 2009). Moeslund et al. (2011) categorize all the pose estimation and tracking methods into four major classes: generative-model-based approaches, discriminative-model-based approaches, part-based-model approaches, and geometric approaches.

**Pose estimation** is defined as the process of finding the set of pose parameters which minimizes an error value between the real observation of the frame and a projection function. Pose estimation is defined for both the model-based and model-free approaches. The projection function is the projection of the human body model in the model-based approaches, whereas in the model-free approaches it is a function obtained from either the learning sets (in learning-based approaches) or example set (in example-based methods) (Poppe, 2007). *In model-based methods, pose estimation is in fact an internal part of the tracking process* (Moeslund et al., 2006). Owing to the scope of this thesis, we leave the other categories here and only discuss the model-based human body tracking concepts and approaches.

### 2.3.1 Top-down Versus Bottom-up

Estimating the pose parameters in the model-based pose estimation and tracking methods can be performed through two different strategies: Top-down and Bottom-up. In the *Top-down* strategy the whole body is found first, then the parts are estimated from the body. In contrast, the Bottom-up strategy finds the parts of the body first, then it assembles them to match the whole body. Combining the pure top-down and bottom-up methods yields the combined top-down and bottom-up estimation. In the following, we review the top-down and bottom-up methods briefly.

- **Top-down approach** This type of pose estimation and tracking method involves analysis-by-synthesis to match the projected human body model to the image observations. In these approaches, an initial estimation is performed by means of prediction. Then, refinement is carried out through a local search around the predicted pose for each frame. At the start, manual pose initialization is performed, which is an obvious drawback of this pose estimation method. For the subsequent frames, the estimated pose in any frame of the video acts as the initial pose for the next frame. Obviously, projection of the human body model over the image as well as calculating the distance between projected model and real data requires a high level of computational cost. This is another drawback of this method (Poppe, 2007).

  Ning et al. (2004) present a hierarchical divide-and-conquer technique for kinematic-based 2D tracking of a walking human in a monocular video sequence. The pose estimation is initiated by roughly predicting the torso position from the silhouette centre of gravity and then refining it using physical forces. The other limbs of the model are hierarchically estimated similarly by minimizing the matching error based on boundary and region information. A similar approach has been proposed by Muhlbauer et al. (2008). The proposed model-based algorithm, which estimates the body poses using stereo vision, uses point clouds and a 27 DOFs model. Using the skin colour information, the point clouds of the head are matched to the head part of the model. Then the algorithm tries to fit the attached links iteratively based on the assumption that the start point of any link will be the end point of the previous link. The link's end point will be searched somewhere near this reference point.

  The top–down tracking approaches often have problems with (self-) occlusions. An inaccurate estimation for the any part such as torso or head propagates the error onto the estimation of the lower body parts in the kinematic tree. Imposing constraints between linked body parts in the kinematic chain has been proposed as the solution by Drummond and Cipolla (2001). So the lower parts can affect the higher parts in the chain.

- **Bottom-up approach** In contrast, the bottom-up approaches find the human body by estimating the body parts and assembling them into a whole body. They need limb detectors for most body parts, which is a drawback of these methods because of the False Positives (FP) and False Negatives (FN) due to wrong detection and missing information respectively (Poppe, 2007). However, physical constraints such as body part proximity improve the assembling process. Temporal constraints also are used to solve the occlusion problems. Obviously, that there is no need for manual initialization, is the main advantage of these tracking methods.

  Felzenszwalb and Huttenlocher (2005) model body parts and the coherence among them using 2D appearance models. The optimal solution for body parts in the tree of body configurations is found using a dynamic programming algorithm. Lan and Huttenlocher (2005) extend trees using the correlations between body parts. For example, correlations between upper arm and leg swings have created more robustness in walking pose estimations. Ronfard et al. (2002) have used pictorial structure along with more sophisticated methods such as Support Vector Machines (SVM) for learning the appearance of the body parts. Ronfard et al. (2002) have reduced the motion tracking to the problem of inference in a Dynamic Bayesian Net using simple appearance-based part detectors.

  Owing to the pros and cons of the top-down and bottom-up methods, more robust methods have been achieved by combining them. For example, Navaratnam et al. (2005) have proposed a search-space decomposition approach to find the lower body part of the kinematic chain within the image region using detectors.

### 2.3.2 Single-hypothesis Versus Multiple-hypothesis

Traditional tracking approaches generate and maintain a single hypothesis as the solution for the pose state over the frame sequence. Although efficient in terms of computational cost, this often causes loss of track. In contrast, more recent works focus on having several hypothesis over time. In this sense, human body pose estimation and tracking approaches are categorized into the *Single-hypothesis* approaches and *Multiple-hypothesis* approaches as follows:

- **Single-hypothesis approach** Single-hypothesis approaches, including simple Kalman filtering and local-optimization methods (Bregler et al., 2004; del Rincón et al., 2011; Kakadiaris and Metaxas, 1998), are relatively old-fashioned pose estimation and tracking approaches which generate and maintain only one solution over the frames. Thus, in ambiguous situations such as occlusion, there is little chance of preventing selecting the wrong pose and losing track. They also face error accumulation, and drift problems, in both of which cases it is hard to recover from the wrong pose and loss of track will happen.

- **multiple-hypothesis approach**

  To overcome the mentioned problems in single-hypothesis methods, multiple solutions are generated and maintained over time. For example, a set of Kalman filters, instead of a single simple Kalman filter, can be used simultaneously to propagate multiple hypotheses (Cham and Rehg, 1999). This helps to recover the pose in situations where a single hypothesis fails. Although it outperforms a single Kalman filter, it still has problems in estimating the motion of the human body, which is a non-linear process due to its joint accelerations.

  Sampling-based approaches such as Particle Filtering (PF) (Deutscher and Reid, 2005) are able to do pose estimation and tracking in non-linear situations. Basically they are multiple-hypothesis approaches, which generate and propagate a number of particles using a dynamic model with a noise component. A weight coefficient is associated to each particle, which is updated iteratively based on a cost function. The particles that generate lower cost will have higher weights. The final estimated pose would be summation of the particles based on their weights. In this fashion, a combination of multiple hypotheses generates the estimated pose.

  As can be seen, the sampling-based methods appear a very promising solution for the pose estimation and tracking. However, they need a large number of particles, due to the high-dimensionality of human body pose space, to span the pose space sufficiently. It increases the computational cost of these dramatically (Arulampalam et al., 2002) because the number of particles required increases exponentially with dimensionality (Moeslund et al., 2006). Another problem that particle filtering methods face, is the fact that particles tend to gather in a very small area of the pose space, which is called sample impoverishment (King and Forsyth, 2000). This problem decreases the number of effective particles. To solve these problems of the particle filtering approaches, different particle sampling schemes such as partitioned sampling of the state space (MacCormick and Isard, 2000), Annealed Particle Filtering (Deutscher et al., 2000) have been proposed.

### 2.3.3 Stochastic versus Heuristic

In one sense, human body pose estimation and tracking approaches can be compared in terms of whether they are stochastic or heuristic. This comparison makes sense because of the concept of a ”particle”, which is used in both stochastic and heuristic methods. Although this may lead the reader mistakenly to think that they are same, there are substantial differences in their nature.

- **Stochastic approaches**:

  Among the various stochastic filtering algorithms and their extended variants, Particle filters and Annealed Particle Filter (APF) are the algorithms which have been extensively used for articulated human body tracking (Saini et al., 2013). In these methods, filtering is the process of estimating the state of a statistical model according to the measured observations. Given the previous state before the measurement, a state estimation is performed by predicting the current state and then correcting it according to measurements. The widely used filtering methods, Kalman and particle filters and their variants, have been described earlier in this Chapter.

- **Heuristic approaches**: The performance of stochastic filtering algorithms in highly non-linear human body pose estimation and tracking problems is limited. They need a big population of particles which leads to huge computational cost, particularly in a high-dimensional pose space. The requirements of human body pose estimation and tracking include:

1. Handling anatomical constraints.
2. Modelling both the observations and the human body.
3. Handling pose estimation and tracking simultaneously.

It is not easy to meet these requirements using stochastic Bayesian approaches (which range from classic Kalman filters (Bregler et al., 2004; Mikić et al., 2003) to particle filters (Deutscher and Reid, 2005; Fablet and Black, 2002)).

In contrast, stochastic evolutionary optimization algorithms such as Ant Colony Optimization (ACO), particle swarm optimization (PSO), genetic algorithms (GA) and their variants have shown a great deal of success at overcoming the above mentioned difficulties. Among the evolutionary methods, PSO in particular, has been applied to different problems in object pose estimation and tracking. In the last few years, PSO has been becoming popular in computer vision applications due to its simplicity and capability of searching for the global optimum of hard non-linear problems. Moreover, PSO has a parallel nature and is able to handle the anatomical constraints straightforwardly. It has no need for any statistical observation modelling process, which reduces its dependency on a high prior knowledge of the optimisation problem (Robertson and Trucco, 2006). Owing to these advantages, PSO is the most commonly used for human motion pose estimation and tracking among the evolutionary methods.

Among the several proposed approaches to human body pose estimation and tracking using PSO, Robertson and Trucco (2006) present a multi-view human upper body pose estimation approach. They fit a skeleton model hierarchically to 3D point cloud stereo data captured from an array of cameras without using a prior motion model. Ivekovič et al. (2008) address the problem of human body pose estimation from multiple-view still images of a person sitting at a table. They formulate the human upper body pose estimation problem as an analysis-by-synthesis optimisation algorithm using PSO. They fit a generic 3-D human body model to the silhouettes extracted from the images. Oikonomidis et al. (2012) propose a model-based method based on PSO to track the full articulation of two unconstrained interacting hands with each other, which is the first and probably the best approach in this regard.

### 2.3.4 IP-Based Object Tracking

Owing to the efficiency of the IP representation, many IP-based object tracking approaches have been proposed recently (Gauglitz et al., 2011). The proposed IP-based object tracking algorithms, which perform tracking as shown in the block diagram in Fig. 2.3 (Gauglitz et al., 2011), are categorized into model-based tracking versus the model-free tracking algorithms, according to the categorization presented by Wang et al. (2003a). The concept of this classification is different from the definition of model-based and model-free human body tracking algorithms mentioned in Section 2.3. To understand this difference and the concept of the classification within the IP-based object tracking methods, we review one cycle of such a typical IP-based object tracking algorithm as follows (without loss of generality):

1. Given a new frame of the video as the current frame, an IP detector is applied to the frame to detect the potential IPs for tracking.
2. A feature descriptor is formed for each of the detected IPs. As stated in Section 2.1.1, this descriptor represents a local neighbourhood around the IPs and is used to find the matched IPs to IPs of the current frame.
3. A matching algorithm is then applied to the detected IPs of the current frame and either the IPs of the previous frame or the IPs of a reference frame. The correspondence between the two IP sets is then used to track the position of the object in the current frame.

The last step is the distinguishing point between the model-based and model-free IP-based object tracking methods. If the IPs of a reference frame are used to search the subsequent

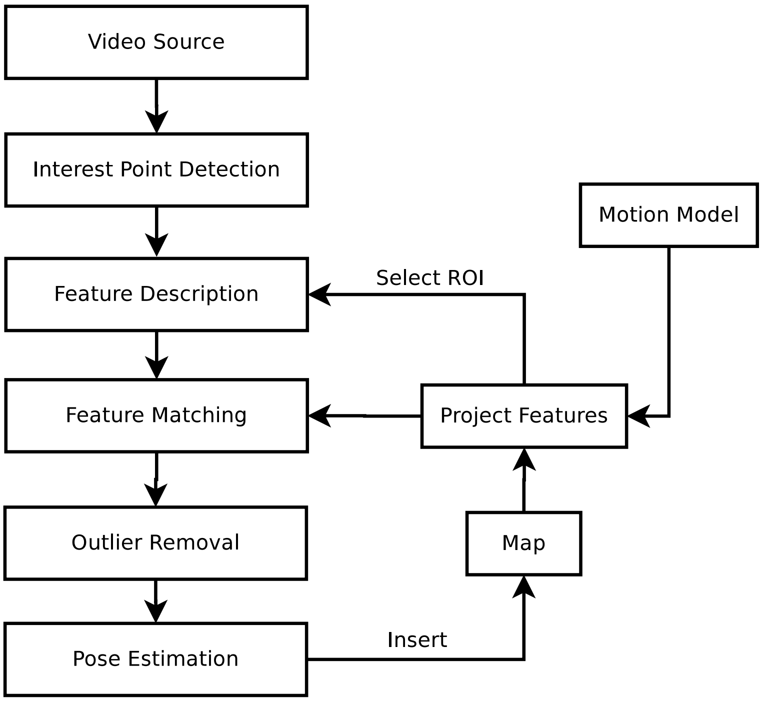


FIGURE 2.3: detector-descriptor-based visual tracking system (Gauglitz et al., 2011).

frames pixel-wise by doing a local search inside a proper-sized window to find the positions with the best similarity according to their descriptors, that is a model-free approach. The method introduced by Lucas and Kanade (1981) along with its improved versions (Shi and Tomasi, 1994; Tomasi and Kanade, 1991) is one of the most famous IP-based trackers in this category. This case is desirable when the cost of descriptor calculation for all the pixels is small and therefore the IP detection step is avoided and instead the matching is performed over all pixels inside the predicted search window. Active search (Davison et al., 2007) and gated search (Klein and Murray, 2008) are two of the search methods.

In contrast, in model-based tracking algorithms the same confident IPs for each frame are selected. Then correspondences between both groups of IPs are established using a prior model. In this case, the IP detector is used to reduce the field of potential candidates and decrease the cost of descriptor calculation. Usually, the number of matched IP pairs is much greater than the degrees of freedom that we need to estimate. Thus the system is highly overdetermined in this case. This fact is used for data cleaning to remove the outliers before being used for pose estimation (Gauglitz et al., 2011). Ma et al. (2013) present robust algorithms for non-rigid point set registration by calculating the transformation between the sets iteratively. Ma et al. (2014) present a similar solution in the Tikhonov regularization problem framework.

A great deal of research has been accomplished on IP-based object tracking (Gauglitz et al., 2011; Gil et al., 2014; Wang, 2006). Table 2.1 summarizes some of the seminal methods in this regard. He et al. (2009) present a motion-based tracking approach, where the object, which can be a human body, is represented by a set of SURF local invariant features, and its motion is tracked by a feature correspondence process. A SIFT-based mean shift algorithm is presented for object tracking by Zhou et al. (2009). However, pose estimation and tracking on non-rigid articulated human body using IPs is a harder task due to the difficulty we face in identification of the subject feature points from the inter-frame tracking information. It requires interest point identification which is the determination of the correspondence between the interest points and model across the frames (Li et al., 2008a). The task addresses the difficult problem of pose recovery over the frames, which is much harder in the monocular situations where we face more ambiguity.

On the other hand, the IP-based tracking methods suffer from the limitation of the unavailability of a sufficient number of matched IPs in all frames of the video sequence, particularly in the case of the articulated human body. (Gupta et al., 2013) present an IP-based human body tracking approach using a dynamic model, consisting of a set of evolving SURF descriptors over time, for modelling the changes of the human body pose and motion throughout the frames. The proposed approach focuses only on modelling the foreground object of tracking, which reduces the computational cost due to obviating the need for learning a classifier separately. In contrast, there are approaches which take into account both the foreground and background areas in the modelling process (Gu et al., 2011; Haner and Gu, 2010; Zhou et al., 2009). Ta et al. (2009) and Zhou et al. (2009) learn the actual motion model of the object, while Bing et al. (2010) create a bag-of-words through clustering for modelling the object of tracking.

## 2.4 Conclusions

Articulated human body pose estimation and tracking, from either a video stream or a set of still images representing the first frame of a video sequence, is an important task in many research topics such as surveillance, motion capture, human gait analysis, medical analysis, sign language recognition and so on. As we reviewed in this chapter, a large body of approaches has been presented on this topic focusing on different combinations of observation representation, matching, and pose estimation. However, vision-based articulated human body pose estimation and tracking is still an unsolved problem with some open ill-posed difficulties due to the high dimensionality, complex motions, occlusion and so on (Moeslund et al., 2006). It could be argued that Kinect has solved this problem. However, it is not applied to all situations, because:

- It is too large to be embedded in the mobile and wearable devices.
- It only works over a range of 1-3 meter whereas an IP-based solution could, in principle, work over any range.
- An IP-based solution such as the system proposed in this thesis, could be implemented on a wearable device with small size (e.g. $3 \times 1.5cm^2$) and very low power consumption. See the Eyes of Things (EoT)[3] project.

Owing to the efficiency of the IP representation, lots of IP-based object tracking approaches have been proposed recently. They perform human body pose estimation and tracking through the model-based tracking and the model-free scenarios. On the other hand, the use of evolutionary optimization methods has been applied to the task of human body pose estimation and tracking recently due to its simplicity and ability in solving complex highly non-linear high-dimensional optimisation problems and finding the global optimum.

Particle swarm optimization, which is a nature-inspired population-based meta-heuristics algorithm (Beheshti and Shamsudding, 2013), has been gaining popularity in the computer

---

[3] Eyes of Things `http://eyesofthings.eu/?page_id=228`

vision application domain (Saini et al., 2013). Different types of observation representation methods along with PSO have been investigated for these tasks. Ivekovič et al. (2008) and John et al. (2010) use silhouette as the image observations. Oikonomidis et al. (2012), in contrast, use depth data for hand pose tracking using PSO. Despite the fact that only 9% of PSO applications are devoted to image and video analysis applications Poli (2007), the success achieved by PSO-based approaches signals that there is yet high-potential to be investigated.

Owing to these, we came up with the idea to investigate interest points as the most light-weight method of object representation in the task of human body pose estimation and tracking using PSO. One of the main differences of our method with the others is in the way we compare the particle solutions of PSO with real observations. Most other approaches generate synthetic observation data independently from the observations in the previous and current frames to compare them with the real observation. This is not possible in our method for the interest points because IPs come from the appearance and texture of the object as well as the scene. Thus, we generate the synthetic data using the relationship between the images and observation in the previous and current frames. This is probably the *central contribution* of our work in this research.

**Summary:** Based on this literature review we decided to investigate IPs in the task of model-based human body pose estimation and tracking in a PSO framework. As our idea depends on the foreground IPs of the frames as the input, an IP-based background subtraction algorithm has to be performed in the first stage of system. To provide the requisite information for the PSO-based pose estimation of the third stage, a robust IP-matching algorithm had to be developed and deployed in this work. The matched IP pairs of the IP-matching stage act as the input for the pose estimation stage of the system. Through a combined bottom-up top-down pose estimation strategy, a two-stage hierarchical-global model-based approach is be presented to infer the pose in the current frame based on the pose of previous frame as well as the matched IP pairs between the frames. Fig. 1.2 shows the framework of our human body pose estimation and tracking system.

# Chapter 3

# IP-Based Background Subtraction

> *The inspiration you seek is already within you, be silent and listen.*
>
> ---
>
> RUMI, 13TH-CENTURY PERSIAN POET.

A great deal of research has been conducted on background subtraction over the past few years and many algorithms have been proposed, most of them pixel-based. They rely on the difference between pixels either *individually* (Stauffer and Grimson, 1999), using the pixels' illumination, or *regionally*, using the texture of a group of pixels in the form of blocks (Fang et al., 2006) or clusters (Bhaskar et al., 2010), to model and update the background (Varcheie et al., 2008). In contrast, IPs, as the most lightweight way of object representation (Aanæs et al., 2012), have not so far been used for BGS.

In contrast to the above *pixel-based* approaches, we propose an *IP-based* BGS algorithm in this chapter, applicable in IP-based CV applications. At this stage, we do not have any specific quantitative requirements for this system. Because this is a novel approach, our motive is simply to demonstrate its feasibility and measure its accuracy and efficiency in comparison to existing pixel-based approaches. The main *contributions* of our proposed approach lie in its difference from pixel-based algorithms in the following respects:

- The number of IPs is much less than the number of pixels in the image. So they reduce the computational cost greatly. Some people might object that, due to the fact that all the pixels must be processed during the IP detection task, the proposed approach must deal with all the pixels, just as the pixel-wise approaches do, and therefore it would not be superior. To answer this, we would mention that, if the background subtraction were the ultimate goal of the system, the criticism would be justifiable and there would be no superiority of the IP-based over the pixel-based approaches in terms of the number of individual entities which need to be processed. However, if the system is an IP-based CV system which is intended to do subsequent additional processing on the IPs, then we would need to run IP detection anyway. Thus, it is better to avoid visiting all the pixels for the background subtraction purpose and instead work on the IPs using an IP-based approach. The foreground IPs then constitute the data which will be subsequently processed by the system.
- IP-based BGS methods are robust to illumination changes due to the robustness of IPs to environmental changes (Leutenegger et al., 2011). In contrast, pixel-based BGS methods face a significant difficulty regarding illumination changes. Although they compensate for the weakness of a single frame by using the temporal information computed from a sequence of frames (Elgammal et al., 2002), it increases the computation cost by itself.
- IP-based BGS approaches can work widely in daylight, twilight or night-time as well as different scene conditions. In contrast, the pixel-based BGS methods need to impose constraints such as the background colour, background contents, and the environmental light conditions in order to make BGS more feasible (Yilmaz et al., 2006).
- IP-based BGS approach delivers more information to CV applications than pixel-based approaches. To put it another way, some of the foreground IPs of the moving object can still be observed during partial occlusion (Musa and Watada, 2008). This may be used to overcome the occlusion problems in tracking applications. In comparison, although some pixels of the foreground may also be visible in the pixel-based approaches, they are not as individually meaningful as the IPs.

- As we discussed in Chapter 2, IPs are the core of many modern CV algorithms and therefore it makes sense to extract them in a preprocessing step. and introduce a BGS algorithm, which is immediately based on IPs. However, there are some disadvantages. The drawback of this approach is that it does not deliver the silhouette, which some systems require as input. Nevertheless, it would be possible to generate a rough silhouette from the foreground IPs.

Another group of background subtraction algorithms, which are more similar to our IP-based method than to the pixel-based approaches, are the edge-based background subtraction methods. These methods rely on edges, which contain the most informative pixels in the images and are less sensitive to intensity changes and noise, just as IPs are (Kim et al., 2013), though they have less stability and information than IPs. Edge-based and IP-based methods provide similar capability of designing more accurate and robust algorithms with similar computational time as they work with fewer pixels. Despite this similarity between edges and IPs, have a different appearance and do not coincide with each other, which makes their comparison somewhat infeasible.

This chapter represents a novel IP-based background subtraction approach which is the first stage of our model-based human upper body tracking system (the **blue** block in Fig. 1.2). Firstly, our motivation for introducing a new BGS algorithm based on IPs as well as the idea behind it are discussed in Section 3.1. Then, the algorithm is described technically in Section 3.2. To evaluate the performance of this algorithm and also to compare it with the other BGS approaches, the results of several experiments are represented afterwards in Section 3.3. Finally, conclusions and future work are discussed in the last section of this chapter.

## 3.1 Motivation

All the objects in the frames of a video are divided into two groups: the background objects (static objects in consecutive frames); and the foreground objects (ones moving around the image throughout the frames). This fact inspires us to propose a novel IP-based BGS approach, which is used to discriminate the moving IPs from the stationary ones (IP-based BGS). On this basis, the extracted IPs from the background areas of the image should remain stationary throughout the frames in contrast to the moving IPs of the foreground, which move around the image. To discriminate these IPs from each other, their position and number of appearances in any location should be recorded and analysed. In future work, we may include additional features e.g. the strength or descriptors of the IPs (see p.70). Although this seems to imply that all the frames should be processed to decide, if any IP belongs to the background IPs group, this is not the case in practice. As will be shown in Section 3.3, the background IPs need to stay motionless only for a specific number of consecutive frames, not necessarily all the frames. More importantly, this enables the algorithm to update the background model following changes (e.g., when something is added to or removed from the scene).

To fulfil this idea, suppose an index $I(x,y)$ is assigned to any location $(x,y)$ of the image plane. $I(x,y)$ is defined as the number of times IPs have appeared at this location. For any IP $i$ of frame $k$ with coordinate $(x_i, y_i)$, the index $I(x_i, y_i)$ is increased by one. This means that an IP has located one more time at that position of image. In this way, the index $I(x,y)$ records and updates the number of times the location $(x,y)$ of image has met IPs. Since the background IPs are extracted from the stationary areas of image, it follows that they hit the specific locations of image repeatedly over the frames and so those locations have a higher value of $I$. In contrast, the foreground IPs are moving around the image throughout the frames. Therefore, their corresponding locations $(x,y)$ have lower values of $I(x,y)$. It might seem that the foreground IPs may appear like the background IPs, when they try to stand still. However, it rarely happens because they have enough movement to cause changes in the location of their extracted IPs.

Although this idea is simple enough and seems to be efficient, it turns out to be too severe and suffers from the problem of spurious background IPs which are actually foreground IPs. This happens when a location in the image is occupied repeatedly by the foreground IPs of different parts of a foreground object, as it moves around the image and hits that specific location. This location looks like a background location because of having a high number for its $I(x,y)$, whereas it is spurious and should be avoided effectively. To overcome this problem, the neighbouring locations in the image are dealt with together as a group using a **block-wise** processing approach, which divides the image into blocks of the same size. On this basis, the combination of locations in a block, hit by IPs inside that block, are considered together as an **Event**. In other words, an Event is a simultaneous observation of a group of IPs, a specific pattern, inside a block.

Accordingly, instead of counting the number of times a location is met by any IP, the number of times a combination of locations (an Event) is met inside a block is recorded. This strategy guaranties that only the specific persistent Events inside the blocks, those with a high level of repetition, are recognized as the static background IPs. The foreground IPs are unlikely to create exactly these patterns. Therefore, counting the number of repetitions of the Events instead of the individual IPs across the frames gives us an accurate interpretation of the location of the real background IPs. Obviously, if there are no IPs in a block, that block would have no events and neither FG nor BG IPs. Fig. 3.4 shows the repetition of Events for a block of an image, for example.

Based on this definition, the blocks of the image are categorized into three different types:

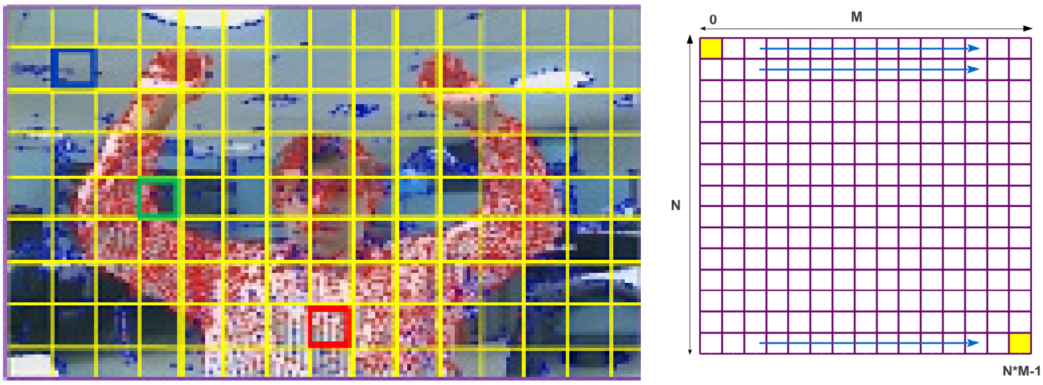
**Background blocks:** These are the blocks which are completely occupied by the background IPs (areas of the image with no moving objects). The numbers of repetitions of the Events in these blocks are significantly greater than of Events in the other blocks. So, these Events are labelled as *dominant* Events. It is expected that the Events inside these blocks are seen in nearly all the frames, approximately. Nevertheless, the number of IPs and their position in each block can be changed because of some phenomena such as changes in illumination, noise in the image, image resolution, and weakness of the IP detector.

**Foreground blocks:** These blocks, areas where moving objects are present, are occupied by the foreground IPs. The IPs of these blocks are unlikely to create the same Events throughout the frames. This implies that the foreground IPs increasingly create new Events for their associated blocks, which cannot appear dominantly. The proposed block-wise processing strategy ensures that they are less likely to create spurious background Events.

**Background-Foreground blocks:** These are the blocks occupied by the outer boundary of the foreground object. Therefore, some IPs in these blocks belong to the background while the others belong to the foreground. Although it may seem the background IPs of these blocks could create dominant Events, it will not happen here because either some background IPs have been occluded or the foreground has added some new IPs, which means a new Event has been created. Consequently, new Events are added to the record of each block, which will never be dominant from the repetition point of view because they do not persist long enough. Although this type of block may look useless during the BGS process, they can be used for further processing due to the valuable information they have about the outer edges of the foreground object.

**Summary of the idea:** According to the concept of motion detection, which is all about determining if a pixel (an IP here) is associated to a moving object at time $t$, the proposed algorithm detects the moving IPs by classifying all the IPs into background and foreground ones. When the algorithm is run, it starts to model the background by finding the dominant Events of all the blocks, which are used later on to determine the location of the background IPs. In a sense, the whole image is background. If there are no foreground objects in the scene, i.e. everything is still, all the background is visible. In contrast, some areas of the background would be invisible in circumstances where a moving object (foreground) is present in the scene. The background subtraction algorithms needs to model and update the whole background. Owing to this, the background blocks here are apparently the blocks which are not covered by foreground objects and the foreground ones are in fact the blocks covered by the foreground objects.

In this regard, the background modelling exhibits two different cases. The first case is when there is no foreground object in the scene and so all the background blocks are visible. In this case, the algorithm learns the dominant Events for all the blocks quickly in less than a few tens of the frames. In comparison, the second case is when there are some foreground blocks or covered background blocks due to the presence of foreground objects. In this case, the algorithm learns the dominant Events only for the uncovered background blocks. The other covered background blocks will remain unlearned until they will become uncovered. At this time, the algorithm starts to learn for these blocks similarly through a few tens of the frames. On the one hand, this process is the background modelling process. On the other hand, the model update process renews the status of dominant Events of the blocks when any change happens in the background. Fig. 3.1 shows an image with its foreground IPs and background IPs, in red and blue, respectively. The three different types of block can be seen in blue, red and green, accordingly.



(a) Image, IPs, and blocks; blue: **B**ackground block, red: **F**oreground block, green: **B**ackground-**F**oreground block.

(b) Structure of a block.

FIGURE 3.1: The block and its different types.

## 3.2 IP-Based BGS Algorithm

In this section, first the proposed BGS approach is described technically. Then, the post-processing tasks which are applied to amend the result and reduce the errors are presented.

### 3.2.1 Technical Description of the Algorithm

For the purpose of block-wise processing, the image of size $H \times W$ is divided into blocks of $N \times M$ pixels (Fig. 3.1).In terms of hardware implementation and efficiency, it would be better to select $N$ and $M$ as a power of 2 because the memory blocks are counted based on powers of 2. Fig. 3.1 shows a frame of $240 \times 320$ pixels and its blocks in yellow as well as the structure of a block. Then the extracted IPs of any frame are assigned to appropriate blocks based on their global coordinates $(x, y)$ in the image plane. Obviously, the blocks related to the smoother areas of the image have fewer IPs (sometimes no IP), and those associated with the highly textured areas of the image have higher numbers of IPs. Afterwards, local coordinates $(r, c)$ are calculated for each IP using its global coordinates in the image plane and the block size as shown in Eq. 3.1. As an instance, for an IP with global coordinate $(155, 85)$ and block size of $15 \times 15$, its local coordinate would be $(6, 11)$.

$$r = (y \bmod N) + 1 \quad , \quad c = (x \bmod M) + 1 \tag{3.1}$$

To store the Events of each block through consecutive frames, firstly we need to define a unique tag for any Event using its constituent IPs. This tag is composed of some numbers, corresponding to the local location of the IPs in the block, separated by commas e.g. $\{9, 21, 39\}$. To make the implementation easier, firstly the 2D local coordinates of pixels in the block are mapped into a 1D coordinate by numbering pixels from 0 at the top left corner of the block, and then counting along each row from left to right to $N \times M$-*1* at the bottom right corner (Fig. 3.1(b)). For example, for the block of Fig. 3.2 with $15 \times 15$ pixels, numbering is started from 0 at the top left corner and is terminated at 224 at the bottom right corner. If this block has 5 IPs at the positions $(2, 2), (3, 6), (5, 8), (9, 13), (11, 4)$ in frame $k$, locations 16, 35, 67, 132, 153 in the block are occupied by the IPs, respectively. By concatenating these numbers to each other by commas, the event $\{16, 35, 67, 132, 153\}$ is created for the mentioned IPs in the block. Finally, a unique tag is created for any Event using this terminology.

Now, the Events of each block as well as their number of repetitions should be stored and renewed at any frame using these tags. This process is accomplished through two different operations:

- **Storage**, which means storing a new Event for the block, if the underlying tagged Event is a new one and has not happened so far.

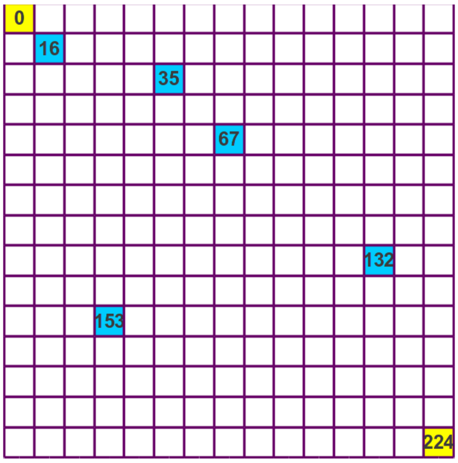


FIGURE 3.2: A sample Event $\{16, 35, 67, 132, 153\}$ for IPs $(2,2), (3,6), (5,8), (9,13), (11,4)$ in a block with size $15 \times 15$ pixels.

- **Update**, when the Event has been seen in this block in the previous frames at least once, its number of repetitions should be increased by one.

As one of the main motivations of our proposed BGS approach is to preserve the speed and efficiency as much as possible, an efficient way of storage, sorting, and searching is important here.

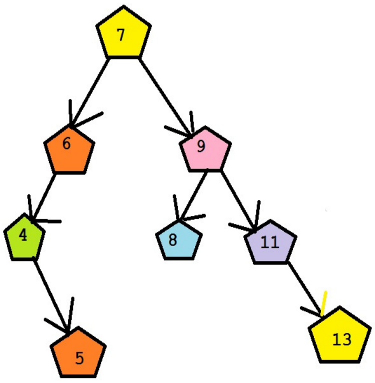


FIGURE 3.3: A binary search tree: the larger inputs feed into the right sub-trees and the smaller ones feed into the left sub-trees.

The Binary Search Tree (BST) (Gilberg and Forouzan, 2001), which is a fast way of storing, sorting and searching, is used in this regard. It stores the values in a tree-shaped graph. To do this, the larger inputs go into the right sub-trees and the smaller ones go into the left sub-trees (Fig 3.3). This strategy makes storage, sort, and search faster. On this basis, a

BST is created for each block and the assigned Events to this block throughout the frames are stored in the tree as well as the numbers of their occurrences (Repetition Index (*RI*)) by means of the mentioned tags. If any Event is happening for the first time, a new node with *RI* of 1 is created in the tree. Otherwise, the node corresponding to the Event is found and its *RI* is increased by 1. So, the BST of each block summarizes the Events and their *RI*s in that block.

As stated before, the background IPs are supposed to be motionless. This implies that they should appear in specific locations in the blocks and therefore specific Events should be created by them in the blocks. Regardless of the rare changes in the position of these IPs in the blocks, due to changes in illumination and low resolution, they appear significantly more often than a threshold criterion. In contrast, the foreground IPs changes their location in the image and so they create different non-repetitive events in the blocks. As can be seen in Table 3.1, these non-repetitive Events created by the foreground IPs, have *RI* values less than the threshold (The threshold has been set to 10 in this table. However, we will discuss the optimum value for threshold in Section 3.3.5). In this regard, the Events of each block are divided into the Dominant Events, created by the background IPs and the non-Dominant ones, created by the foreground IPs. Therefore, the algorithm classifies any Event as dominant if its *RI* is equal to or greater than a threshold. The proposed algorithm is summarized in Algorithm 3.1.

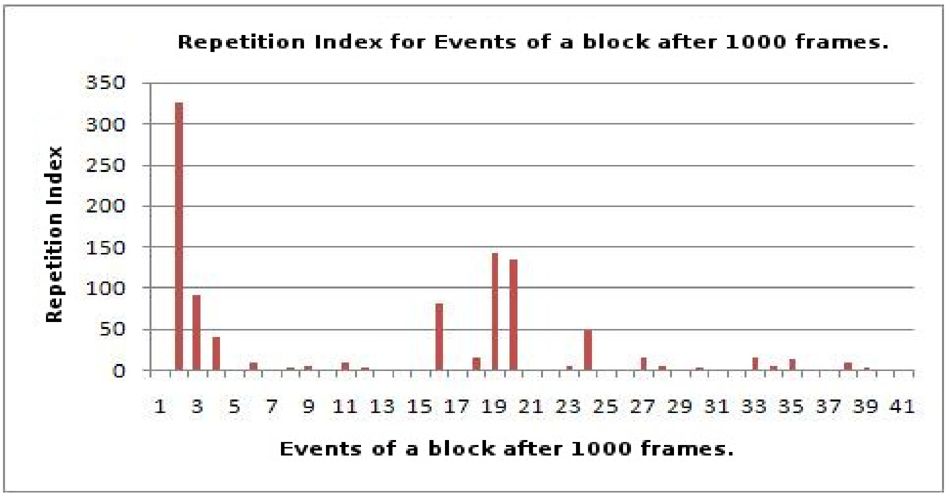


FIGURE 3.4: Repetition Index for Events of a block after 1000 frames.

TABLE 3.1: The stored Events of block 102 of image for 1000 frames.

| # | Event | RI | D | # | Event | RI | D | # | Event | RI | D |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | (363,110) | 1 | | 15 | (280,110,353) | 1 | | 29 | (303,171) | 1 | |
| 2 | (**303,14**) | 325 | √ | 16 | (**285,110**) | 82 | √ | 30 | (303,353,14) | 2 | |
| 3 | (**285,14**) | 92 | √ | 17 | (285,11) | 1 | | 31 | (303,353) | 1 | |
| 4 | (**285**) | 40 | √ | 18 | (**285,12**) | 16 | √ | 32 | (303,394) | 1 | |
| 5 | (125,333) | 1 | | 19 | (**303,110**) | 143 | √ | 33 | (**323,14**) | 13 | √ |
| 6 | (110,353) | 9 | | 20 | (**303**) | 134 | √ | 34 | (323) | 6 | |
| 7 | (110) | 1 | | 21 | (285,9) | 1 | | 35 | (**305,14**) | 14 | √ |
| 8 | (124,353,14) | 2 | | 22 | (285,14,394) | 1 | | 36 | (305,12) | 1 | |
| 9 | (144,353) | 4 | | 23 | (303,11) | 5 | | 37 | (323,110,353) | 1 | |
| 10 | (144,12,353) | 1 | | 24 | (**303,12**) | 49 | √ | 38 | (**323,110**) | 10 | √ |
| 11 | (**144,353,14**) | 10 | √ | 25 | (303,110,353) | 1 | | 39 | (343,14) | 3 | |
| 12 | (280,110) | 2 | | 26 | (303,110,394) | 1 | | 40 | (323,353) | 1 | |
| 13 | (222,110,353) | 1 | | 27 | (**305,110**) | 16 | √ | 41 | (353,14) | 1 | |
| 14 | (280,171) | 1 | | 28 | (305) | 4 | | | | | |

Fig. 3.4 and Table 3.1 show the stored Events for block 102 (7th row and column) in its BST after 1000 frames. As the algorithm assumes, only a few Events appear dominant. On the other hand, the non-dominant Events have been created by the foreground IPs when they have met this block. The dominant Events have been marked with the ”√” sign in the ”*D*” column of the table. IPs which create these dominant Events are classified as background IPs (bold numbers in ”Event” columns of table). They are stored in a list of background IPs.

Fig. 3.5 shows the real image with its extracted IPs (FAST corner IPs in this case), and the foreground and background IPs in red and blue colours, respectively. As can be seen, there are some erroneous background and foreground IPs, which will be named later as FN and FP, respectively. To reduce these errors and amend the results, two post-processing stages are applied to the results as will be described in the next section.

**Algorithm 3.1** IP-Based BGS Algorithm

```
Input: IPs of frames $k = 1, \ldots, K$.
Output: background IPs & foreground IPs of each frame.
Initialization:
for frame $k = 1$ do
    Divide image frame into $B$ blocks.
    Create a Binary Tree (BT) for each block.
    Create the lists of background events (BG-E-list) and background IPs (BG-IP-list).
end for
Background Modelling & Subtraction:
for every frame $k > 1$ do
    Background Modelling:
    for each IP $i$ do
        Assign IP $i$ to the corresponding block based on its $(x, y)$ coordinate.
    end for
    for each block $j$ do
        Create an Event $E_m$ from its assigned IPs.
        Search Binary Tree $BT_j$ for Event $E_m$.
        if $E_m \in BT_j$ then
            Increase Repetition Index of Event $E_m$ by 1 ($RI_m = RI_m + 1$).
            if $RI_m > threshold$ then
                Add Event $E_m$ to BG-E-list.
                Add IPs of Event $E_m$ to BG-IP-list.
            end if
        else
            Add a new node in $BT_j$ and set its $RI$ to 0.
        end if
    end for
    Background Subtraction:
    for each IP $i$ do
        if IP $i \in$ BG-IP-list then
            IP $i \Rightarrow$ background IP.
        else
            IP $i \Rightarrow$ foreground IP.
        end if
    end for
    $k = k + 1$
end for
```

### 3.2.2 Post-Processing

This section describes the post-processing tasks which improve the performance of the proposed IP-based BGS algorithm.

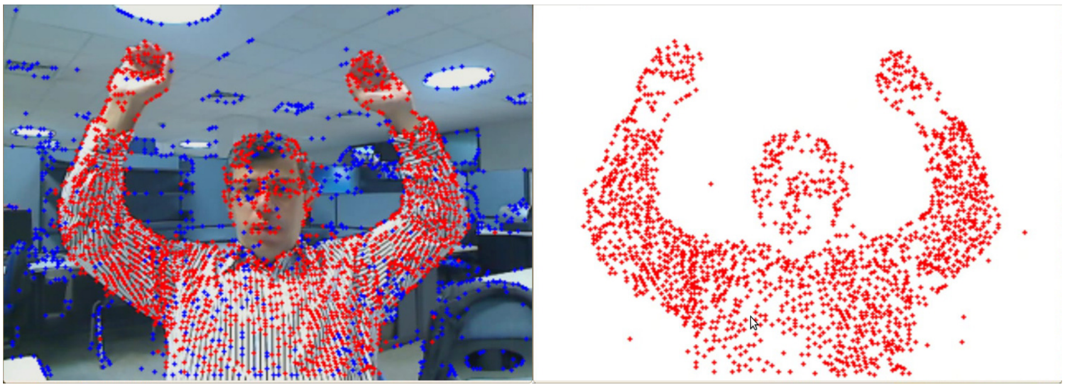
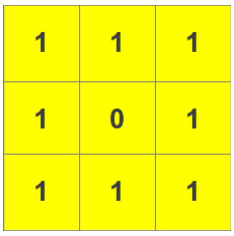


FIGURE 3.5: The real image with its IPs, red: foreground IPs, blue: background IPs.

**Task 1:** As can be seen in Table 3.1, the positions of some IPs have been changed because of changes in illumination, noise, and the quality of the images. For instance, the Events $(285, 14)$ and $(285, 12)$ have happened 92 and 16 times throughout 1000 frames, respectively. This means that IPs in locations 12 and 14 should presumably be the same, their position having been changed across the frames. Since IP 14 has a greater repetition in its corresponding Event than IP 12, it could be said that IP 14 has been mislabelled as 12 in some frames because of noise. Although after 1000 frames the *RI* of Event $(285, 12)$ has satisfied the threshold criterion, which is 10 for this experiment (This value has been selected by trial and error. In future work a more systematic method can be used in this regard.), and so IP 12 has been added to the BG-IP-list, it has been misclassified as a foreground IP for all the frames before satisfying the threshold criterion. To overcome this type of error, which increases the False Positive (FP) rate, a simple comparison between such non-dominant and dominant Events of the block ($(285, 12)$ and $(285, 14)$ for example) is performed in terms of Euclidean distance. This process saves the IPs of these non-dominant Events from being misidentified as foreground IPs. Another example in this experiment is ambiguity between Events $(303, 12)$ and $(303, 14)$.



| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

FIGURE 3.6: The 2D kernel of $3 \times 3$ filter.

**Task 2:** The second post-processing task is to decide about the foreground blocks with no foreground blocks in their neighbourhood. To do this, an image with the same size as the number of blocks ($(H \div N) \times (W \div M)$) is constituted (the Foreground Block Binary (**FBB** image), hereafter). The effect of varying block size will be examined in Section 3.3.2. The value of each pixel in this image is 1 if the corresponding block has at least one foreground IP. Otherwise it is 0. Then a $2D$ $3 \times 3$ filter with the kernel shown in Fig. 3.6 is applied to FBB image to calculate how many foreground blocks each block has around itself. By applying this filter, the foreground blocks with no foreground block in their neighbourhood are identified and considered as spurious foreground blocks and then the status of their IPs is changed from background to background. Also, the background blocks which have foreground blocks in their neighbourhood more than a threshold, are reclassified as foreground blocks. In fact, this task refines the status of some erroneous blocks, which reduces FN and FP. Fig. 3.7 shows the result of the post-processing procedures.

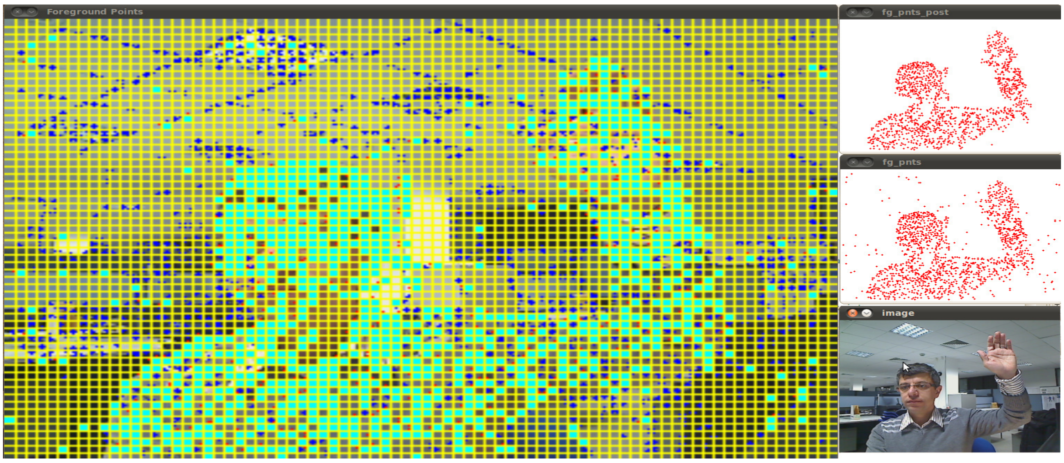


FIGURE 3.7: Left: foreground blocks, Right from bottom to top: the real image, foreground IPs before, and after post-processing.

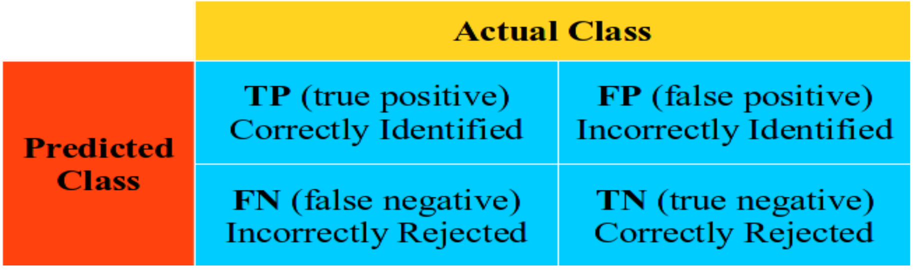
## 3.3 Experimental Results

To evaluate the performance of the proposed IP-based BGS algorithm and also to compare it with the other BGS approaches, we present the results of several experiments in this section. First, the evaluation factors are explained, briefly.

### 3.3.1 Evaluation Factors

Since the background subtraction is a classification task, it can be evaluated based on the overall number of False Negatives (FN) and False Positives (FP) which are produced in each video (Benezeth et al., 2010). Below, Table 3.2 illustrates these factors.

TABLE 3.2: Evaluation factors in classification context



| | **Actual Class** | |
|---|---|---|
| **Predicted Class** | **TP** (true positive)<br>Correctly Identified | **FP** (false positive)<br>Incorrectly Identified |
| | **FN** (false negative)<br>Incorrectly Rejected | **TN** (true negative)<br>Correctly Rejected |

On this basis, the precision and recall are defined as (Olson and Delen, 2008):

$$precision = \frac{TP}{TP+FP} \quad , \quad recall = \frac{TP}{TP+FN} \tag{3.2}$$

Specificity and accuracy, as the other two measures used to measure the performance of binary classification tasks, are used as complementary evaluation factors in some experiments. They are defined as follows (Olson and Delen, 2008):

$$specificity = \frac{TN}{TN+FP} \quad , \quad accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.3}$$

To take into account the precision and recall factors concurrently, the F-measure, which is the harmonic mean of precision and recall values, is considered. As will be seen, we use this factor to find the best value for the threshold of the BGS algorithm. The $F_\beta$ measure for non-negative real values of $\beta$ is:

$$F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \tag{3.4}$$

$\beta$ provides the user the flexibility to vary the weight of the recall $\beta$ times with respect to the precision. The special case of this formula for $\beta$ equal to 1 is known as the $F_1$ measure, which weights the precision and recall evenly.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.5}$$

$F_2$ and $F_{0.5}$ are two other commonly used F measures, which put more emphasis on recall and precision, respectively.

$$F_2 = 5 \cdot \frac{\text{precision} \cdot \text{recall}}{4 \cdot \text{precision} + \text{recall}} \quad , \quad F_0.5 = 1.25 \cdot \frac{\text{precision} \cdot \text{recall}}{0.25 \cdot \text{precision} + \text{recall}} \tag{3.6}$$

### 3.3.2 Experiment 1: Effect of Block Sizes on the Precision and Recall

Through this experiment, the effect of block size on the performance of the algorithm is evaluated according to the above mentioned evaluation factors (Fig. 3.8, 3.9, 3.10). For a given image with the fixed size 256 × 512, several block sizes of powers of 2 (for the sake of better hardware performance) from 1 × 1 to 32 × 32 are examined. The smaller block size tends to process IPs more individually, while the larger size has a higher level of block processing concept. It is supposed that neither the smallest nor the largest block sizes deliver the highest performance. So, through this experiment we try to find a trade-off between the size of the block and robustness to the level of noise and occlusion. As will be seen, the larger the block, the more robust it is to noise but the more prone it is to occlusions.

As Fig. 3.8 shows, the blocks of sizes of 4 × 4 and 8 × 8 deliver higher performances than the other sizes (particularly than the smallest size of 1 × 1 and the larger sizes of 16 × 16 and 32 × 32). On the one hand, it is better than the higher sizes because:

- The smaller block sizes contain lower numbers of IPs. This decreases the risk of losing the background Events in cases where some background IPs of these Events disappear due to: covering by foreground objects; changing the position due to noise; and so

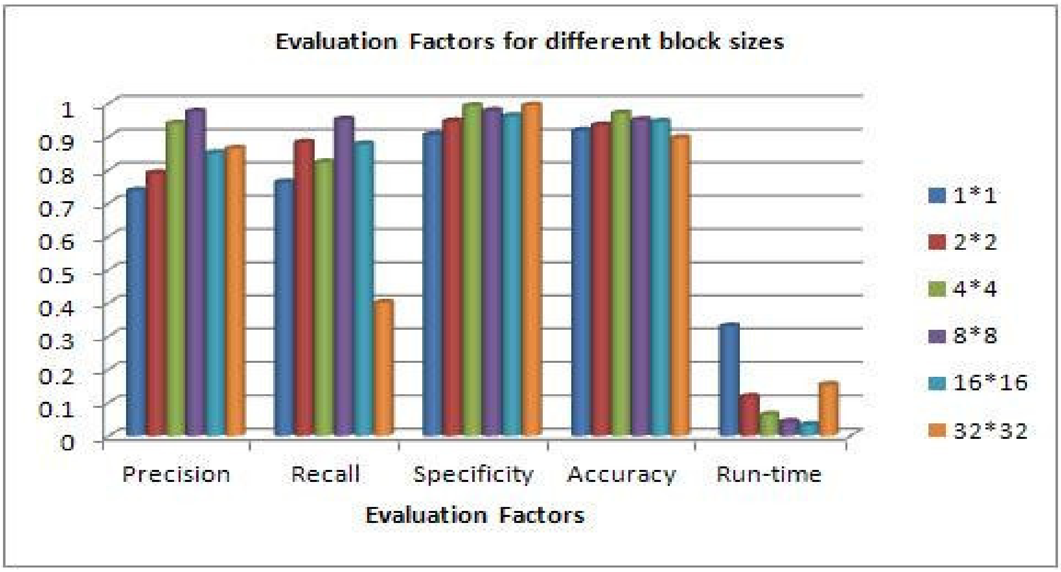


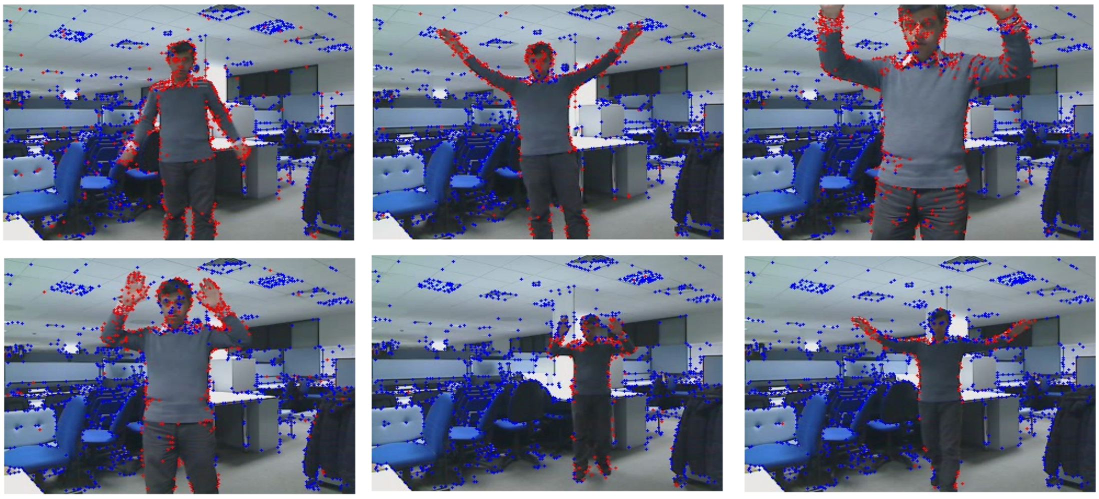
FIGURE 3.8: Evaluation Factors versus block sizes for resolution 256 × 512.

on. This situation is more drastic around the outer contour of foreground object (in Background-Foreground blocks). In other word, the smaller the size of block, the lower the number of IPs in Events.

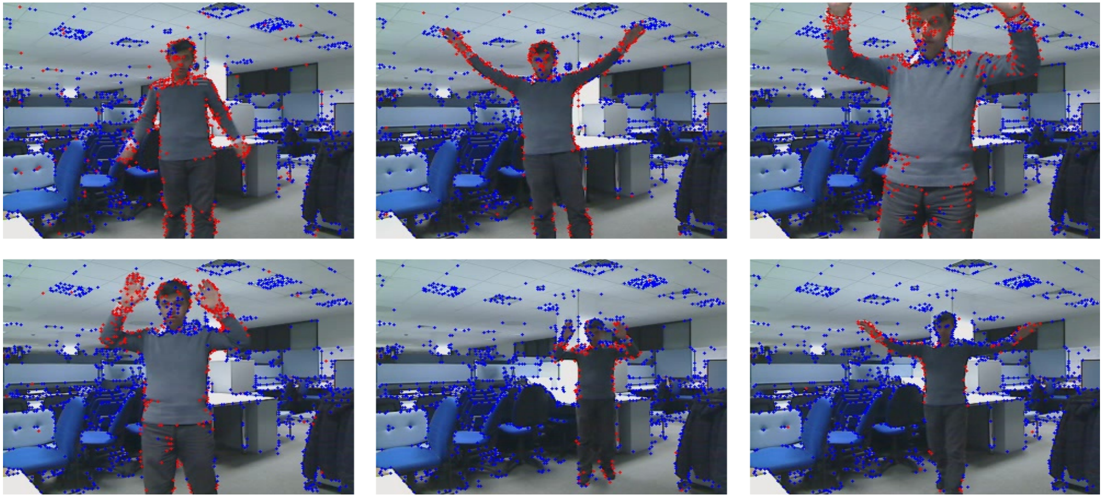
- The foreground objects involve fewer blocks, which decreases the FN & FP errors, even more.
- The smaller the block size, the lower number of Events and the faster the BST. This affects the run time speed of the algorithm as is shown in Fig. 3.10.

On the other hand, the smallest sizes of $1 \times 1$ and $2 \times 2$ cannot deliver the same performance as $4 \times 4$ and $8 \times 8$ because they implicitly ignore the block-wise processing strategy and behave as when the IPs are dealt with individually. This justifies the superiority of block-wise processing over individual processing of IPs.

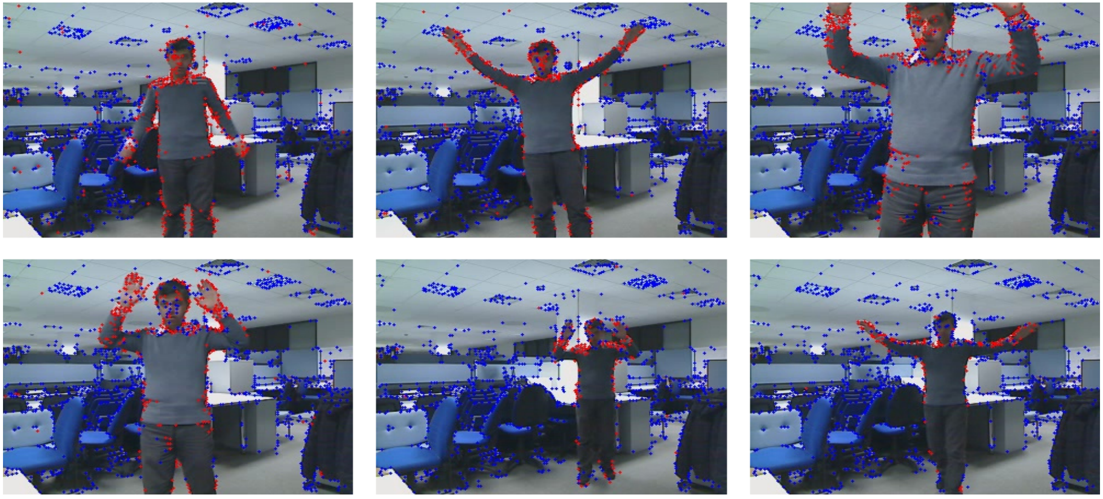
According to the definition of the evaluation factors, the higher value of precision is equal to the higher percentage of retrieved foreground IPs out of all retrieved IPs. On the other hand, recall shows the percentage of correctly retrieved foreground IPs out of the ground truth foreground IPs. Obviously, the block of $8 \times 8$ offers higher precision and recall, particularly better than the $1 \times 1$ and $2 \times 2$ sizes (Fig. 3.8). In addition, specificity determines the percentage



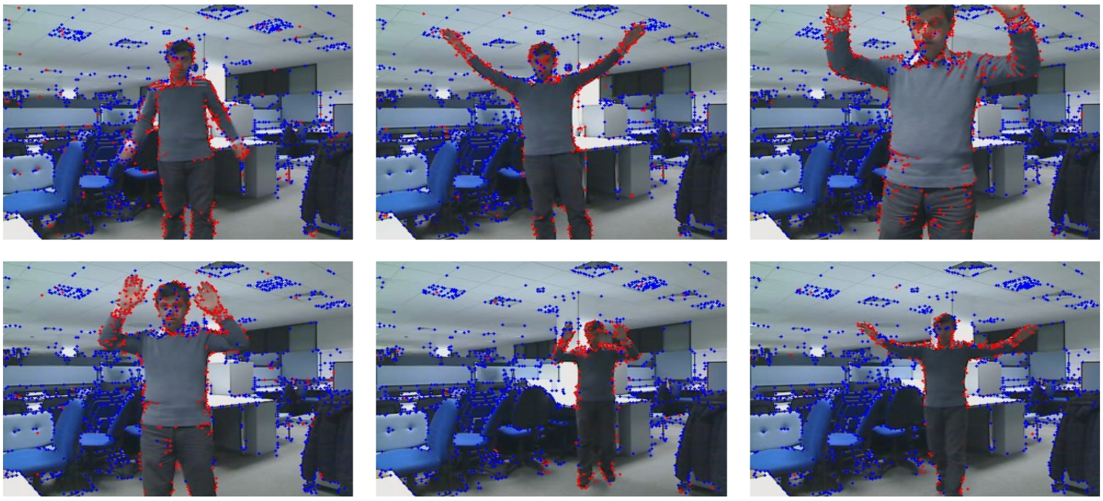
(a) Block size: $1 \times 1$



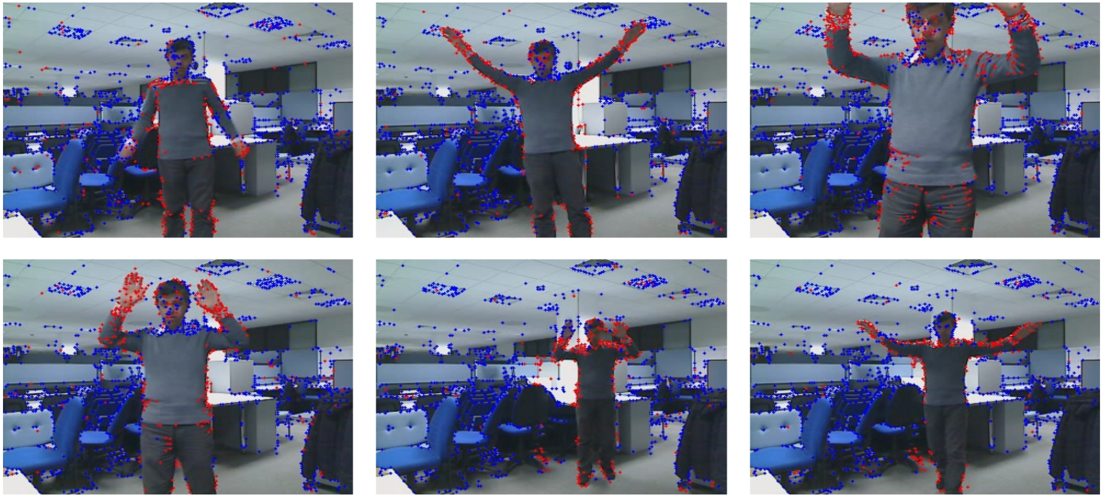
(b) Block size: $2 \times 2$



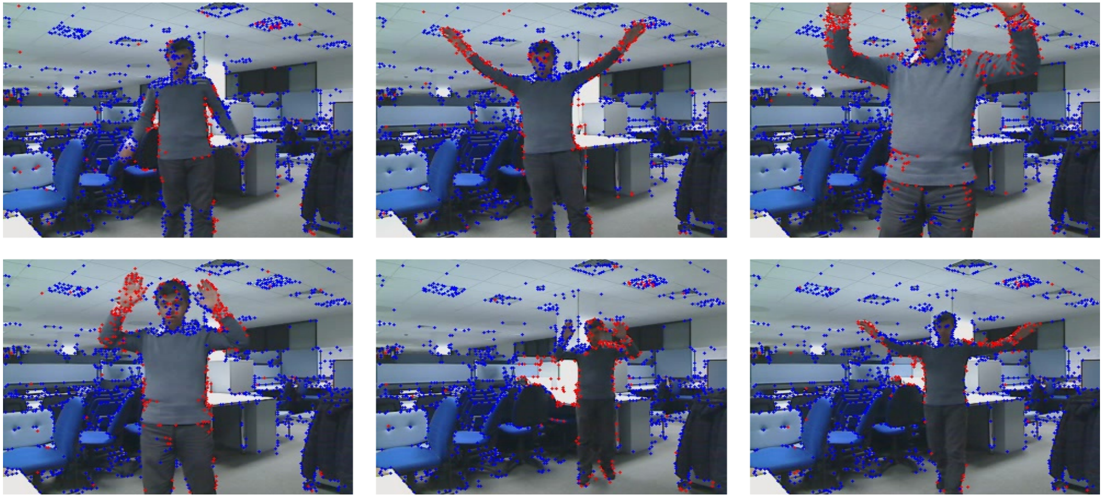
(c) Block size: $4 \times 4$



(d) Block size: 8 × 8



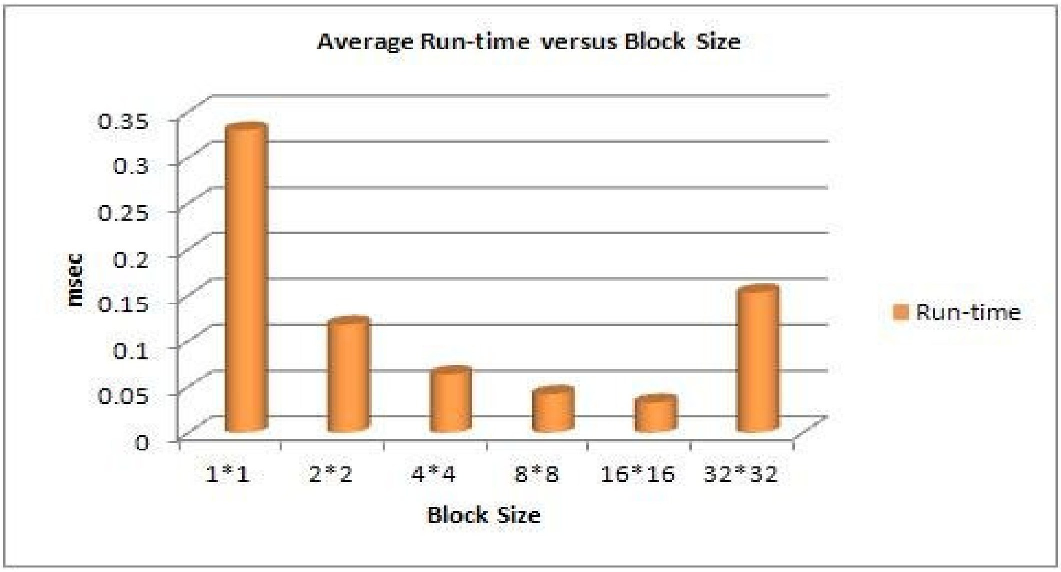
(e) Block size: 16 × 16



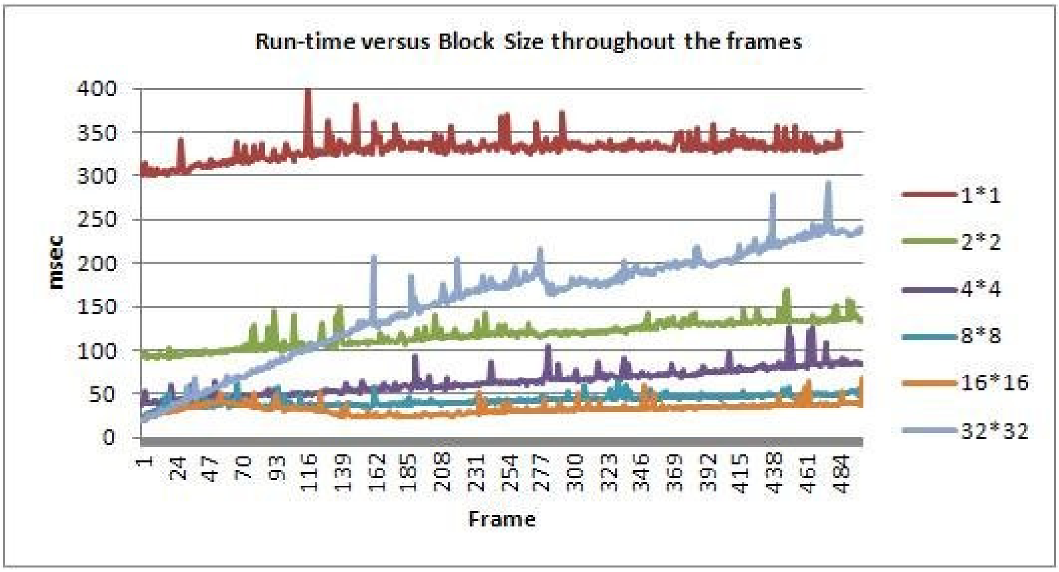
(f) Block size: 32 × 32

FIGURE 3.9: 6 frames vs different block sizes of 8 × 8, 16 × 16, and 32 × 32.

of the ground truth background IPs which has been correctly rejected (the algorithm tries to identify the foreground IPs and reject the background ones). As can be seen, the higher block sizes deliver higher specificity than (or at least equal to) the lower sizes when the algorithm aims to identify the background IPs and reject the foreground ones.



(a) Average run-time versus block size.



(b) Run-time versus block size.

FIGURE 3.10: Run-time versus block size

This experiment has been conducted on images with 256 × 512 pixels. Although the block size is dependent on the image resolution, the result of this experiment can be generalized to any image resolution. To do this, the block size can simply be scaled in proportion to the image resolution.

Fig. 3.9 compares visually the effect of block size on the result of the algorithm on 6 random frames of a video with 450 frames. It is obvious from this figure that the algorithm has a faster and more accurate background Modelling and Update for the block sizes of $4 \times 4$ and $8 \times 8$ than the other sizes. As can be seen in Figs. 3.9(a), 3.9(b), 3.9(e), and 3.9(f), the FP (red IPs in background areas) and FN (blue IPs in foreground areas) rates are higher than 3.9(c) and 3.9(d). It means that the algorithm has incorrectly classified background IPs as foreground IPs in the former case and conversely the foreground IPs as background IPs in the later ones. Again, it confirms that the block sizes of the $4 \times 4$ and $8 \times 8$ outperform than the other sizes.

Fig. 3.10 compares the run-time speed of the algorithm against the block size. As can be seen (Fig. 3.10(b)), the sizes of $4 \times 4$, $8 \times 8$, and $16 \times 16$ are faster. Among these, although the $16 \times 16$ has the fastest execution time, it performs poorly in terms of the evaluation factors (Fig. 3.8). Consequently, the sizes $4 \times 4$ and then $8 \times 8$ look to be the best sizes for the blocks in the proposed algorithm.

It is worth to analyse why the sizes $1 \times 1$ and $32 \times 32$ are slower than the other sizes. The size $1 \times 1$ implies dealing with the IPs individually. Although the Events in this case have only one IP which makes their storage and retrieval faster in the BTS, the higher number of BTS causes a higher computational cost. The size $32 \times 32$ also performs as fast as size $16 \times 16$ for approximately the first 25 frames in Fig. 3.10(b). However, afterwards it turns out to become slower than most of the sizes and gets close to the size $1 \times 1$. This is due to the fact that the bigger size of the block causes a higher number of IPs in each Event. Although it looks to lead to the lower number of Events in the block and be the reason for the fastness throughout the first 25 frames, the less stable IPs (which are related to weaker IPs) of the Events change their position randomly across the frames and therefore they increase the number of Events radically.

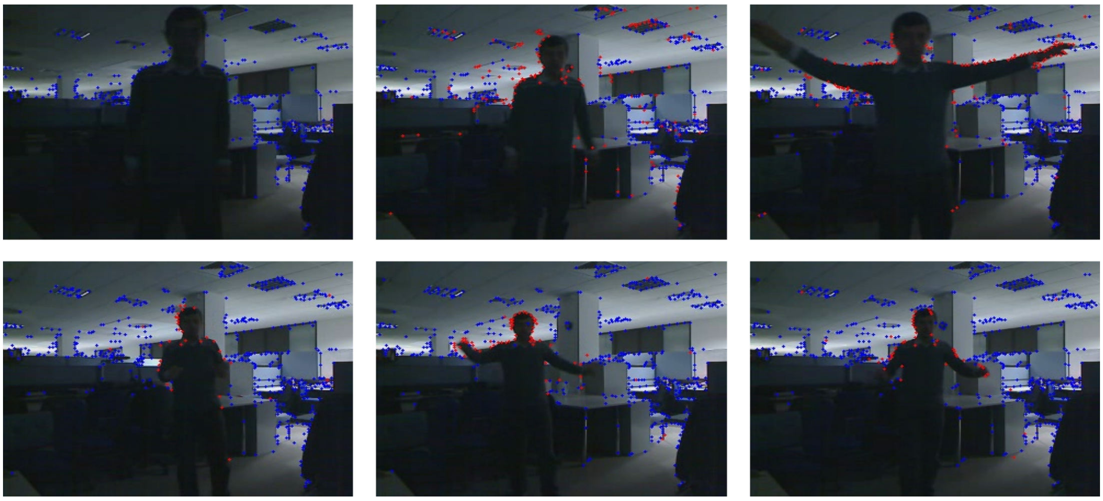
### 3.3.3 Experiment 2: Robustness Under Low-Light conditions

In this experiment, the proposed algorithm is examined under low-light conditions. The results are shown in Fig. 3.11. The pixel-based BGS algorithms produce white and black pixels for the foreground and background areas of image, respectively. In low-light conditions, the foreground white pixels look like separated islands which cannot be interpreted individually to localize the foreground areas (3.11(b)). Whereas, the proposed approach effectively removes the background IPs and presents the foreground IPs, which can be used meaningfully.

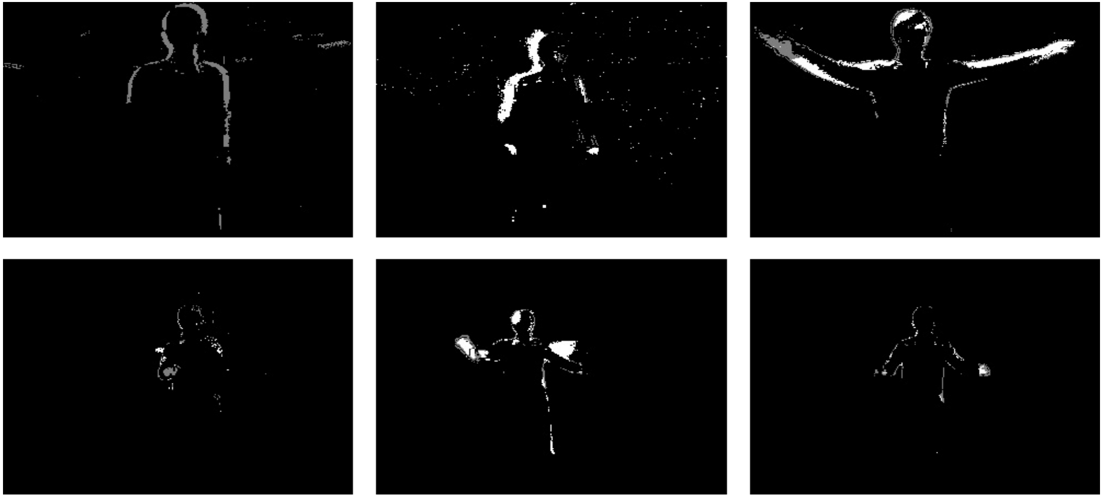
To get some intuition for why this point is true, here is an example. In applications like IP-based motion tracking, the IPs of the object should be tracked across the frames. In this sense, if the IPs of the moving object (foreground IPs) can be discriminated first from the background IPs, as the proposed algorithm does, it makes the tracking process easier. Not only the foreground IPs, but also the background IPs can be used for IP matching and tracking purposes. In contrast, the traditional pixel-based BGS algorithms just delete the background pixel of the image. Moreover, IPs are the significant information of the image, which is more robust to low-light conditions. Consequently, the combination of these properties of IPs delivers more robustness to low-light conditions than pixels. In a sense, it is not efficiency which the proposed algorithm delivers. Instead, it is robustness that IPs give to the proposed algorithm. Fig. 3.11 shows this comparison over 5 random frames of a video with 450 frames.

### 3.3.4 Experiment 3: Comparison of the IP-Based BGS with Different BGS Algorithms

For the purpose of performance evaluation, in this section we compare the efficiency of the proposed algorithm against some of the well-known and state-of-the-art BGS algorithms on the Wallflower dataset (Toyama et al., 1999) based on the total number of errors as the sum of FN and FP, in a similar manner to state-of-the-art BGS papers. This dataset has been widely used in this context and consists of seven different scenarios with a diverse range of problems that might happen in practice. Table 3.3 summarizes these scenarios (Toyama et al., 1999).



(a) IP-based BGS algorithm with block size of 8 × 8.

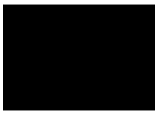
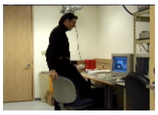
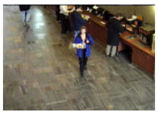
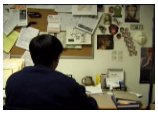


(b) Pixel-based BGS algorithm.

FIGURE 3.11: Comparison between IP-based and pixel-based BGS methods under low-light conditions over 5 random frames of a video with 450 frames.

Although our algorithm is IP-based and looks a little different from the pixel-based BGS algorithms, it similarly separates the foreground IPs from the background ones. Nevertheless, the number of IPs in the image is much less than the number of pixels. Therefore, the total number of erroneous IPs *would not be comparable with the total number of erroneous pixels*. To compensate for this, we define the *Error Ratio* as the ratio of the total number of erroneous IPs (pixels) to the total number of IPs (pixels). Though it may seem unfair to compare an IP-based BGS method with the pixel-based approaches in this way, to the best of our knowledge that is the only way to compare our method against the ground-truth and the state-of-the-art

TABLE 3.3: The Wallflower dataset scenarios.

| **Scenario** | **Description** | **Evaluation Image** | **Ground Truth** |
|---|---|---|---|
| **Moved Object** | A person enters into a room, makes a phone call, and leaves. | |  |
| **Time of Day** | A person enters and sits down in a room where the light is changing gradually from dark to light. | | |
| **Light Switch** | A person enters a lighted room and turns off the lights. Afterwards, person walks in the room, turns on the light, and moves the chair. |  | |
| **Waving Trees** | A person walking in front of a swaying tree. | | |
| **Camouflage** | A person walking in front of a monitor with rolling interference bars with similar colour to the person's clothing. | | |
| **Boostrapping** | A busy café containing several people. |  | |
| **Foreground Aperture** | A person with a uniformly coloured shirt waking up and moving slowly. |  | |

methods. This is due to the novelty of IP-based BGS methods and the lack of a standard ground-truth dataset for them. Tables 3.4 and 3.5 summarize the results for this performance comparison, numerically and visually, respectively.

$$ErrorRatio = \frac{Total\ erroneous\ IPs\ (pixels)}{Total\ IPs\ (pixels)} \tag{3.7}$$

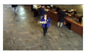
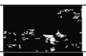
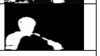
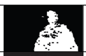
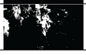
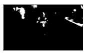
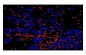
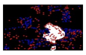
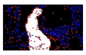
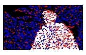
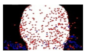
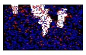
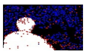
As Table 3.4 shows, our proposed approach delivers a lower Error Ratio rate, and so better performance, in comparison with most of the existing algorithms. Although some of the algorithms in the MoG family work better than ours, these cannot deliver the same advantages

TABLE 3.4: Comparison on the Wallflower dataset for different BGS algorithms.

| **Algorithm** | **Error** | **Moved Object** | **Time Day** | **Light Switch** | **Waving Tree** | **Camouflage** | **Bootstrap** | **FG Aper** | **Total Error** | **Error Ratio** |
|---|---|---|---|---|---|---|---|---|---|---|
| **SG** (Pande et al., 2007) | FN | 0 | 945 | 1857 | 3110 | 4101 | 2215 | 3464 | 35133 | 0.2614 |
| | FP | 0 | 535 | 15123 | 357 | 2040 | 92 | 1290 | | |
| **MOG** (Stauffer and Grimson, 1999) | FN | 0 | 1008 | 1633 | 1323 | 398 | 1874 | 2442 | 27053 | 0.2012 |
| | FP | 0 | 20 | 14169 | 341 | 3098 | 217 | 530 | | |
| **MoG-PSO** (White and Shah, 2007) | FN | 0 | 807 | 1716 | 43 | 2386 | 1551 | 2392 | 13916 | 0.1035 |
| | FP | 0 | 6 | 722 | 1689 | 1463 | 519 | 572 | | |
| **MoG-IHLS** (Setiawan et al., 2006) | FN | 0 | 379 | 1146 | 31 | 188 | 1647 | 2327 | 9739 | 0.0724 |
| | FP | 0 | 99 | 2298 | 270 | 467 | 333 | 554 | | |
| **Improved MOG** (Wang and Suter, 2005) | FN | 0 | 597 | 1481 | 44 | 106 | 1176 | 1274 | 7081 | 0.0526 |
| | FP | 0 | 358 | 669 | 288 | 413 | 134 | 541 | | |
| **MoG-MRF** (Schindler and Wang, 2006) | FN | 0 | 47 | 204 | 15 | 16 | 1060 | 34 | 3808 | 0.0283 |
| | FP | 0 | 402 | 546 | 3011 | 467 | 102 | 604 | | |
| **S-TAPP-MOG** (Cristani and Murino, 2007) | FN | - | - | - | 153 | 643 | 1414 | 1912 | 7844 | 0.1021 |
| | FP | - | - | - | 1152 | 1382 | 811 | 377 | | |
| **ASTNA** (Cristani and Murino, 2008) | FN | - | - | - | 253 | 823 | 2349 | 1900 | 7031 | 0.0915 |
| | FP | - | - | - | 100 | 1173 | 73 | 360 | | |
| **KDE** (Elgammal et al., 2000) | FN | 0 | 1298 | 760 | 170 | 238 | 1755 | 2413 | 26450 | 0.1968 |
| | FP | 0 | 125 | 14153 | 589 | 3392 | 933 | 624 | | |
| **SL-PCA** (Oliver et al., 2000) | FN | 0 | 579 | 963 | 1027 | 350 | 304 | 2441 | 17677 | 0.1315 |
| | FP | 1065 | 16 | 632 | 2057 | 1548 | 6129 | 537 | | |
| **SL-ICA** (Tsai and Lai, 2009) | FN | 0 | 1199 | 1557 | 33720 | 3054 | 2560 | 2721 | 15308 | 0.1138 |
| | FP | 0 | 0 | 210 | 148 | 43 | 16 | 428 | | |
| **SL-INMF** (Bucak et al., 2007) | FN | 0 | 724 | 1593 | 3317 | 6624 | 1401 | 3412 | 19098 | 0.1420 |
| | FP | 0 | 481 | 303 | 652 | 234 | 190 | 165 | | |
| **SL-IRT** (Li et al., 2008b) | FN | 0 | 1282 | 2822 | 4527 | 1491 | 1734 | 2438 | 17053 | 0.1268 |
| | FP | 0 | 159 | 389 | 7 | 114 | 2080 | 12 | | |
| **Our Method** | FN | 0 | 20 | 25 | 208 | 18 | 172 | 17 | 649 | 0.0938 |
| | FP | 0 | 15 | 42 | 57 | 39 | 2 | 34 | | |

in terms of the scope of this chapter. To put it another way, due to the intrinsic properties of the IPs and also the block processing structure of our proposed approach, it delivers an interpretation of the scene in the image plane, which can be used for further processing such as occlusion-detection usable in tracking applications. Some of the properties, which support this interpretation are: classifying the blocks into Foreground, Background, and Background-Foreground blocks; hiding and disclosing the background IPs with foreground IPs; geometrical

TABLE 3.5: Results on the Wallflower dataset for different BGS algorithms.

| **Methods** | **Moved Object** | **Time of Day** | **Light Switch** | **Waving Trees** | **Camouflage** | **Bootstrap** | **FG Aper** |
|---|---|---|---|---|---|---|---|
| **Test Image** | | | | | |  | |
| **Ground Truth** | | | | | | | |
| **SG** (Pande et al., 2007) | |  | | | | | |
| **MOG** (Stauffer and Grimson, 1999) | | | | | | |  |
| **MOG-PSO** (White and Shah, 2007) | | | | | | | |
| **MOG-IHLS** (Setiawan et al., 2006) | | | | | | | |
| **Improved MOG** (Wang and Suter, 2005) | | | | | | | |
| **MOG-MRF** (Schindler and Wang, 2006) | | | | | | | |
| **S-TAP-PMOG** (Cristani and Murino, 2007) | - | - | - | | | | |
| **ASTNA** (Cristani and Murino, 2008) | - | - | - |  | | | |
| **KDE** (Elgammal et al., 2000) | | | | | |  | |
| **Sl-PCA** (Oliver et al., 2000) | | | | | | | |
| **SL-ICA** (Tsai and Lai, 2009) | | | | | | | |
| **SL-INMF** (Bucak et al., 2007) | | | | | | | |
| **SL-IRT** (Li et al., 2008b) | | | | | |  | |
| **Our Method** |  |  |  |  |  |  |  |

relationship between the IPs in the same and the adjacent blocks; identifying the background IPs in low-light conditions.

In contrast, the pixel-based BGS algorithms only represent the foreground pixels in the form of white pixels in a silhouette without providing any information about the relationships among these pixels and the background pixels. Besides, they are not suitable for IP-based computer vision algorithms, unless the output foreground mask of a BGS stage is applied to the IPs to subtract the foreground IPs from the background ones. Although the proposed IP-based BGS algorithm works only on the IPs and delivers the foreground IPs, it would be possible to extract the silhouette roughly using the foreground IPs. As mentioned in Section 3.2.2, the FBB image, which shows the foreground blocks, can be imagined as the silhouette of the foreground object. The resolution and the accuracy of the formed silhouette depends on the number of IPs extracted by the IP detector. The left part of Fig. 3.7 shows a typical silhouette, which we can extract using the algorithm (before the post-processing, which is improved to reduce the noise by post-processing).

Table 3.5 compares the test frames of 7 different scenarios in the Wallflower Dataset for several BGS algorithms alongside the ground truth. The algorithms perform non-uniformly for different scenarios, i.e. they work well for some scenarios while the result for other scenarios is not so good. For example, KDE shows the best and worst results for ”Waving Trees” and ”Time of Day” scenarios, respectively, while SL-IRT has the best and worst results for ”Camouflage” and ”Waving Trees”. In other word, there is no BGS algorithm which handles all of the background subtraction challenges completely and works well in all scenarios. Since the scenarios address different challenges presented by background subtraction, the performance of each algorithm on different scenarios reveals the strength and weakness of the algorithm in addressing the BGS challenges.

Similarly, our approach works relatively well for all of the scenarios, except for ”Waving Trees” and ”Bootstrap”. To justify our claim, we need to understand the importance of FN and FP in the context of this thesis. Normally, these two values can be tuned so one is bigger than the other depending on the application in hand. Alternatively, we try to minimize the sum of these two as our ultimate goal. Owing to the importance of FG-IPs in our HUBT system, the priority for BGS is to reveal the FG-IPs correctly as much as possible. Then, the BG-IPs are of secondary importance. Therefore, the lower values for FN (red IPs in background areas) are more important than the lower values for FP (blue IPs in foreground areas).

Taking a look at Table 3.4, it can be seen that the FN value for ”Waving Trees” and ”Bootstrap” scenarios are relatively higher than the other scenarios. That is the reason for why our algorithm performs weaker for the ”Waving Trees” and ”Bootstrap” scenarios. In contrast, the number of FN errors is less than the number of FP errors for ”Light Switch”, ”Camouflage”, and ”Foreground Aperture” scenarios. This is an important point because it demonstrates that the algorithm can find the foreground IPs over a moving object and it has a low level of foreground misclassification in foreground areas of the image. The FP errors, i.e. the misclassified foreground IPs in background areas, can be compensated for by using some additional post-processing procedures.

The scenarios in the dataset simulate the challenges of background subtraction and the result of our evaluations in Table 3.5 show how our proposed algorithm deals with these challenges. However, we should mention that it is not the purpose of this thesis to introduce an algorithm which competes with the state-of-the-art BGS algorithm, although it does. Instead, we were looking for a way to provide foreground IPs of each frame for the next stages of our human body tracking system at a satisfactory level.

### 3.3.5 Experiment 4: Effect of Threshold on the Results

As described in Section 3.2.1, a threshold value is used to discriminate the background and foreground Events. To evaluate the effect of threshold on the results and find the best value for it, the performance of the algorithm for a range of thresholds from 1 to 100 is evaluated using the F-measure factor of Eq. 3.4, described in Section 3.3.1. Fig. 3.12 shows the $F_1$, $F_{0.5}$, $F_2$, as well as the total number of errors for the experiments on the Wallflower dataset in Section 3.3.4. As can be seen, the threshold values between 20 to 25 is where the lowest total number of error and highest $F_1$, $F_{0.5}$, $F_2$ are gained.

On one hand, the threshold values lower than the optimal value increase the FN error, while keeping FP roughly constant. On the other hand, the higher threshold values, higher than the optimal value, cause a greater FP and keep the FN approximately constant. Both cases yield an increase in the total number of errors. So, to keep the total error at the lowest possible value, a balance between these two source of errors is established by selecting a value between 20 to 25 for threshold.

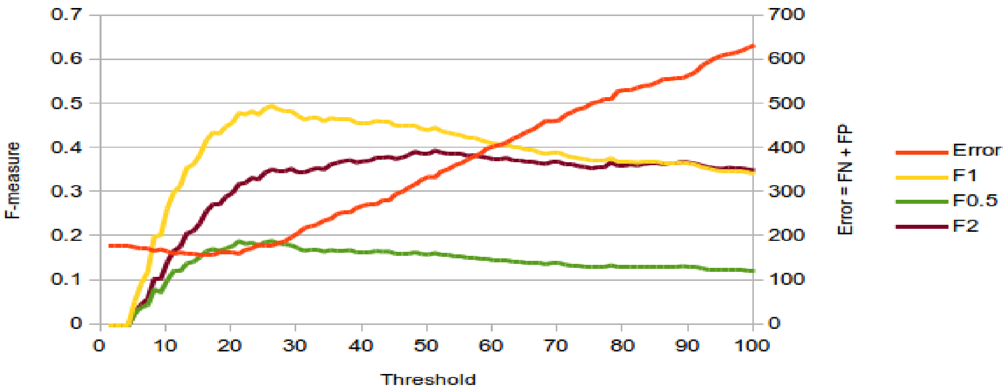


FIGURE 3.12: F-measures (Eq. 3.4) and total number of errors versus threshold.

## 3.4 Concluding Remarks and Future Work

In this chapter, we have proposed a completely new IP-based BGS algorithm which can be used in any IP-based CV application. The key characteristic of our approach is its robustness to illumination changes. According to a block-wise processing strategy, the algorithm divides images into the blocks of the same size. IPs inside blocks are dealt with together as Events across the frame sequence. It stores the Events as well as their number of occurrences (*RI*) throughout the frame sequence. Meanwhile, *RI* is used to classify Events into the background and foreground Events. If any Event has *RI* more than a threshold, it is classified as background Event; otherwise it is classified as foreground Event. The Event classification is used to label the IPs of frames into the foreground and background IPs.

In comparison with the traditional BGS algorithms, the proposed IP-based algorithm has the following key characteristics:

- Is real-time like most of the existing algorithms. As the first stage of any CV system, it is fast enough and adds no latency to the system.
- Is adaptive to changes in background. Since the algorithm needs only a few frames (same number as the threshold value) to find out the location of the background Events and IPs, whenever something is added to or removed from the background, the Events

of the corresponding blocks are affected by the change. First of all, the RI of all existing BG Events will no longer be increased and will remain at the same value as before the change. Second of all, new Events are created for the new appearance of the background in the blocks. In this way, the Events of the block are updated and new BG Events are established, if the change sustains for more than the threshold.

- Works well in low-light indoor environments.
- Not only separates the foreground and background objects, but creates an interpretation of scene, which can be used for any further processing in CV systems.
- Is supposed to work relatively well when the foreground objects do not move and try to stay still. Although in this case it seems the IPs of the foreground should behave like the background and be still, they have enough movement to be recognized as the foreground IPs. In contrast, the traditional approaches only subtract the background when the foreground moves.

We applied our algorithm to BGS under different circumstances. Also, experiments on a public dataset, Wallflower, show that our approach outperforms most of state-of-the-art methods, while it delivers some valuable advantages over those which perform better than our proposed algorithm. The proposed algorithm works in static camera conditions. For future work, we plan to utilize the current algorithm to work in moving camera scenarios, by updating the positions of IPs using Structure from Motion. Moreover, combining the IPs with their parameters gained from IP extraction (such as the strength and local descriptors) should increase the discriminative ability of the IPs which may reduce the errors, though might add to the complexity. The other strategies for decomposition of the image into blocks such as the layout of Shape Context, which is less sensitive to small changes in the position of IPs, can be examined. The effect of frame rate on the performance is worth evaluating, though it is supposed that a higher frame rate would give better results because it is associated with less changes in the images in terms of texture, lighting conditions, etc.

# Chapter 4

# Interest Point Matching

> *No matter what people tell you, words and ideas can change the world.*
>
> ROBIN WILLIAMS, 1951 – 2014

Following the previous chapter, which gave us the foreground IPs of each frame, this chapter goes through the IP-matching algorithm which aims to match the foreground IPs of any two consecutive frames for the purpose of human upper-body tracking. Two approaches, both combined local-spatial IP matching algorithms, are presented in this chapter. They are similar in the local stage, which tries to find confidently matched IPs, while two different strategies, a clustering and graph-matching strategy for the first approach and a Shape Context (SC) based approach for the second one, are used for the spatial stage. While the first approach looks at the spatial relationship between the IPs locally, by clustering them into the groups, the second one behaves globally. To compensate for the problem of Reference List Leakage (RLL), which decreases the number of reference IPs throughout the frame sequence and causes failure of tracking, an IP List Scoring and Refinement (LSR) strategy is presented to maintain the number of reference IPs around a specific level.

At this stage, we do not have any specific quantitative requirements for this chapter. Because this is a novel IP-based matching approach for tracking, our motive is simply to demonstrate its feasibility and measure its accuracy and efficiency,which we will do in Section 4.4.

The rest of this chapter outline is follow: firstly, our motivation for introducing new IP matching algorithms is discussed in Section 4.1. Then, two approaches are described technically in Section 4.2. The LSR algorithm is presented afterwards in Section 4.3. Finally, experimental results and conclusions are discussed in Sections 4.4 and 4.5, respectively.

## 4.1 Motivation

The interest point representation is widely used in image registration, pattern recognition, human motion tracking, etc. IP matching, which aims to find a reliable correspondence between reference and target IPs (extracted from reference and target images) using some similarity criteria, is a crucial and challenging process and has been studied widely. IPs are supposed to be persistent across successive frames and robust to changes in illumination, pose and viewpoint (Maji, 2006). Current IP-matching algorithms mainly use either local or spatial similarity to establish a correspondence between IPs. The local-based methods mainly use feature descriptors to measure the local similarity of points, while the spatial-based methods use geometric distance and spatial structure among IPs (Liu et al., 2012).

Local feature descriptors use image properties such as pixel intensities, colour, texture, and edges to measure the similarity between IPs in the matching process. Many remarkable local feature descriptors such as the SIFT (Lowe, 2004), SURF (Bay et al., 2008), and Gradient Location and Orientation Histogram (GLOH) (Mikolajczyk and Schmid, 2005) have been proposed in the literature. The ORB (Rublee et al., 2011), which is rotation-invariant and resistant to noise, performs as well as SIFT and better than SURF, while being twice as fast. The different feature descriptors have been compared in the literature (Mikolajczyk and Schmid, 2005).

The above mentioned local descriptors are used to match IPs in different applications. However, they may perform poorly in some ambiguous situations such as monotonous backgrounds, similar features, low resolution images, etc. In these cases, spatial-based IP matching methods, which use information like geometric distance or neighbourhood relations between points, can be used to compensate for these drawbacks. The iterative Random Sample Consensus (RANSAC) method (Fischler and Bolles, 1981a), which fits a mathematical model to a set of points including outliers, can be reasonably used only when there are not many outliers. To compensate for this, the spatial relation between points has been dealt with by many authors. Consideration of local relations between IPs (Zheng and Doermann, 2006), graph establishment by Delaunay triangulation in a two-step algorithm (Li et al., 2005), a Graph Transformation Matching (GTM) strategy for finding a consensus nearest neighbour graph from candidate matches (Aguilar et al., 2009), and using relative positions and angles of points for reduction of false matching have been introduced in this regard.

Although the spatial-based methods are more accurate and robust than the local-based ones, they are not as quick particularly when there is a high number of IPs. Owing to both the pros and cons of these local and spatial IP matching strategies, combined approaches (Wen et al., 2008) can be proposed to complement each other. The local feature similarity used in local-based IP matching approaches can be used to cut down the search space for the spatial-based methods. On the other hand, the spatial-based methods can compensate for the defects of local-based methods in ambiguous situations such as duplicated local feature patterns between two reference and target images.

In articulated object tracking, the Reference-list IPs are dynamically matched to the Target-list IPs over the frame sequence (Li et al., 2003; Ma et al., 2013; Zhou et al., 2009). During this process, the IPs in the Reference-list are replaced by the matched points in the Target-list at each frame. Since the matched Target-list will always be shorter than the Reference-list, (because of noise; changes in illumination; articulation of the tracking object, and even the weakness of the background subtraction algorithms) the total number of IPs will be reduced at each frame and eventually this will lead to loss of tracking. We call this problem the RLL problem in this chapter. To tackle this problem, our IP-matching algorithm is equipped with a novel IP LSR strategy.

**Summary of the idea:** In this chapter we propose a dynamic combined local-spatial IP matching algorithm for human-body tracking. In the first stage, confidently matched points are found using a local-based IP matching strategy. Then, to compensate for mismatched and unmatched IPs, two new spatial-based matching methods based on graph matching and shape context are applied. As a remedy for the problem of RLL, a novel LSR strategy is applied. The proposed approach benefits from: local-based IP matching to avoid the expense of the distance and neighbourhood comparison of the spatial-based methods; spatial-based IP matching to compensate for the drawback of the first stage; and an IP List Scoring and Refinement strategy to refine the IP lists and solve the problem of RLL.

## 4.2 Interest Point Matching Algorithms

Since the proposed combined local-spatial IP matching approaches are similar in the first local stage, we describe the local IP matching stage first. Two different situations, the slow and fast movements of the foreground object between the reference and target images, are discussed in this regard. Then, two different spatial matching strategies are discussed, sequentially. Fig. 4.1 shows the IPM algorithm block diagram.

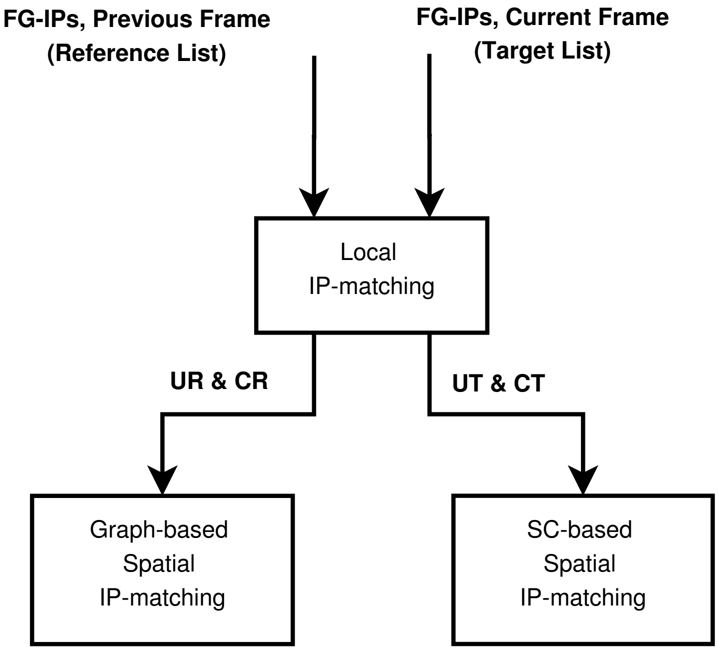


FIGURE 4.1: IP-matching algorithm block diagram.

### 4.2.1 Notations and Definitions

Before going into the details, we introduce some notations and definitions as follows:

1. *Reference-list* and *Target-list*: list of IPs extracted from the reference and target images, $RL = \{r_i\}_{i=1}^{N}$ & $TL = \{t_j\}_{j=1}^{M}$, respectively.
2. $CR$ & $CT$: Confidently matched reference and target sets as the outputs of the local matching stage, $CR = \{cr_i\}_{i=1}^{C}$ & $CT = \{ct_j\}_{j=1}^{C}$, respectively.
3. $UR$ & $UT$: Unmatched reference and target sets as the inputs for the spatial matching stage, $UR = \{ur_i\}_{i=1}^{N-C}$ & $CT = \{ut_j\}_{j=1}^{M-C}$, respectively.
4. *Matches_RT* and *Matches_TR*: result of matching the Reference-list IPs to the Target-list IPs and vice versa. Each list composed of matched-pairs, where each pair shows the IPs of the reference and target list matched to each other.
5. $CR_i$ & $CT_i$: $i^{th}$ cluster of IPs from $CR$ & $CT$ out of $K$ clusters (clustered using the k-means algorithm), $\{CR_i\}_{i=1}^{K}$ & $\{CT_i\}_{i=1}^{K}$.
6. $MR$ & $MT$: Final matched reference and target sets as the outputs of the spatial matching stage.
7. $C(x_c, y_c)$: Centroid of any cluster $CR_i$.
8. $g_R$: a star-shape graph composed for $ur_i$ and the confidently matched IPs of the closest cluster $CR_k$.
9. $g_T$: a star-shape graph composed for $ut_j$ and the confidently matched IPs of the closest cluster $CT_k$.
10. $\mathbf{r} = [r_1, r_2, \ldots, r_n]$: ROC feature vector for a graph $g_R$.
11. $\mathbf{t} = [t_1, t_2, \ldots, t_n]$: ROC feature vector for a graph $g_T$.

### 4.2.2 Stage 1: Local-Based IP Matching

In this stage, first the local feature descriptors of the IPs of Reference-list and Target-list are extracted. Then, the IPs of these lists are matched to each other in two directions, i.e. the IPs of Reference-list to the IPs of Target-list and vice versa. This is carried out because the results of matching in two different directions are not same, no matter what type of matcher and distance measure is used (Fig. 4.2). This process creates two matched-pair lists Matches_RT and Matches_TR. To eliminate the mismatched pairs from these lists, two filtering steps are applied to them as follows:

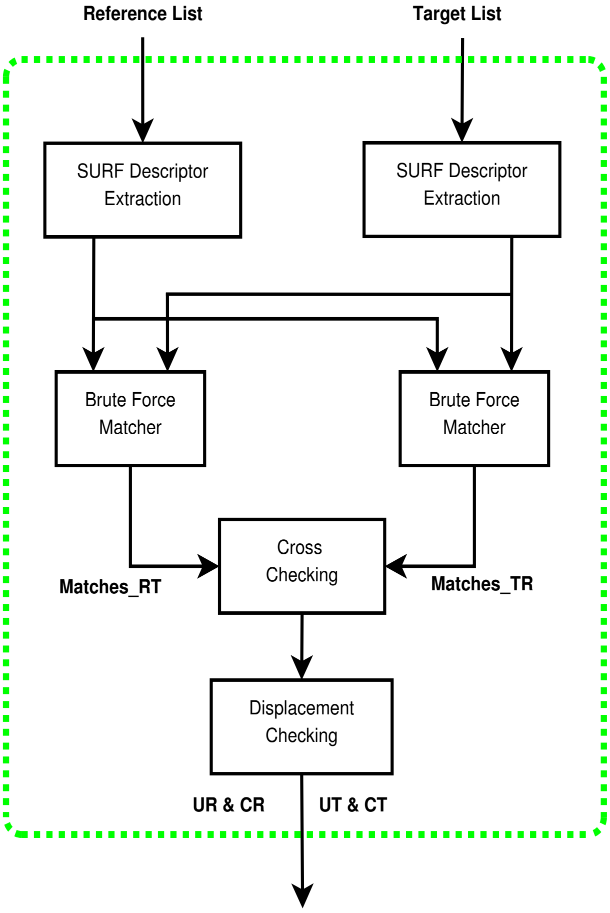


FIGURE 4.2: Local IPM block diagram.

#### 4.2.2.1 Cross-checking

This filtering step (lines 6 to 14 of algorithm 4.1 on Page 81) is used to remove any IPs which do not match both ways. We begin with, the Matches_RT list. Any pair $(i_1, j_1)$ of this list is compared with the pairs of Matches_TR list to find the pair $(j_1, i_2)$ in the list. If $i_1 = i_2$, then it means that the results of matching in two directions are same and so, pair $(i_1, j_1)$ is kept in Matches_RT list. Otherwise, it is thrown away from the list. Fig. 4.3 shows how the knnMatcher of OpenCV creates different results in two different directions and how cross-checking can amend the results by removing the mismatched pairs from the output. Although the cross-checking step filters out many of the mismatched pairs, still the Matches_RT list has some remaining mismatched pairs. These will be eliminated through the next filtering, displacement-checking, step.

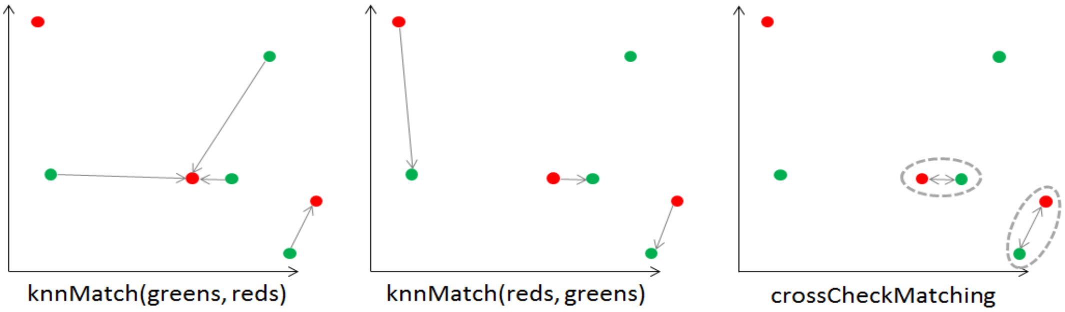


FIGURE 4.3: Why cross-checking? Left to right: green to red matching, red to green matching, and Cross-checked matching.

#### 4.2.2.2 Displacement-checking

This filtering step (lines 15 to 22 of algorithm 4.1 on Page 81) is used to remove any matched IP pairs, whose displacement distance is greater than a maximum allowable distance. Although, in terms of matching, this filtering stage looks similar to optical flow, it is used here to just find the best match to any IP, whereas optical flow uses it to match the IPs across the frames and track them.

A naive approach for displacement-checking would be to delete any matched-pair of Matches_RT list whose displacement length (the Euclidean distance between its reference and target IPs) is greater than a threshold. Although this might be acceptable under smoothness or small inter-frame motion assumptions, which are valid to assume in human body tracking (Herda et al., 2000), this would be too severe in cases where faster movements between consecutive frames happen. As a remedy, we need to use a dynamic Displacement Threshold (DT) for displacement-checking, which is calculated for any two consecutive frames. Seeing that the well matched-pairs introduce a roughly similar displacement length than the mismatched pairs, calculating the dynamic DT is equivalent to finding the most probable displacement length between the IPs of matched-pairs for any two successive frames.

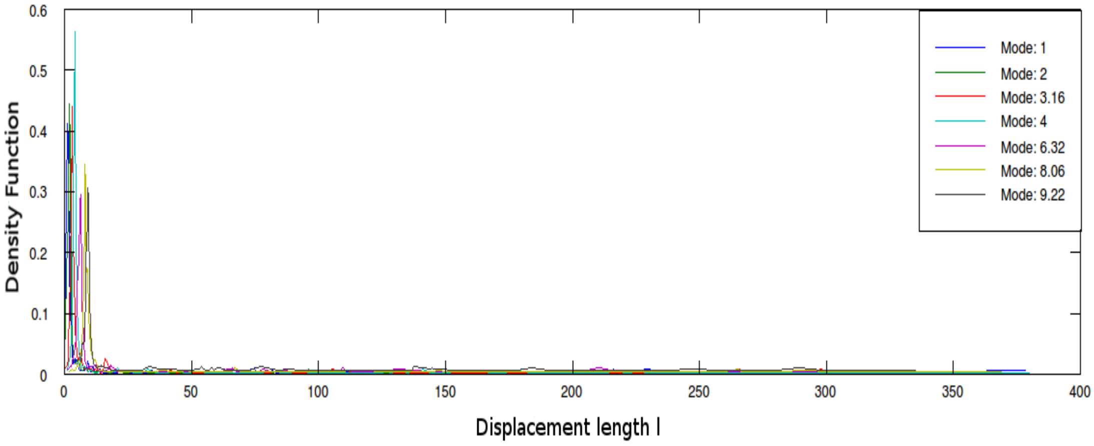
The well-known Kernel Density Estimation (KDE) method, a non-parametric method for estimating the pdf of a random variable (Parzen, 1962), is used to estimate the displacement length between the matched IPs of any two consecutive frames. In this regard, the length of all lines passing through the matched-pairs of the Matches_RT list, starting from the Ref-IP and ending at Tar-IP, are calculated. For example, for each matched pair $(P_{i_k}, P_{j_k})$ in Matches_RT, the displacement length would be:

$$l_k = \|P_{i_k} - P_{j_k}\| \tag{4.1}$$

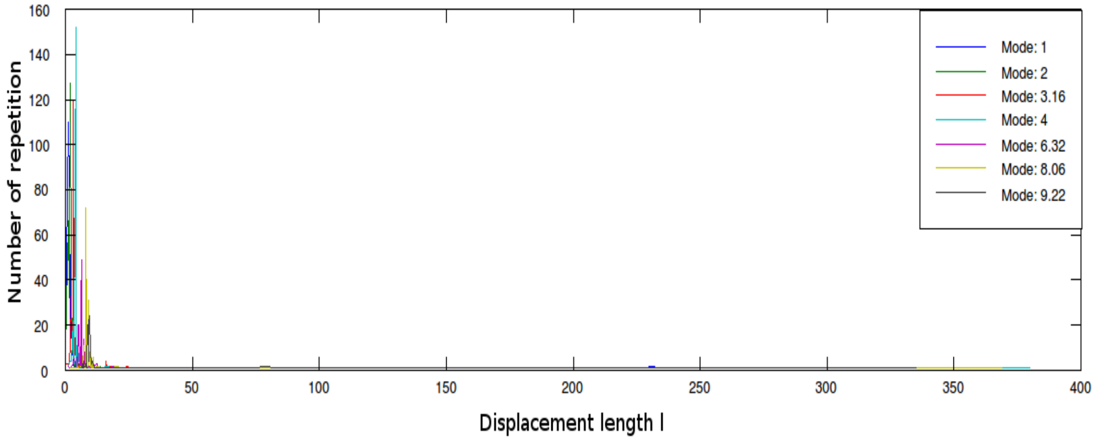
This process creates a continuous Random Variable (RV) $l = (l_1, l_2, \ldots, l_n)$, whose pdf we aim to calculate. The statistical *Mode* of this multi-modal RV (Forbes et al., 2011) gives the most probable displacement length among all the pairs of the Matches_RT list. Also, this pdf can be used to calculate the probability of any displacement length. Fig. 4.4 shows the estimated pdf as well as the number of repetitions for different displacement length across 7 different frames. Each curve shows the displacement length $l = (l_1, l_2, \ldots, l_n)$ for any two subsequent frames. The *Mode* for any curve also has been shown on the figures.

The kernel density estimator for estimating the pdf of an independent and identically distributed RV $l = (l_1, l_2, \ldots, l_n)$ is (Parzen, 1962):

$$\hat{f}_h(l) = \frac{1}{n}\sum_{i=1}^{n} K_h(l - l_i) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{l - l_i}{h}) \tag{4.2}$$



(a) Density function versus displacement length l.



(b) Number of repetition versus displacement length l.

FIGURE 4.4: Estimated PDF as well as the number of repetitions for different displacement length throughout several frames.

where $h$ is a smoothing parameter called the bandwidth, a free parameter which has a strong influence on the resulting estimate, and $K$ is called the kernel function. A range of kernel functions are commonly used, among which the normal kernel, due to its convenient mathematical properties, is often used and defined as:

$$K\left(\frac{l-l_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(l-l_i)^2}{2h^2}} \tag{4.3}$$

Fig. 4.5 shows the Displacement Threshold estimation over 1200 frames. The estimated dynamic DT is applied to the matched-pairs of the Matches_RT list to determine and delete

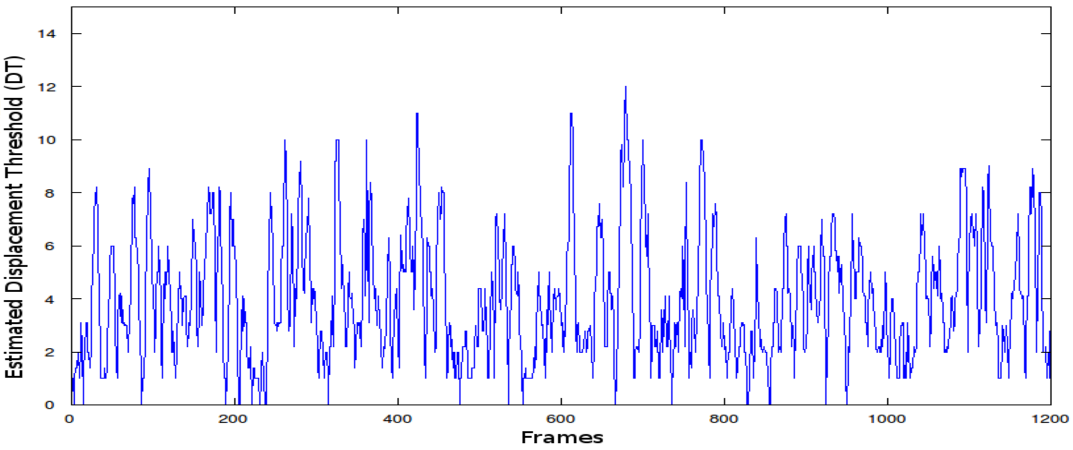


FIGURE 4.5: Displacement Threshold estimation over 1200 frames.

the mismatched pairs (lines 15 to 27 of algorithm 4.1 on Page 81). Pairs with displacement $l_k$ outside the interval $[DT - \delta, DT + \delta]$ are recognized as mismatched pairs and are deleted from the Matches_RT list. As will be discussed later in Section 4.4, δ determines the level of accuracy that we need to create the “Confidently”matched sets. The smaller δ, the more confident the filtered matched IPs.

The two mentioned filtering steps amend the Matches_RT list and deliver the Confidently matched-pairs. From that, the Confidently matched IPs of reference and target lists, $CR$ & $CT$, are created, respectively. The deleted (unmatched) IPs of Matches_RT list compose the $UR$ and $UT$ IP-sets, which will be processed in the next spatial-based stage of the algorithm to recover and increase the number of matched-pairs. Algorithm 4.1 outlines the local-based IP matching part of the algorithm.

### 4.2.3 Stage 2: Spatial-Based IP Matching

Obviously, the local matching stage increased the precision rate of the algorithm, whereas the recall rate is still at a low level. To increase the recall rate whilst keeping the precision at its high level, other matched pairs have to be recovered. In this regard, the confidently matched pairs from the local stage are used as a foundation for the spatial stage of algorithm. Therefore, the accuracy of the local stage is more important than the number of the matched points. In this

**Algorithm 4.1** Local-Based IP Matching Algorithm

```
Input: Two IP set Reference-list: $\{r_i\}_{i=1}^{n}$ &
Target-list: $\{t_j\}_{j=1}^{l}$
Output: Confidently matched set $CR$ & $CT$
Unmatched set $UR$ & $UT$
Extract feature descriptor for both IP lists.
Match Reference-list to Target-list: Matches_RT.
Match Target-list to Reference-list: Matches_TR.
Cross-Check:
for each matched pair $(i_1, j_1)$ in Matches_RT do
    Find matched pair $(j_1, i_2)$ in Matches_TR.
    if $i_1 = i_2$ then
        Keep pair $(i_1, j_1)$ in Matches_RT.
    else
        Push-back $P_{i_1}$&$P_{j_1}$ to $UR$ & $UT$ respectively.
    end if
end for
Displacement-Check:
for each matched pair $(i, j)$ in Matches_RT do
    if $\|P_i - P_j\| < threshold$ then
        Push-back $P_i$&$P_j$ to $CR$ & $CT$ respectively.
    else
        Push-back $P_i$&$P_j$ to $UR$ & $UT$ respectively.
    end if
end for
```

section, we present two different strategies for the spatial IP matching, the graph-matching-based and the shape-context-based, as follows, which are just two different ideas we examine in this thesis. Since descriptors have been used in the local stage, the position of IPs and their spatial relationship is considered in this stage to compensate for the drawbacks of the local stage.

Our motivation for trying two different approaches is that the first approach looks at the spatial relationship between IPs locally by clustering them into groups and the second approach behaves globally. We wished to see if there was a significant difference between them but as shown in the experiments of Section 4.4.3, the proposed methods perform at the same level approximately.

#### 4.2.3.1 Graph-Based Spatial IP Matching

After finding the confidently matched sets $CR$ and $CT$, the unmatched IPs of the Reference-list, i.e. $UR$, are dealt with one by one to find their possible corresponding matched IPs in the unmatched IPs of the Target-list, $UT$. It would also be possible to match IPs in the reverse direction i.e from the target list to the reference list. We will deal with that in the future works (see p. 106). Before that, the IPs of $CR$ are clustered into $K$ groups $\{CR_1, \dots, CR_K\}$ using the k-means clustering algorithm. The optimum values for $K$ is selected by trying a range of values from 0 to 20 and picking 6 as the best one in terms of evaluation factors we use in Section 4.4. Fig. 4.7(a) shows clusters with different colours. The centroid of each cluster $C(x_c, y_c)$ is calculated by:

$$x_c = \frac{1}{N}\sum_{i=1}^{N} x_i \quad , \quad y_c = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{4.4}$$

In the meantime, the corresponding $K$ clusters of $CT$, i.e. $\{CT_1, \dots, CT_K\}$, are obtained from correspondence between the confidently matched IPs of $CR$ and $CT$. Then for each unmatched point $ur_i$ of $UR$, the closest cluster $CR_k$ is found by comparing its Euclidean distance to the centre of each cluster. Now, $ur_i$ and the confidently matched IPs of the closest cluster $CR_k$ compose a star-shape graph $g_R$ similar to Fig. 4.6.

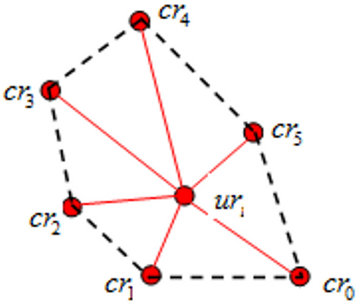
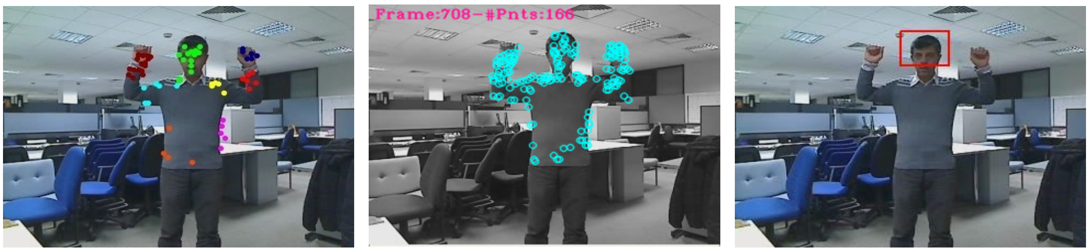
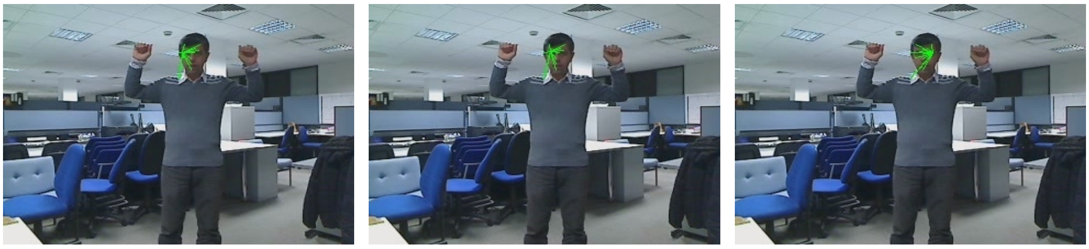


FIGURE 4.6: The Graph $g_r$, confidently matched points $cr_1, cr_2, \dots, cr_n$, and unmatched point $ur_i$.

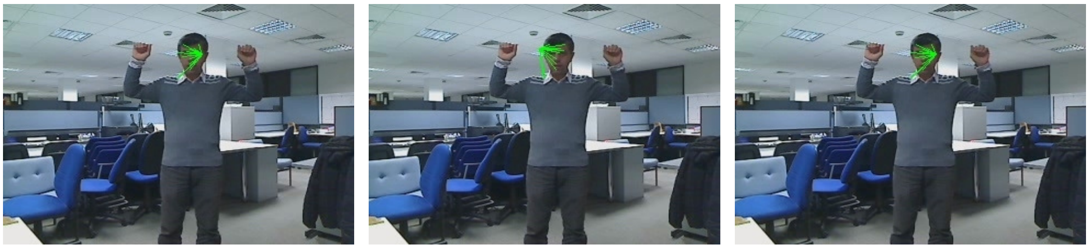
To find a possible matching IP to this point, a rectangular search area is defined in the target image whose centre is the position of $ur_i$ plus the displacement vector, estimated during the displacement-checking step. The more precise the displacement vector, the smaller the size of the search area. All the unmatched points $ut_j$ in this search area are examined one by one to see if there is any point which can be matched to the unmatched point $ur_i$. To do this, a similar graph $g_T$ is constituted for any unmatched point $ut_j$ (inside the search area) and its confidently matched IPs of the corresponding cluster $CT_k$. The point $ur_i$ is matched to one of points $ut_j$ if graph $g_R$ is matched to one of the graphs $g_T$. Otherwise, this IP remains unmatched. Fig. 4.7 shows the graph formation process of the graph-based spatial matching stage. Based on the size of the search area (Fig. 4.7(c)), a few possible graphs $g_T$ (Fig. 4.7(e)-4.7(i)) are composed, for the candidate unmatched points $ut_j$ inside the search area, to be matched to graph $g_R$ of reference unmatched point $ur_i$ (Fig. 4.7(d)).



(a) Clustering. (b) Target IPs. (c) Search area.



(d) $g_R$. (e) 1st $g_T$. (f) 2nd $g_T$.



(g) 3rd $g_T$. (h) 4th $g_T$. (i) 5th $g_T$.

FIGURE 4.7: Graph formation: (a) the clustered confidently matched IPs, (b) the unmatched target IPs, (c) search areas, (d) a reference graph, and (e-i) five target graphs for unmatched target IPs in the search area.

For the task of graph matching, the cyclic string matching algorithm (Maes, 1990), which has been an effective matching method in pattern recognition problems, is used. This process is detailed as follow:

- **Primitive feature extraction**: For the purpose of graph matching, firstly a primitive feature vector is extracted for any of the candidate graphs $g_R$ and $g_T$. This feature should be as light-weight and small-size as possible, while robust to translation, rotation, and scale. The reciprocal of compactness (ROC) (Wu and Wang, 1999) is a good choice, in this sense, and satisfies these requisites. To extract this feature vector for any graph $g_R$, with central point $ur_i$ and confidently matched IPs $\{cr_1, cr_2, \ldots, cr_n\}$, the triangle $ur_i cr_i cr_{i+1}$ is composed for each point $cr_i$, then the feature value $r_i$ is calculated for that triangle as:

  $$r_i = \frac{a_i}{p_i^2} \tag{4.5}$$

  where $p_i = |\overline{cr_i cr_{i+1}}| + |\overline{cr_i ur_i}| + |\overline{cr_{i+1} ur_i}|$ is the perimeter and $a_i$ is the area of the triangle.

  Therefore, the feature vectors $\mathbf{r} = [r_1, r_2, \ldots, r_n]$ and $\mathbf{t} = [t_1, t_2, \ldots, t_n]$ of real numbers are created for the graphs $g_R$ and $g_T$, respectively (The reference and target graphs have the same number of matched IPs). These feature vectors compose strings **s** and **t**, which are applied to the string matching algorithm to measure the similarity of two graphs. Before going through the matching process, first the cyclic string matching technique is described briefly.

- **Cyclic string matching technique**: To match two strings $\mathbf{s} : [s_1, s_2, \ldots, s_n]$ and $\mathbf{t} : [t_1, t_2, \ldots, t_m]$ with length $n$ and $m$ respectively, the linear string-to-string correction problem is used. The string with zero length is called the null string and showed by $\lambda$. Given an *edit cost function* $\varepsilon$, three types of edit operations are defined as follows:

  - *Insertion*: $v(i, j) \rightarrow v(i, j+1)$ with weight $w_{0,j+1} = \varepsilon(\lambda, t_{j+1})$,
    $for \quad i = 0, 1, \ldots, n \;\&\; j : 0, 1, \ldots, m-1.$

- *Deletion*: $v(i,j) \rightarrow v(i+1,j)$ with weight $w_{i+1,0} = \varepsilon(s_{i+1},\lambda)$,
  $for \quad i = 0,1,\ldots,n-1 \ \& \ j : 0,1,\ldots,m.$
- *Change*: $v(i,j) \rightarrow v(i+1,j+1)$ with weight $w_{i+1,j+1} = \varepsilon(s_{i+1},t_{j+1})$,
  $for \quad i = 0,1,\ldots,n-1 \ \& \ j : 0,1,\ldots,m-1.$

Using the edit cost function $\varepsilon$ and the three above operations, an edit weighted graph $G$ with vertices $v(i,j)$; $i : 0,1,\ldots,n$ & $j : 0,1,\ldots,m$ is constructed (Wu and Wang, 1999). The graph matching using string-to-string matching is equivalent to finding the shortest path in the graph from $\mathbf{s}$ to $\mathbf{t}$ which minimizes the edit distance $\delta([\mathbf{s}],[\mathbf{t}])$ (Wagner and Fischer, 1974). The edit distance, here, is the summation of the cost of all operations in the edit sequence. Fig. 4.8 shows an example edit graph $G$ for string $\mathbf{s}$ and $\mathbf{t}$ with $n = 5$ and $m = 4$, respectively, the edit operations, and the shortest path from $\mathbf{s}$ to $\mathbf{t}$. In this example, the shortest path in the graph happens through the edit sequence: $e_1 : (s_1 \rightarrow t_1)$, $e_2 : (s_2 \rightarrow \lambda)$, $e_3 : (s_3 \rightarrow t_2)$, $e_4 : (s_4 \rightarrow \lambda)$, $e_5 : (\lambda \rightarrow t_3)$, $e_6 : (s_5 \rightarrow t_4)$. Weights in Figs. 4.8 and 4.9 are calculated using the edit cost function of Eq. 4.7.

Since strings $s$ and $t$ are extracted from cyclic graphs, no matter which confidently IP $cr_i$ is considered as the first point during the feature extraction process, the *cyclic* string-to-string correction problem would be more applicable to our graph matching scenario. In this situation, string $\mathbf{t}' : [t_1,t_2,\ldots,t_m,t_1,t_2,\ldots,t_m]$ is firstly composed from string $\mathbf{t}$ by concatenating $\mathbf{t}$ to itself. Then, the edit graph $H$ (Wu and Wang, 1999) is constructed for strings $\mathbf{s}$ and $\mathbf{t}'$ similar to graph $G$. The shortest path, and the edit sequence for the example of Fig. 4.9 is: $e_1 : (s_1 \rightarrow t_3)$, $e_2 : (s_2 \rightarrow t_4)$, $e_3 : (s_3 \rightarrow \lambda)$, $e_4 : (s_4 \rightarrow t_1)$, $e_5 : (s_5 \rightarrow t_2)$.

To determine the minimum cost edit sequence from $\mathbf{s}$ to $\mathbf{t}'$, the edit distance $\delta(\mathbf{s},\sigma^j(\mathbf{t}'))$ is examined to find the shortest path from $v(0,j)$ to $v(n,m+j)$ for $j : 0,1,\ldots,m-1$ (Maes, 1990). The minimum edit distance and its corresponding edit sequence determines the edit sequence from $\mathbf{s}$ to $\mathbf{t}'$ as follow:

$$\delta(\mathbf{s},\sigma^j(\mathbf{t}')) \quad = \quad min\{\delta(\mathbf{s},\sigma^j(\mathbf{t}'))\}, \quad for \quad j : 0,1,\ldots,m-1 \tag{4.6}$$

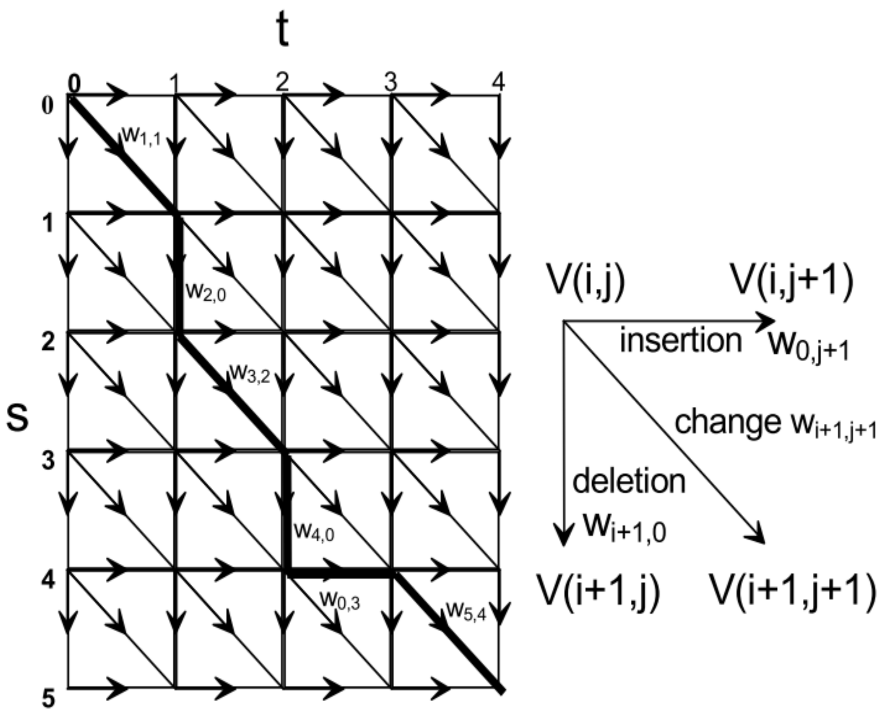


FIGURE 4.8: An example edit graph G with $n = 5$ and $m = 4$, the edit operations, the shortest path from **s** to **t**, and its corresponding edit sequence, adapted from (Wu and Wang, 1999). Weights are calculated using the edit cost function of Eq. 4.7.

where $\sigma^j(\mathbf{t}')$ is the string obtained from $\mathbf{t}'$ after $j$ cyclic shifts.

- **Graph matching using cyclic string matching technique**: To do this, an edit-weighted graph $H$ is constructed for the feature vectors extracted **r** and **t** from the graphs $g_R$ and $g_T$, respectively. The string matching algorithm finds a minimum cost edit sequence from "**r**" to "**t**". Algorithm 4.2 summarizes this stage. To find the shortest path in the edit-weighted graph during the cyclic string matching, the Dijkstra algorithm (Leiserson et al., 2001) which is a graph search algorithm for finding the shortest path in a graph, is used. Here, the edit cost function is considered as:

$$\varepsilon(r_i \rightarrow t_j) = |r_i - t_j| \tag{4.7}$$

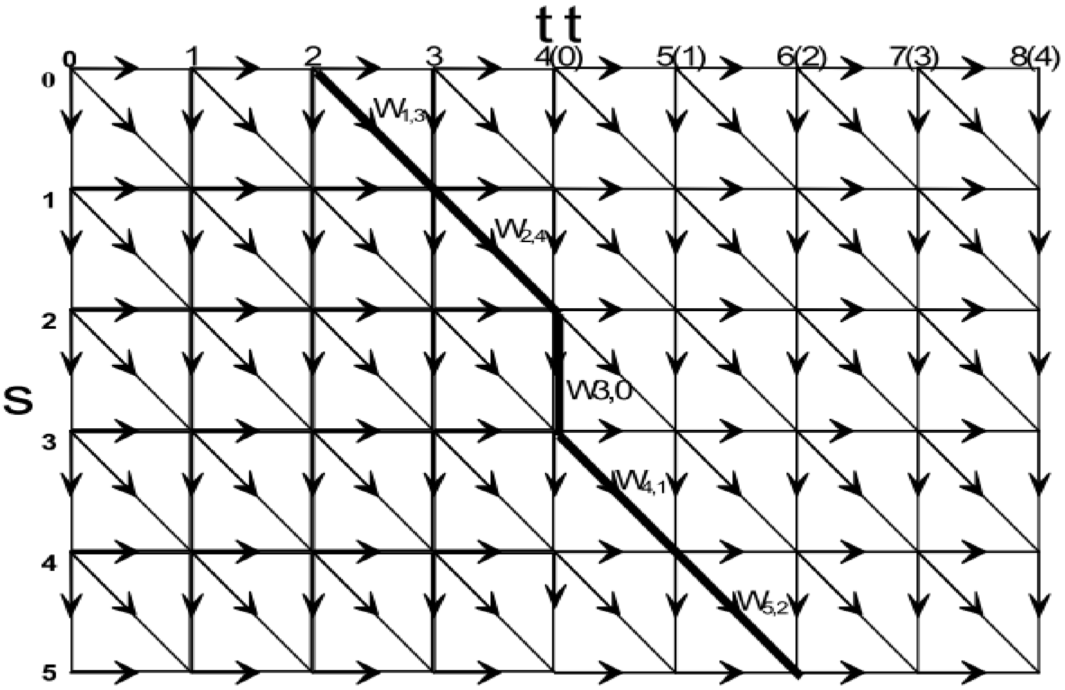


FIGURE 4.9: The edit graph $H$ for the graph $G$ of Fig. 4.8 adapted from (Wu and Wang, 1999). Weights are calculated using the edit cost function of Eq. 4.7

#### 4.2.3.2 Shape-Context-Based Spatial IP Matching

After finding the Confidently matched sets $CR$ and $CT$, the unmatched IPs of $UR$ are dealt with one by one to find their possible corresponding matched IPs in the $UT$ set. To do this, a spatial feature descriptor is calculated for any unmatched IP using the Confidently matched IPs of its corresponding set. This feature descriptor should reflect the spatial relationship of the point to the IPs of the corresponding Confidently matched Set.

In this section we use the shape context (Mori et al., 2005) for spatial feature extraction. SC is a spatial feature descriptor invariant to translation, scale, small perturbations, and rotation depending on the application. It has been empirically (Chui and Rangarajan, 2000) shown that SC is robust to deformations, noise, and outliers (Belongie et al., 2000). All these features make SC a good choice for IP matching (Zheng and Doermann, 2006). As the basic idea of SC shows in Fig. 4.10, the spatial relationships of a point to its neighbouring points is

**Algorithm 4.2** Graph-Based Spatial IP Matching Algorithm

```
Input: IP sets $CR$ & $CT$ and $UR$ & $UT$.
Output: Matched IP sets $MR$ & $MT$.
Push back $CR$ & $CT$ into $MR$ & $MT$.
Cluster $CR$ into $K$ clusters $\{CR_1, \ldots, CR_K\}$.
Compose $K$ clusters $\{CT_1, \ldots, CT_K\}$.
for each IP $ur_i$ of $UR$ do
  Find its closest cluster, compose graph $g_R$, and extract its ROC feature vector (string $s$).

  Define search area around $ur_i$ in target image.
  $min\_cost \Leftarrow \infty$
  for each IP $ut_j$ inside search area do
    Compose graph $g_T$, extract its ROC feature vector (string $t$).
    Find minimum cost from string $s$ to $t$ using cyclic string matching and Dijkstra algorithms.
    if minimum cost$<$ $min\_cost$ then
      $min\_cost$ $=$ minimum cost.
    end if
  end for
  if $min\_cost < threshold$ then
    Push back $ur_i$ and $ut_j$ to $MR \& MT$, respectively.
  end if
end for
```

reflected in a spatial histogram. For any point $p_i$, a histogram $h_i$ of the relative coordinates of the neighbouring points is calculated using Eq. 4.8 (Mori et al., 2005):

$$h_i^m = \#\{q \neq p_i : (q - p_i) \in bin(m)\} \tag{4.8}$$

This histogram is called the SC descriptor of point $p_i$. The log-polar bins are used to make the descriptor more sensitive to the nearby points than to the farther away ones. As can be seen in Fig. 4.10, the SC descriptor is different for different points (the diamond and triangle points), while it is similar for homologous points[1] (the diamond and rectangle points). In addition, since the SC descriptor gathers coarse information extracted from the entire shape, it is relatively insensitive to the occlusion of any particular part, which makes it more robust

---

[1] corresponding points on two similar shapes.

for tracking applications. We use the same parameters as (Mori et al., 2005) for SC extraction in this thesis.

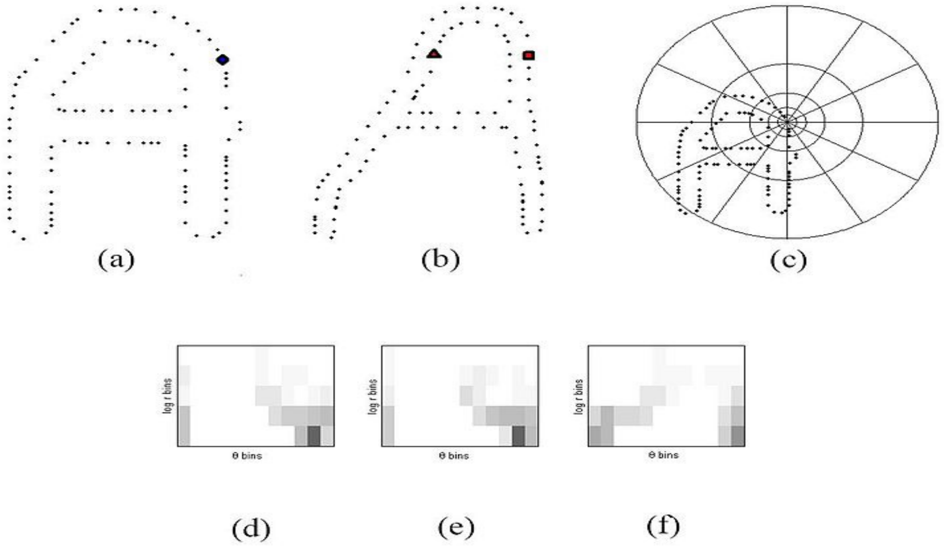


FIGURE 4.10: Shape Context feature descriptors (Mori et al., 2005).

After calculating the SC descriptor for all the unmatched IPs of $UR$ and $UT$, the matching cost calculation is accomplished. To do that and find a possible matched IP to the unmatched point $ur_i$, a search area in the target image is defined and only the unmatched IPs $ut_j$ inside this area are examined, unlike the state-of-the-art algorithms which usually measure the similarity of any IP with all the unmatched IPs in the target set. This simplification, which decreases the computational cost considerably, works due to the smoothness or small inter-frame motion assumptions (Herda et al., 2000) valid in tracking applications. Furthermore, the dynamic threshold (DT) estimation makes this idea feasible even in situation with faster movements.

On this basis, a rectangular search area is defined in the target image whose center is the position of $ur_i$ plus the displacement vector, estimated during the displacement-checking step. The more precise the displacement vector, the smaller the size the search area. To find the best matched IP to $ur_i$, all the unmatched points $ut_j$ in this search area are examined one by one. The points $ur_i$ and $ut_j$ are matched to each other if the distance measure between their SC descriptor is less than a threshold. Otherwise, $ur_i$ remains unmatched. The distance measure between two SC feature descriptors with normalized K-bin histograms $g(k)$ and $h(k)$,

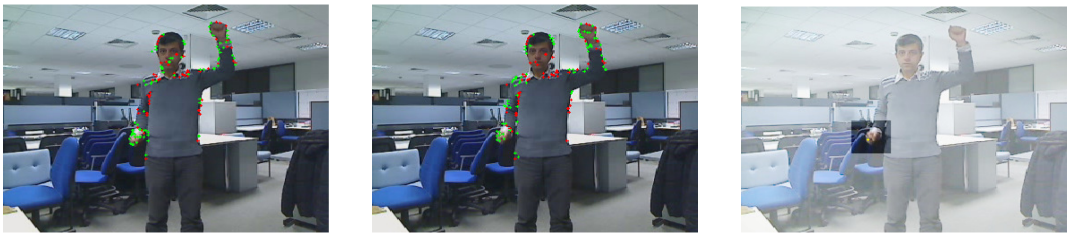
$k = 1, \ldots, K$, namely shape context cost $C_S$, ranges from 0 to 1 and is calculated using the $\chi^2$ test (Chi-squared test) (Greenwood, 1996) as follow:

$$C_S = \frac{1}{2} \sum_{k=1}^{K} \frac{[g(k) - h(k)]^2}{g(k) + h(k)} \tag{4.9}$$

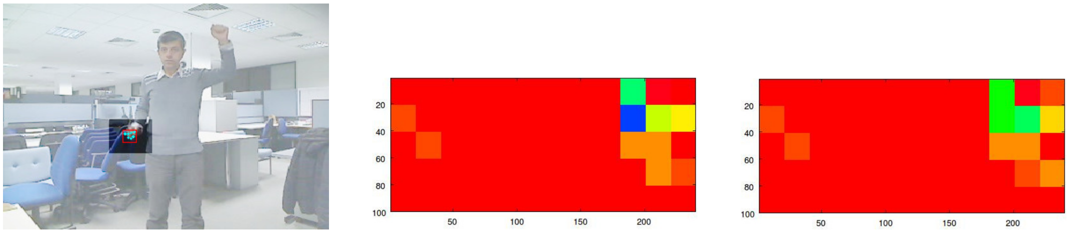
Fig. 4.11 shows the SC matching process of the spatial-based IP matching stage. Based on the size of the search area (Fig. 4.11(d)), a few possible SC feature vectors (Fig. 4.11(f)-4.11(l)) are extracted for the candidate unmatched IPs $ut_j$. These SC feature vectors are compared with the SC feature vector of the reference unmatched IP $ur_i$ (Fig. 4.11(e)) to find the best possible matched IP to it. In this example, the SC descriptor of the 6$^{\text{th}}$ $ut_j$ (Fig. 4.11(k)) has the lowest distance from the SC of the unmatched IP $ur_i$ (Fig. 4.11(e)). So, IPs corresponding to these SC descriptors are matched to each other. Algorithm 4.3 summarizes the spatial-based IP matching part of the algorithm.

**Algorithm 4.3** SC-Based Spatial-based IP Matching Algorithm

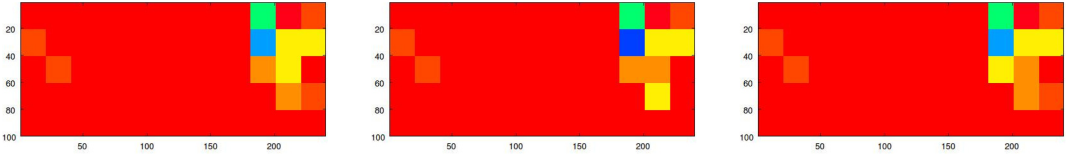
```
Input: Confidently matched IP sets $CR$ & $CT$ &
    unmatched IP sets $UR$ & $UT$.
Output: Matched IP sets $MR$ & $MT$.
Push-back $CR$ & $CT$ into $MR$ & $MT$.
Calculate the SC descriptor for all the IPs of $UR$ & $UT$ sets.
for each IP $ur_i$ of $UR$ do
    Define search area around $ur_i$ in target image using the displacement vector.
    $min\_cost \Leftarrow \infty$
    for each IP $ut_j$ inside search area do
        Calculate the $C_{S_{ij}}$ (shape context cost) between SC of $ur_i$ and $ut_j$ using equation 4.9.
        if $C_{S_{ij}} < min\_cost$ then
            $min\_cost = C_{S_{ij}}$.
        end if
    end for
    if $min\_cost < threshold$ then
        Push-back $ur_i$ and $ut_j$ to $MR\&MT$, respectively.
    end if
end for
```



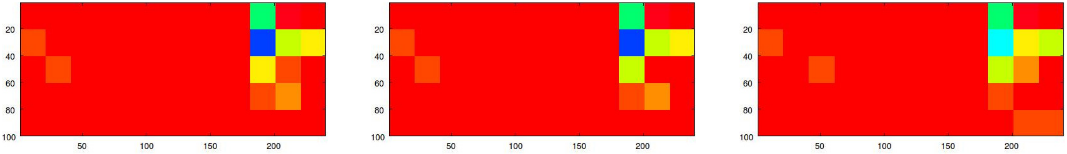
(a) Reference IPs. (b) Target IPs. (c) $ur_i$.



(d) $ut_j$ & search area. (e) SC - $ur_i$. (f) SC - $1^{\text{st}}$ $ut_j$.



(g) SC - $2^{\text{nd}}$ $ut_j$. (h) SC - $3^{\text{rd}}$ $ut_j$. (i) SC - $4^{\text{th}}$ $ut_j$.



(j) SC - $5^{\text{th}}$ $ut_j$ (k) SC - $6^{\text{th}}$ $ut_j$. (l) SC - $7^{\text{th}}$ $ut_j$.

FIGURE 4.11: SC matching process: (a) the matched and unmatched Reference IPs, red and green respectively (b) the matched and unmatched Target IPs, red and green respectively, (c) a highlighted unmatched Reference IP, yellow (d) some highlighted candidate unmatched Target IPs, cyan colour, inside the search area, red colour, (e) SC histogram of unmatched Reference IP, and (f-l) SC histogram of seven unmatched Target IPs inside the search area.

## 4.3 LSR Strategy

In this section we discuss how to track IPs associated with an articulated object. This is different from tracking the object itself because at this stage the IPs have not been connected with specific points on the object. The IPs will be used for object-pose estimation at a later stage described in Chap 5. Though the proposed LSR approach, which tries to track the trajectories of IPs over the frames, looks like a tracking algorithm by itself, it is different from the tracking and pose estimation algorithm of Chapter 5, which estimates the pose of human upper body using the IPs of the articulated parts individually. Benefits of this approach are given in Section 6.1.2.

Using IP matching to track IPs associated with an articulated object through a long sequence of frames is much more complicated than simply matching IPs of two static frames. As the object changes its pose and shape throughout the sequence, the two main problems which occur are:

- IPs in the initial frame rapidly become obsolete.
- New IPs, which were not in the previous frames, emerge.

To keep track of the IPs associated with the object throughout the frame sequence, we must find some way of removing obsolete IPs and replacing them with new IPs.

Two lists of IPs are involved in any round of matching: the Reference-list; and Target-list. The Reference-list contains those IPs in the previous frame, which we are reasonably confident represent the previous state of the object. We match these to the Target-List, which contains IPs from the current frame. Any IP of the Reference-list, which finds a matching IP in the Target-list, is replaced by the IP of the Target-list. What about the unmatched IPs of the Reference-list?

A naive approach would be to delete any unmatched Reference-list IPs on the grounds that they are now obsolete. But this would be too severe. An IP may fail to find a match in a particular round because of noise or occlusion and yet may find a match in subsequent rounds. Therefore, we should retain unmatched IPs in the Reference-list for a certain number of rounds and delete them only if they fail to find a match for several rounds in successions.

If we wish to replace deleted IPs, a naive approach would simply be to use unmatched IPs from the Target-list on the grounds that these represent new IPs generated by changes in the object. However, new IPs may also be generated by noise or occlusions. Therefore, we have to subject new IPs to a test before we admit them to the Reference-list. To do this, we include unmatched IPs of the Target-list in a third list, which we call the ”Reserved-list”. If an IP in this list finds a match over a certain number of consecutive frames then we promote it to the Reference-list. As can be seen, the unmatched IPs of the Target-list are moved into the Reserved-list for a few frames. They will be brought back if they can be matched over a certain number of consecutive frames successfully.

The LSR strategy works based on two parameters: Score ($S$); and Matching-Index ($MI$). These parameters are assigned to each IP of the Reference-list and Reserved-list at each round. The $S$ parameter reflects the success or failure of any IP through the previous rounds of matching. The $MI$ parameter, on the other hand, shows the number of times IP has been either matched or unmatched in previous rounds.

The LSR strategy comprises two stages:

- **IP scoring:** the $S$ and $MI$ parameters of each IP in the Reference-list and the Reserved-list are updated, based on the result of matching, using two empirical score values as the reward and penalty scores. Whenever an IP is matched, its $MI$ is increased by 1; otherwise it is decreased by 1. The $S$ parameter is increased by a reward score of 3, each time the IP is matched; otherwise it is decreased by a penalty score given by $MI$, the number of previous unmatched rounds. Algorithm 4.4 summarizes the IP scoring system after each round of matching.
- **List refinement:** the $S$ value of the IPs are compared with two empirical thresholds, namely the Eligibility $(E)$ and Merit $(M)$ thresholds, to find the obsolete IPs of the Reference-list and Reserved-list and the competent IPs of the Reserved-list. At each round, for each IP of the Reference-list, if $S < E$, then that IP is deleted. For each IP of the Reserved-list, if $S > M$, then that IP is promoted to the Reference-List. The IPs of the Reserved-list with $S < E$ also are deleted to prevent explosion in this list. The detail of this stage is described in Algorithm 4.5.

As an example, if an IP of the Reference-list is matched for the first time in round $k$, it receives an $S$ value of 3. If it is matched in round $k+1$, the $S$ value will go up to 6. If it is matched in round $k+2$, it will go up to 9. But, if it fails to match in round $k+3$, $S$ will go

**Algorithm 4.4** LSR strategy: IP Scoring Algorithm

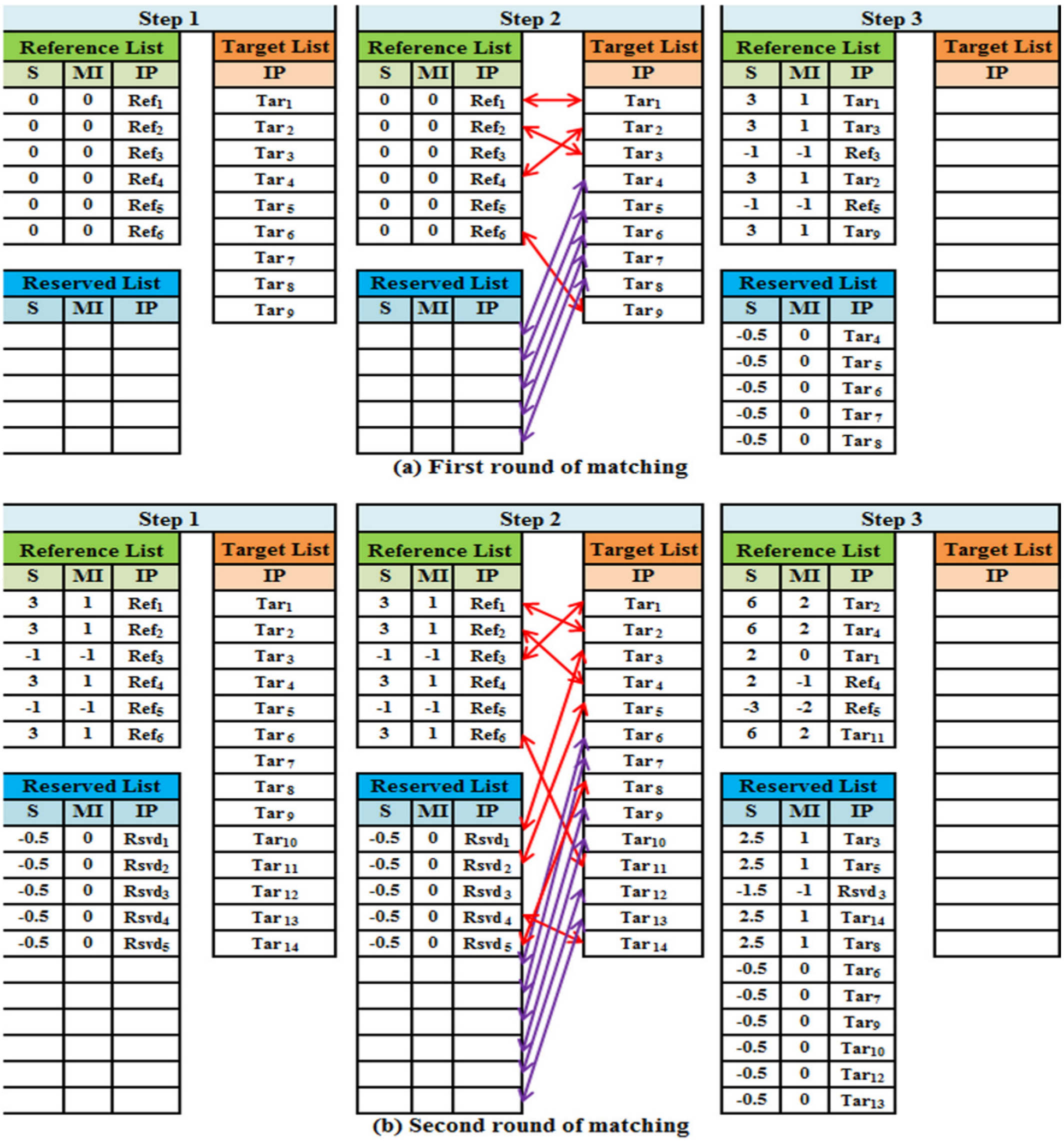
```
k: round of matching ($k \Leftarrow 1$)
for each IP $i$ in Reference-list do
    if IP $i$ matched any IP $j$ in Target-list then
        Substitute IP $i$ with IP $j$, $S_i^k = 3$, $MI_i^k = 1$
    else
        $S_i^k = -1$ & $MI_i^k = -1$
    end if
end for
for each IP $j$ in Target-list do
    if IP $j$ not-matched then
        Move IP $j$ to the Reserved-list
        $S_j^k = -0.5$ & $MI_j^k = 0$
    end if
end for
for rounds $k > 1$ do
    for each IP $i$ in Combined-list = [Reference-list Reserved-list] do
        if IP $i$ matched any IP $j$ in Target-list then
            Substitute IP $i$ with IP $j$
            if IP $i$ matched in round $k-1$ then
                $MI_i^k = MI_i^{k-1} + 1$
            else if IP $i$ not-matched in round $k-1$ then
                $MI_i^k = 1$
            end if
            $S_i^k = S_i^{k-1} + 3$
        else
            if IP $i$ matched in round $k-1$ then
                $MI_i^k = -1$
            else if IP $i$ not-matched in round $k-1$ then
                $MI_i^k = MI_i^{k-1} - 1$
            end if
            $S_i^k = S_i^{k-1} + MI_i^k$
        end if
    end for
    for each IP $j$ in Target-list do
        if IP $j$ not-matched then
            Move IP $j$ to the Reserved-list
            $S_j^k = -0.5$ & $MI_j^k = 0$
        end if
    end for
end for
```

**Algorithm 4.5** LSR strategy: List Refinement Algorithm

```
k: round of matching
for rounds $k > 1$ do
    for each IP $i$ in Reference-list do
        if IP $S_i^k < E$ (Eligibility threshold) then
            Remove IP $i$ from Reference-list
        end if
    end for
    for each IP $l$ in Reserved-list do
        if IP $S_i^k > M$ (Merit threshold) then
            Move IP $l$ to the Reference-list
        else if IP $S_i^k < E$ (Eligibility threshold) then
            Remove IP $l$ from Reserved-list
        end if
    end for
end for
```

down to 8 (because $MI = -1$). If it fails to match in round $k+4$, $S$ will go down to 6 (because $MI = -2$). However, if it matches again in round $k+5$, $S$ will go up to 9.

Fig. 4.12 shows the different steps of LSR for the first two rounds of matching. Step 1 is where the Combined-list (Reserved-list concatenated to the end of the Reference-list) and Target-list are prepared to be fed into the matching algorithm. As can be seen, the Reserved-list is empty in the first round and the $S$ and $MI$ values of the IPs are zero. Step 2 displays the status of the IPs after matching. The red arrows show the matched pairs while the purple ones show the unmatched IPs in Target-list, which are moved to the Reserved-list. This leads into Step 3, where the matched Reference-list IPs are replaced with their corresponding IPs in the Target-list and the unmatched Target-list IPs are moved to the Reserved-list with a penalty score of $-0.5$, which is a bias penalty for unmatched IPs of Target-list. This step is a basis for Step 1 in the next round of matching where: the IPs of the Reference-list and Reserved-list are relabelled with Ref and Rsvd labels; the list refinement procedure is applied to the Reference-list and Reserved-list; and the Target-list is loaded with the new IPs of target image.



**(a) First round of matching**

Step 1 – Reference List

| S | MI | IP |
|---|---|---|
| 0 | 0 | Ref$_1$ |
| 0 | 0 | Ref$_2$ |
| 0 | 0 | Ref$_3$ |
| 0 | 0 | Ref$_4$ |
| 0 | 0 | Ref$_5$ |
| 0 | 0 | Ref$_6$ |

Step 1 – Target List

| IP |
|---|
| Tar$_1$ |
| Tar$_2$ |
| Tar$_3$ |
| Tar$_4$ |
| Tar$_5$ |
| Tar$_6$ |
| Tar$_7$ |
| Tar$_8$ |
| Tar$_9$ |

Step 1 – Reserved List

| S | MI | IP |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |

Step 2 – Reference List

| S | MI | IP |
|---|---|---|
| 0 | 0 | Ref$_1$ |
| 0 | 0 | Ref$_2$ |
| 0 | 0 | Ref$_3$ |
| 0 | 0 | Ref$_4$ |
| 0 | 0 | Ref$_5$ |
| 0 | 0 | Ref$_6$ |

Step 2 – Target List

| IP |
|---|
| Tar$_1$ |
| Tar$_2$ |
| Tar$_3$ |
| Tar$_4$ |
| Tar$_5$ |
| Tar$_6$ |
| Tar$_7$ |
| Tar$_8$ |
| Tar$_9$ |

Step 2 – Reserved List

| S | MI | IP |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |

Step 3 – Reference List

| S | MI | IP |
|---|---|---|
| 3 | 1 | Tar$_1$ |
| 3 | 1 | Tar$_3$ |
| -1 | -1 | Ref$_3$ |
| 3 | 1 | Tar$_2$ |
| -1 | -1 | Ref$_5$ |
| 3 | 1 | Tar$_9$ |

Step 3 – Target List

| IP |
|---|
| |
| |
| |
| |
| |
| |
| |
| |
| |

Step 3 – Reserved List

| S | MI | IP |
|---|---|---|
| -0.5 | 0 | Tar$_4$ |
| -0.5 | 0 | Tar$_5$ |
| -0.5 | 0 | Tar$_6$ |
| -0.5 | 0 | Tar$_7$ |
| -0.5 | 0 | Tar$_8$ |

**(b) Second round of matching**

Step 1 – Reference List

| S | MI | IP |
|---|---|---|
| 3 | 1 | Ref$_1$ |
| 3 | 1 | Ref$_2$ |
| -1 | -1 | Ref$_3$ |
| 3 | 1 | Ref$_4$ |
| -1 | -1 | Ref$_5$ |
| 3 | 1 | Ref$_6$ |

Step 1 – Target List

| IP |
|---|
| Tar$_1$ |
| Tar$_2$ |
| Tar$_3$ |
| Tar$_4$ |
| Tar$_5$ |
| Tar$_6$ |
| Tar$_7$ |
| Tar$_8$ |
| Tar$_9$ |
| Tar$_{10}$ |
| Tar$_{11}$ |
| Tar$_{12}$ |
| Tar$_{13}$ |
| Tar$_{14}$ |

Step 1 – Reserved List

| S | MI | IP |
|---|---|---|
| -0.5 | 0 | Rsvd$_1$ |
| -0.5 | 0 | Rsvd$_2$ |
| -0.5 | 0 | Rsvd$_3$ |
| -0.5 | 0 | Rsvd$_4$ |
| -0.5 | 0 | Rsvd$_5$ |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Step 2 – Reference List

| S | MI | IP |
|---|---|---|
| 3 | 1 | Ref$_1$ |
| 3 | 1 | Ref$_2$ |
| -1 | -1 | Ref$_3$ |
| 3 | 1 | Ref$_4$ |
| -1 | -1 | Ref$_5$ |
| 3 | 1 | Ref$_6$ |

Step 2 – Target List

| IP |
|---|
| Tar$_1$ |
| Tar$_2$ |
| Tar$_3$ |
| Tar$_4$ |
| Tar$_5$ |
| Tar$_6$ |
| Tar$_7$ |
| Tar$_8$ |
| Tar$_9$ |
| Tar$_{10}$ |
| Tar$_{11}$ |
| Tar$_{12}$ |
| Tar$_{13}$ |
| Tar$_{14}$ |

Step 2 – Reserved List

| S | MI | IP |
|---|---|---|
| -0.5 | 0 | Rsvd$_1$ |
| -0.5 | 0 | Rsvd$_2$ |
| -0.5 | 0 | Rsvd$_3$ |
| -0.5 | 0 | Rsvd$_4$ |
| -0.5 | 0 | Rsvd$_5$ |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Step 3 – Reference List

| S | MI | IP |
|---|---|---|
| 6 | 2 | Tar$_2$ |
| 6 | 2 | Tar$_4$ |
| 2 | 0 | Tar$_1$ |
| 2 | -1 | Ref$_4$ |
| -3 | -2 | Ref$_5$ |
| 6 | 2 | Tar$_{11}$ |

Step 3 – Target List

| IP |
|---|
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

Step 3 – Reserved List

| S | MI | IP |
|---|---|---|
| 2.5 | 1 | Tar$_3$ |
| 2.5 | 1 | Tar$_5$ |
| -1.5 | -1 | Rsvd$_3$ |
| 2.5 | 1 | Tar$_{14}$ |
| 2.5 | 1 | Tar$_8$ |
| -0.5 | 0 | Tar$_6$ |
| -0.5 | 0 | Tar$_7$ |
| -0.5 | 0 | Tar$_9$ |
| -0.5 | 0 | Tar$_{10}$ |
| -0.5 | 0 | Tar$_{12}$ |
| -0.5 | 0 | Tar$_{13}$ |

FIGURE 4.12: The LSR for the first two rounds of matching.

## 4.4 Experimental Results

To evaluate the performance of the proposed IP matching algorithms, we present the results of several experiments in this section. As mentioned on page 7, due to the lack of a suitable public-domain ground-truth dataset for comparing the efficiency of the proposed algorithms of this chapter with other works, we have prepared our own test video. This video contains a person moving in front of a static camera with a range of pose changes.

### 4.4.1 Results of different stages of the Algorithm

According to the main purpose of the proposed IP matching algorithm in this chapter, which is matching the foreground IPs of consecutive frames for the sake of human upper body tracking, the extracted IPs from Red-Green-Blue (RGB) acquired images with resolution of $240 \times 320$ pixels are passed to the IP-based background subtraction algorithm of Chapter 3. Fig. 4.13 shows this preprocessing stage before applying the IP matching algorithm.

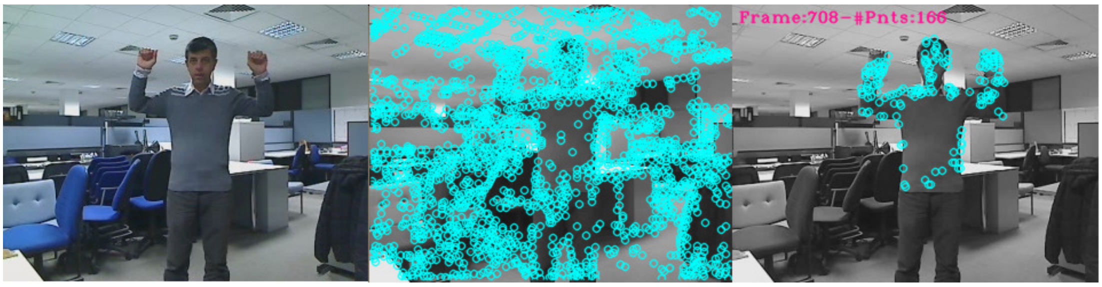


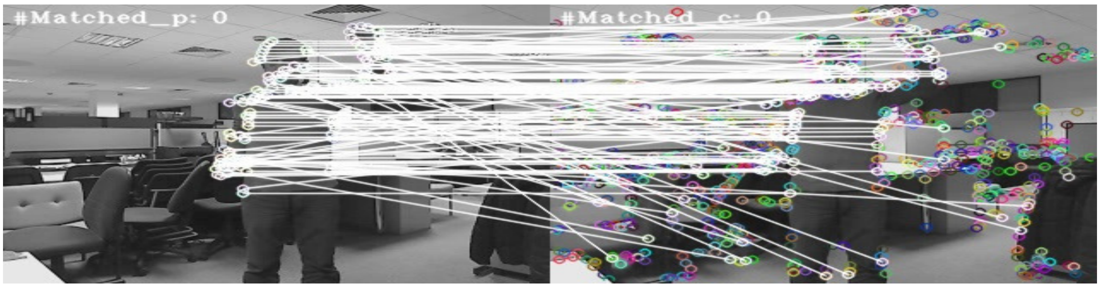
FIGURE 4.13: Preprocessing for IP matching; Left to right: image, FAST IPs, foreground IPs.

Then, the resultant foreground IPs of any two consecutive frames, are fed to the local-based stage of the algorithm, where the SURF descriptor extractor and the “BruteForce” matcher of OpenCV are used to estimate the initial correspondence. Then the cross-checking and displacement-checking procedures are applied to reject the outliers as well as to keep as many inliers as possible. This point is the end of the local stage of the algorithm and the resultant matched IPs are called the Confidently matched IPs. Fig. 4.14 shows the initial match and the outputs of the filtering steps, respectively.

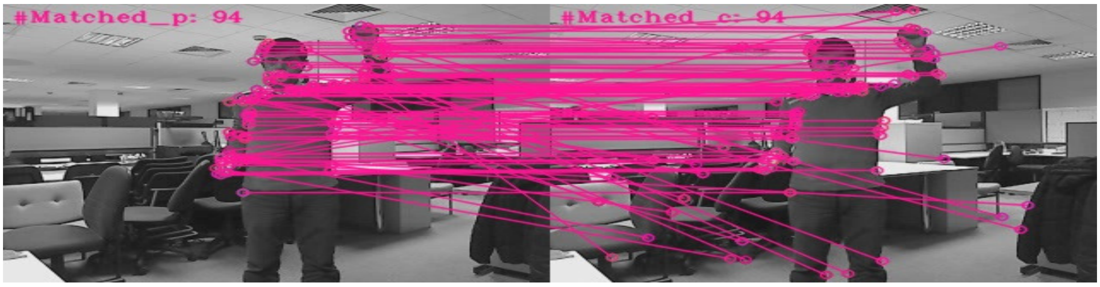
Afterwards, the spatial IP matching stage is applied to the confidently matched IPs. As described earlier, we have two different methods for spatial IP matching: Graph-based method; and SC-based. Fig. 4.15 shows and compares the result of the spatial matching approaches.

### 4.4.2 Effect of LSR on the Algorithm

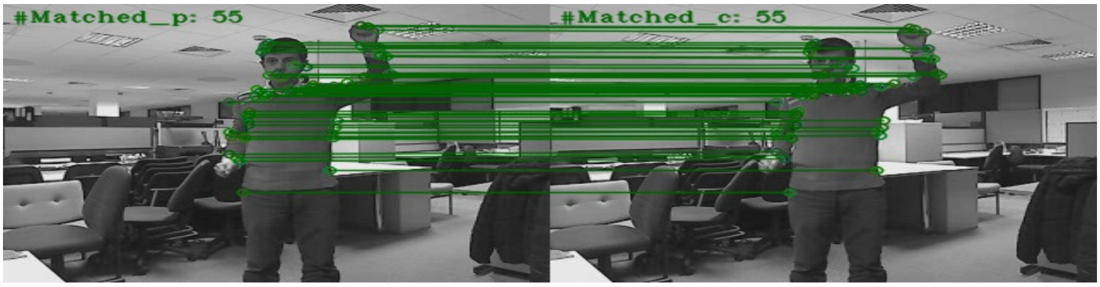
Fig. 4.16 presents the visual comparison of our algorithm ”without” and ”with” the LSR approach over eight successive rounds of matching. As can be seen in Fig. 4.16(a), the RLL



(a) Initial correspondence, IP matching using “BruteForce” matcher.



(b) Matched IPs after cross-checking.

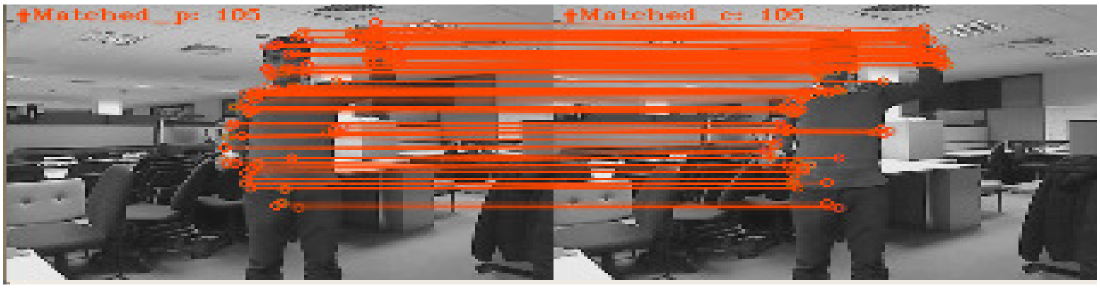


(c) Matched IPs after displacement-checking.

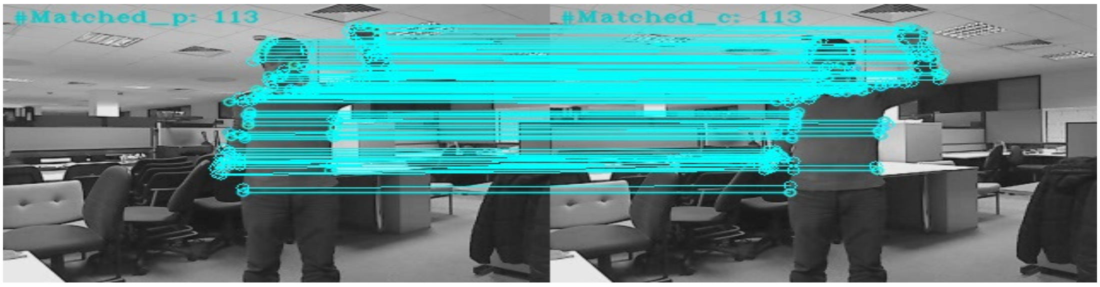
FIGURE 4.14: Results of local-based stage of the proposed IP matching algorithms: (a) “BruteForce” matching, (b) cross-checking, (c) the confidently matched IPs.

problem causes loss of track after a few rounds while the proposed LSR approach prevents it and holds the number of reference IPs at the same level as the first round, approximately. Moreover, if the matching algorithm fails to find the matched pair for many IPs, the LSR approach compensates for that in the subsequent rounds. For instance, as the fifth round of matching shows ($3^{rd}$ row and $1^{st}$ column of 4.16(a) and 4.16(b)), about half the IPs (those over the torso area) have not been matched. This is the starting point for the failure of tracking in Fig. 4.16(a), whereas the LSR has compensated for that in the next round (Fig. 4.16(b)).

Besides, LSR refines the Reference-list by removing its obsolete IPs and replacing them with new competent IPs from the Reserved-list. This advantage of the LSR approach helps the matching algorithm to follow the dynamic of the tracked object. These pros of LSR deliver a



(a) Graph-based spatial matching algorithm.



(b) SC-based spatial matching algorithm.

FIGURE 4.15: Results of spatial-based stage of the proposed IP matching algorithms: (a) Graph-based method; (b) SC-based method.

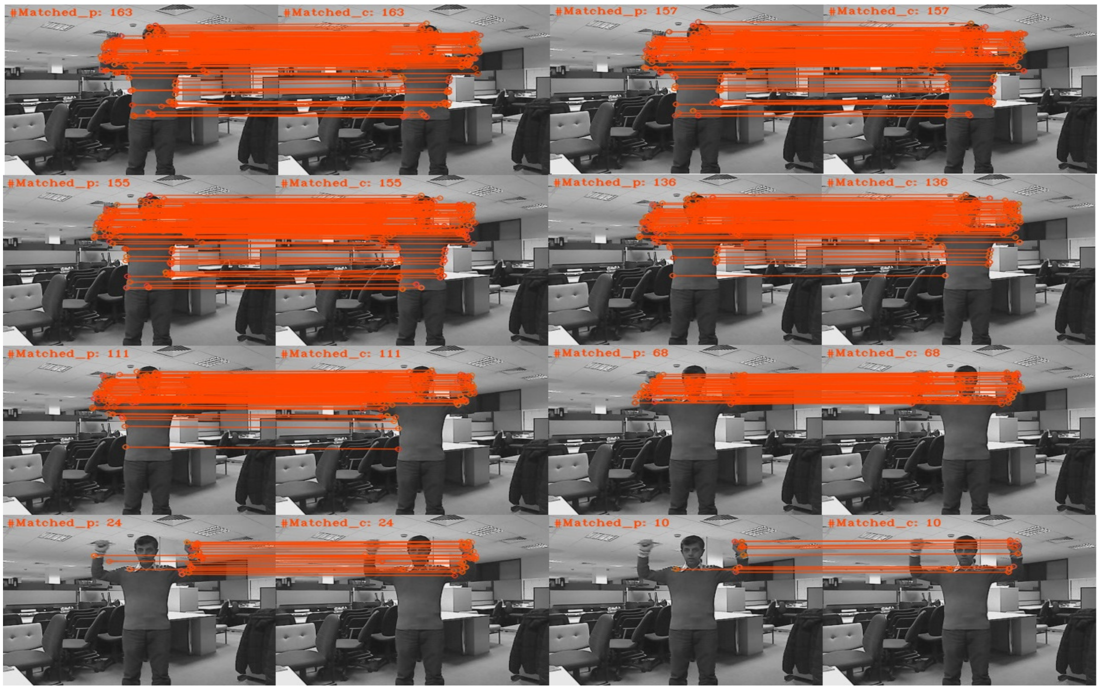
significant improvement to the IP matching algorithm particularly in articulated object tracking applications.

Fig. 4.17 also shows the matched IPs of some consecutive and non-consecutive frames for both ”without” and ”with” the LSR approach. Figs. 4.17(a) and 4.17(b) compares the effect of LSR over eight consecutive frames, while 4.17(c) shows the result for some random frames over a 100 frames of video with different level of articulation and deformations.

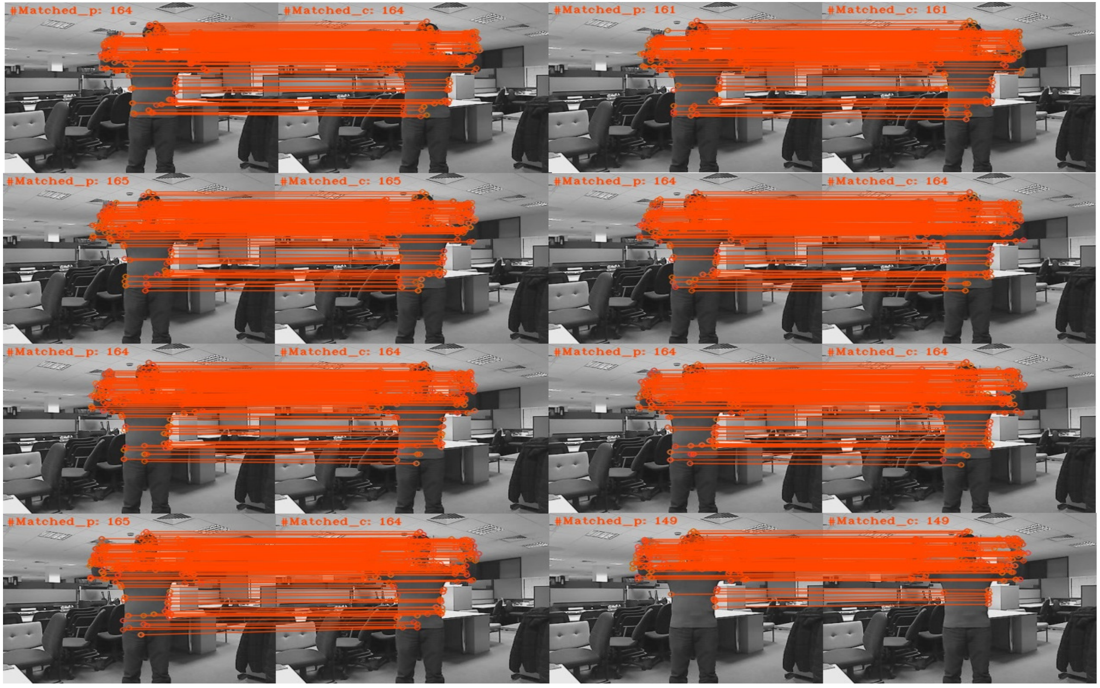
### 4.4.3 Performance in terms of Evaluation Factors

As mentioned in Section 1.2, due to lack of a suitable ground-truth dataset, we evaluate the proposed IP-matching algorithms over the test video which we have recorded for this purpose.

**Precision and recall curves**: To evaluate the IP matching algorithm, we use the same evaluation factor of Chapter 3 (Section 3.3.1) here. Figs. 4.18 and 4.19 statistically compare different stages of the proposed algorithm over 100 frames (randomly selected from 450 frames) with different levels of articulation and deformation in terms of precision and recall.

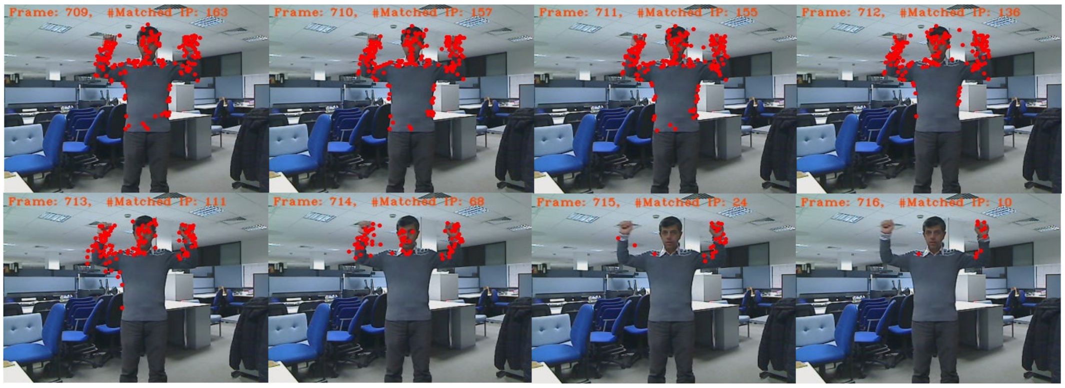


(a) Result of IP matching *without* LSR.

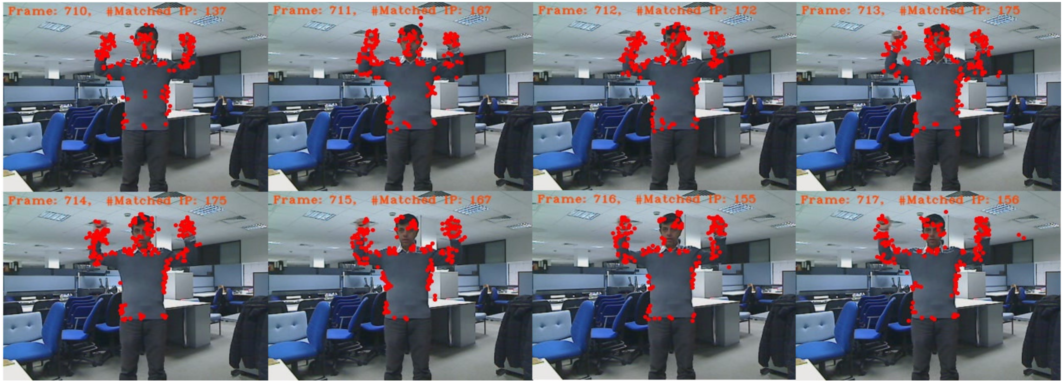


(b) Result of IP matching *with* LSR.

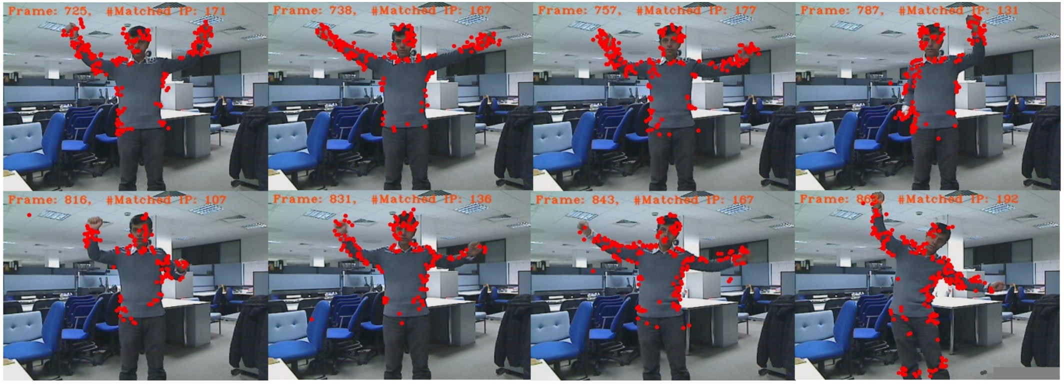
FIGURE 4.16: Final result of proposed IP matching algorithm for some consecutive frames ”without” (a) and ”with” (b) the LSR strategy.


Frame: 709, #Matched IP: 163
Frame: 710, #Matched IP: 157
Frame: 711, #Matched IP: 155
Frame: 712, #Matched IP: 136
Frame: 713, #Matched IP: 111
Frame: 714, #Matched IP: 68
Frame: 715, #Matched IP: 24
Frame: 716, #Matched IP: 10

(a) Matching ”without” LSR over 8 consecutive frames.


Frame: 710, #Matched IP: 137
Frame: 711, #Matched IP: 167
Frame: 712, #Matched IP: 172
Frame: 713, #Matched IP: 175
Frame: 714, #Matched IP: 175
Frame: 715, #Matched IP: 167
Frame: 716, #Matched IP: 155
Frame: 717, #Matched IP: 156

(b) Matching ”with” LSR over 8 consecutive frames.


Frame: 725, #Matched IP: 171
Frame: 738, #Matched IP: 167
Frame: 757, #Matched IP: 177
Frame: 787, #Matched IP: 131
Frame: 816, #Matched IP: 107
Frame: 831, #Matched IP: 136
Frame: 843, #Matched IP: 167
Frame: 86[illegible] #Matched IP: 192

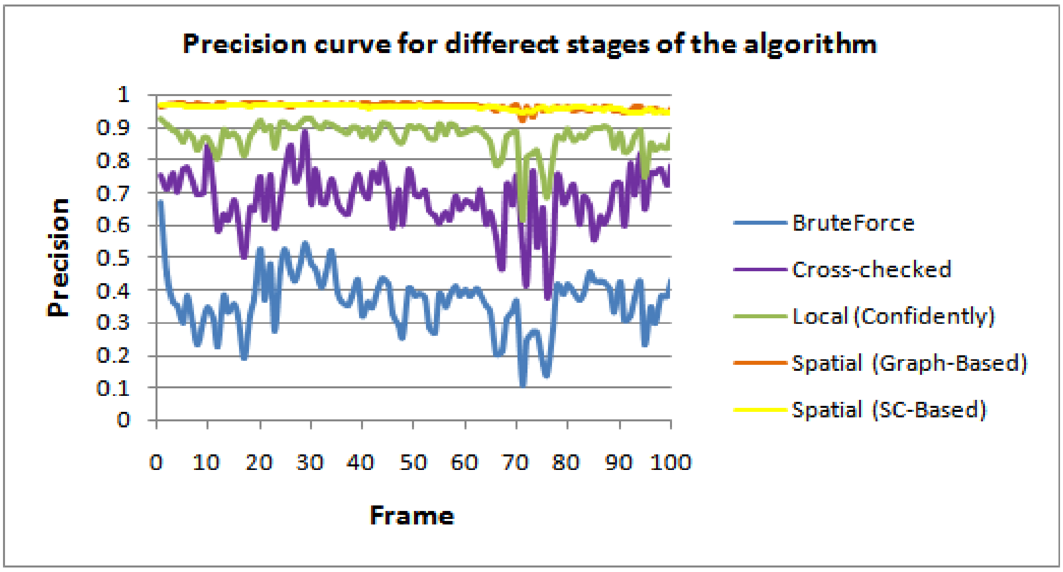
(c) Matching ”with” LSR for 8 non-consecutive frames.

FIGURE 4.17: Matched IPs of some consecutive and non-consecutive frames, ”without” and ”with” LSR.

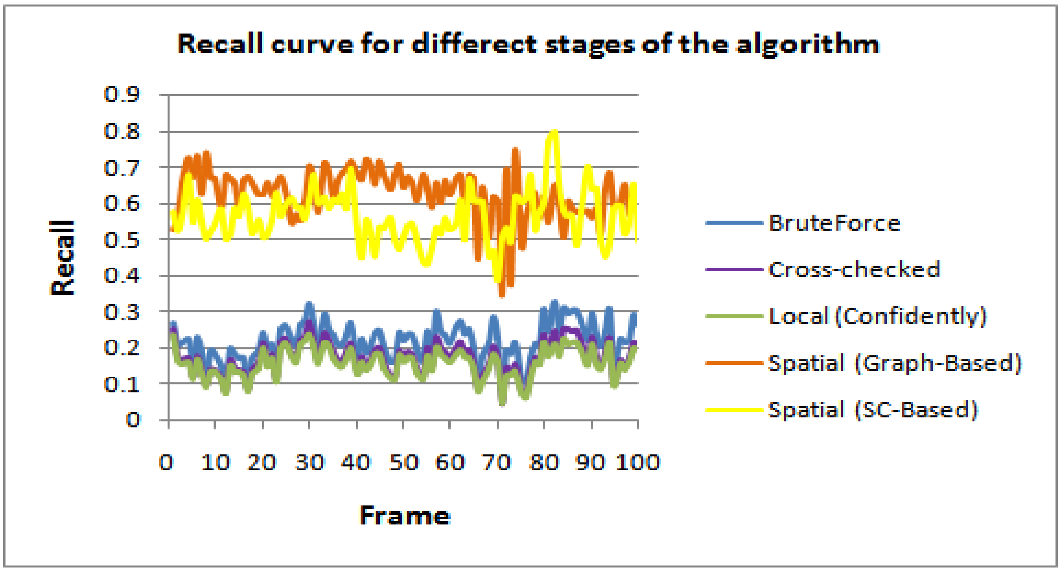
It is obvious from these figures that the proposed combined algorithm delivers the best precision and recall rates compared with the local methods. Although the precision curve of the confidently-matched stage is very close to the combined method (Fig. 4.18(a)), its recall value is quite far from it (Fig. 4.18(b)). It confirms that the local-based matching stage only delivers high accuracy (high precision) to the algorithm by filtering out the mismatched pairs, while it leaves lots of IPs unmatched. The Graph-based and SC-based methods look similar in terms of evaluation factors. Both improve the recall rate while keeping the precision rate at the same high level as the local method. As Fig. 4.18(b) shows, they outperform a little bit against each other in different frames. Sometimes, the graph-based method recovers more unmatched IPs while the SC-based outperform at other times. Moreover, the first approach looks at the spatial relationship between IPs locally by clustering them into groups and the second approach behaves globally. This is one of the reasons why we have tried two different approaches for the second stage of matching.

According to Fig. 4.18(b), the recall rate for the first stage of the algorithm is low i.e. the number of matched IPs over two consecutive frames is low. Although this would be adequate for tracking simple rigid objects with regular shapes e.g. rectangles, it is not enough for tracking more complicated objects like the articulated human body. In these situations, the reference IPs should be accurately matched to the target IPs as much as possible. In fact, Fig. 4.18 shows the capability of the proposed combined IP matching algorithm in improvement of the recall value while preserving the precision rate. The efficiency of our approach in terms of precision-recall is shown in Fig. 4.19. The output of the local-based stage of the algorithm performs roughly the same as the combined method for recall values less than 0.1. However, they are not so steady and good for the higher recall values, which it is essential for articulated object tracking.

**Performance evaluation on a round of matching**: The result of proposed algorithm on two frames of video (a round of matching) is presented in Table 4.1. In this experiment, there are 142 foreground points in the Reference-list which are matched to the Target-list IPs. As can be seen from the first row of the table, the traditional local matchers like BruteForce do not deliver good precision and recall rates. Nevertheless, the cross-checking and displacement-checking procedures improve the accuracy of the local-based IP matching stage (increasing the precision rate from 61.53% for BruteForce to 91.80% for Confidently matched IPs); meanwhile, they decrease the number of confidently matched IPs (the recall rate) from 52.33% to 40.87%. Although they pull down the recall rate (up to 40.87%), the improvement in precision



(a) Precision curve for different stages of the algorithm.



(b) Recall curve for different stages of the algorithm.

FIGURE 4.18: Precision and Recall curves of the algorithm.

(up to 91.80%) is used as a basis for the spatial-based matching stage to cut down its cost of search in comparison with the spatial-only IP matching algorithm. Finally, the last two rows of Table 4.1 show the improvement which the spatial-based stage creates in precision and recall rate. We can say the SC-based method outperforms the graph-based method overall probably because it takes into account all the IPs in the image. It is also noteworthy to compare Figs.

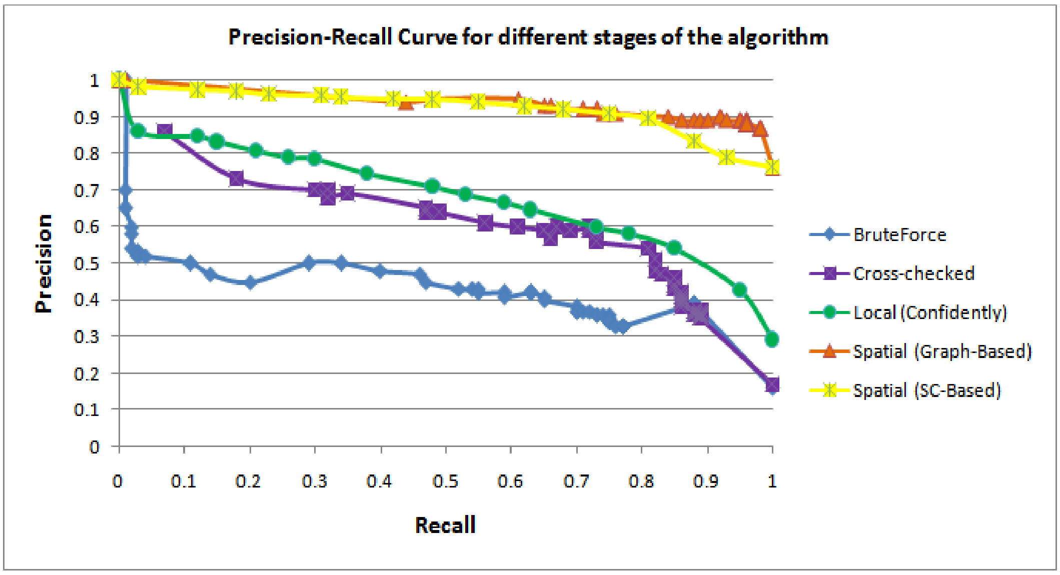


FIGURE 4.19: Precision-Recall curve of the algorithm.

4.14(c) and 4.15 to realize the delivered improvement of the combined local-spatial algorithm in comparison with the local-only IP matching algorithm. It would be good to compare the proposed algorithms with existing approaches but it is not possible for the moment because of the lack of a public-domain dataset.

The matched IPs found by this algorithm will be the input to the pose estimation algorithm in the next chapter. At the point we will estimate whether the IPs are adequate for accurate pose estimation.

TABLE 4.1: Performance comparison on the image pairs in Figs. 4.14 and 4.15.

| | **TP** | **FP** | **FN** | **Precision** | **Recall** |
|---|---|---|---|---|---|
| **BruteForce** | 55 | 68 | 13 | 44.71 | 80.88 |
| **Cross-checked** | 55 | 39 | 42 | 58.51 | 56.71 |
| **Confidently** | 55 | 5 | 76 | 91.66 | 41.98 |
| **Graph-based spatial** | 105 | 5 | 26 | 95.45 | 80.15 |
| **SC-based spatial** | 113 | 3 | 20 | 97.41 | 84.96 |

## 4.5 Concluding Remarks and Future Work

In this chapter, we discussed a new IP matching algorithm for articulated object (human body) tracking applications. The key characteristic of our approach is the increase of precision and recall rates in two sequential stages: Firstly, a local-based IP matching algorithm is performed to find the confidently matched pairs between the reference and target sets of IPs (*increasing the precision rate*); Secondly, a spatial-based matching algorithm is applied to the confidently matched pairs to recover more matched pairs from the remaining unmatched IPs (*enhancing the recall rate while the precision rate is kept at a high level*). We evaluated two different spatial-based IP matching strategies in this chapter: The graph-based method, which matches the unmatched IPs through graph matching and cyclic string matching; and the SC-based method, which uses the SC feature vector for matching. As can be seen from the experimental results, these two spatial-based IP matching strategies deliver approximately the same performance (4.19) except for the precision, for which the Graph-based method delivers a higher rate for recall values greater than 0.8. The main difference between the two methods is: the graph-based method deals with the neighbouring matched IPs of an unmatched IP locally (it considers the nearest neighbourhood of the IP), whereas the SC-based method deals globally and considers all the neighbouring matched IPs to calculate the SC descriptor of the unmatched IP.

The proposed algorithms benefit from:

- Local-based IP matching in the first stage to cut down the cost of distance and neighbourhood comparison of the spatial-based methods.
- Spatial-based IP matching to compensate for the drawback of the first stage where it fails in ambiguous situations, such as monotonous backgrounds, similar features and low resolution images.
- An IP list scoring and refinement strategy to refine the IP lists and solve the problem of RLL.

Besides, we introduced a dynamic Displacement Threshold estimation, which is calculated for any two consecutive frames, for displacement-checking. This improvement overcomes filtering out lots of good matches caused by using an absolute displacement threshold.

We applied our approach to a sequence of frames with different levels of articulation and deformations. Experimental results show, promisingly, that not only does the proposed algorithm increases the precision rate from 44.71% for BruteForce to 95.45% for graph-based and 97.41% for SC-based, but also it improves the recall rate from 52.33% for BruteForce to 80.15% for graph-based and 84.96% for SC-based. We have shown that our algorithm can obtain the foreground IPs with the mentioned accuracy. The foreground IPs are the input to the next stage of the system (as explained on page 107).

For future work, it would be worthwhile to examine a mixture of graph-based and SC-based methods to complement each other. The proposed matching algorithms go from the reference space to the target space. As another idea for future work matching in the opposite direction also can be investigated and a mixture of both can be proposed to amend the results. We should also carry out a study of the effects of different values for the parameters in the algorithms e.g. K the number of clusters in graph-based method.

# Chapter 5

# Two-Stage Hierarchical-Global Model-Based Human Upper Body Pose Estimation and Tracking using PSO

*Conquer yourself rather than the world.*

RENÉ DESCARTES.

Based on what we have done in the previous chapters, which finally gave us the foreground matched IP pairs of any two consecutive frames, this chapter deals with a two-stage Hierarchical-Global (H-G) model-based articulated human upper body pose estimation and tracking approach based on heuristic Particle Swarm Optimisation (PSO). This algorithm, which is a combined bottom-up top-down approach, estimates the skeletal pose for the current frame given the pose in the previous frame as well as the matched foreground IP pairs between the previous and current frames (see p. 104). The algorithm uses two PSO-based human body pose estimators sequentially in two different hierarchical and global ways.

In terms of their similarity, the PSO-based pose estimators hypothesize a population of particles around the potential pose vector in the current frame. Then, a set of IPs is rendered for any hypothesized particle. The number of IPs in the rendered set is same as the number of IPs in the previous frame. A discrepancy function, which calculates the distance between the IPs of the current frame and the rendered IPs of a pose hypothesis, measures the cost for that hypothesised pose. PSO, generates a random generation of particles in the first iteration, then moves them toward the optimum solution where it finds the pose with the lowest cost.

On the other hand, the two sequential PSO-based pose estimators differ from each other in terms of the way they deal with the pose vector. The pose estimator of the first stage, which is a hierarchical one, estimates the parameters of the n-D pose vector joint by joint hierarchically. In contrast, the PSO-based pose estimator of the second stage operates on the complete pose estimated of the first stage to do a consistency check and refine the estimated pose. This stage acts as a post-processing stage on the result of the first stage to compensate for the inaccurate estimated parameters of the pose vector.

The rest of this chapter is outlined as follow: firstly, our motivation for introducing the PSO-based HUB pose estimation and tracking algorithm is discussed in Section 5.1. Then, the concepts of PSO and the various factors to be considered will be demonstrated in Section 5.2. The PSO-based HUB pose estimation problem is formulated in Section 5.3. Then, the proposed two-stage hierarchical-global approach is described, together with the experimental results in Section 5.4. Finally, conclusions are discussed in Section 5.5.

## 5.1 Motivation

Articulated human body pose estimation and tracking, from either a video stream or a set of still images representing the first frame of a video sequence, is an important task in many research topics such as surveillance, motion capture, human gait analysis, medical analysis, sign language recognition and so on. The most significant challenges in human pose estimation are:

1. Diversity of human visual appearance in images.
2. Changes in lighting conditions.
3. Diversity in the physique of human body.
4. Partial occlusions due to self-occlusion or covering by other objects in the scene.
5. Complexity of human body kinematic structure.
6. High dimensionality of the pose.
7. The lack of 3d information caused by capturing the world in 2D.

As discussed earlier in Chapter 2, many approaches have been presented on this topic focusing on different combinations of observation-representation, matching, and pose-estimation to address the above mentioned challenges. Recently the Kinect has provided a solution to this problem in certain circumstances. But, as discussed on p. 39, there are situations where it is not applicable e.g. wearable devices. However, when monocular vision-based articulated human body pose estimation and tracking is carried out with ordinary low-resolution cameras, it is still an unsolved problem with some open ill-posed difficulties due to high dimensionality, complex motion, occlusion and so on (Moeslund et al., 2006).

Among the different approaches, introduced for human body pose estimation and tracking, a category has been devoted to the use of evolutionary optimization methods such as the genetic algorithm, particle swarm optimization, and the imperialist competitive algorithm[1] (John et al., 2010; Kwolek et al., 2012; Li and Sun, 2013; Zhang et al., 2008, 2010) due to their simplicity as well as ability in solving complex highly non-linear high-dimensional optimisation problems and finding the global optimum. Despite the vast range of applications which evolutionary optimization methods have been investigated for, Poli (2007) reports for example that only 9% of PSO applications are devoted to image and video analysis applications. This

---

[1] Imperialist Competitive Algorithm: `http://en.wikipedia.org/wiki/Imperialist_competitive_algorithm`

implies that the applications of evolutionary optimization methods, particularly PSO, to computer vision are not examined as much as the other areas. However, the promising results on the evaluated areas shows there is yet high-potential to be investigated.

Particle swarm optimization, which is a nature-inspired population-based meta-heuristic algorithm (Beheshti and Shamsudding, 2013), has been gaining popularity in the computer vision application domain (Saini et al., 2013). In this regard, a good deal of research has been conducted on using PSO in different tasks in computer vision such as object recognition (Perlin et al., 2008), image clustering (Omran et al., 2005), articulated hand tracking (Oikonomidis et al., 2012), object tracking (Cheng et al., 2014), multi-object tracking (Zheng et al., 2014), and human body tracking (Lei et al., 2013), etc. Different types of observation representation methods along with PSO have been investigated for these tasks. Ivekovič et al. (2008) and John et al. (2010) use silhouette as the image observations. Oikonomidis et al. (2012), in contrast, use depth data for hand pose tracking using PSO.

Owing to these, we decided to investigate interest points as the most light-weight method of object representation in the task of human body pose estimation and tracking using PSO in a monocular fashion in 2D. The main difference of our method with similar PSO-based approaches is in the way we compare the particle solutions of PSO with real observations. Most other approaches generate synthetic observation data independently from the observations in the previous and current frames to compare them with the real observation. This is not possible in our method for the interest points because IPs comes from the appearance and texture of the object as well as the scene. Thus, we generate the synthetic data using the relationship between the images and observation in the previous and current frames. This is probably the *central contribution* of our work in this research. Through this idea, we will try to address the above mentioned challenges as much as possible.

**Summary of the idea:** In this chapter we propose a new two-stage hierarchical-global model-based human upper body pose estimation and tracking algorithm using interest points in a particle swarm optimization framework. In the first stage, the pose vector in the current frame is estimated using a hierarchical PSO-based articulated human upper body pose estimation strategy. Then, to perform a consistency check and compensate for the inaccurate estimated parameters of the pose vector, another PSO-based pose estimator is applied to the estimated pose of the first stage. The proposed algorithms benefit from:

- A hierarchical model-based pose estimation method (HPSO) in the first stage to cut down the complexity and computational cost of a high-dimensional optimisation problem.
- A post-processing pose refinement method (GPSO) to compensate for the spatial and temporal propagation errors caused by the first stage.

## 5.2 Particle Swarm Optimisation

Particle swarm optimization is a heuristic global optimization method, proposed by Kennedy and Eberhart (1995). This method, which was inspired by the social behaviour of animals and biological populations, is a technique used to explore the search space of a given problem to find the parameters which optimizes a particular objective function. The objective function could be either the cost or the fitness function and the PSO attempts to minimize or maximize it, respectively. Without loss of generality, in this chapter, we assume that the objective function here is the cost function which is going to be minimized. Although PSO was originally proposed to solve the optimisation problem of continuous non-linear problems, where both the search space and decision variables are continuous, several discrete versions have been proposed afterwards (Kashan and Karimi, 2009; Tasgetiren et al., 2007; Wang et al., 2009). As will be discussed in Appendix B, the phrase ”particle” causes the reader confusedly to think that PSO is an implementation of a Bayesian filter, similar to the Particle Filter (PF). Nevertheless, this is a misconception and the concept of PSO is completely different.

The PSO algorithm starts with generating a random population of candidate solutions, called the particles, within the search space. The particles, which are represented by their positions and velocities, are evaluated by a cost function throughout the iterations. Based on the result of evaluation, they change their position around the search space to approach to a local or global minimum (usually a global one), where it minimizes the cost function.

To do this, the algorithm records the best minimum cost, i.e. the *personal best cost*, and its corresponding position, i.e. the *personal best position*, which each particle has achieved so far during the iterations. In addition, the best cost value achieved over all the particles in the population, is remembered and called the *global best cost*. The position corresponding to the global best cost, which is named the *global best position*, is recorded too. The particles move during the iterations based on their personal best and the global best. This process is repeated until some stopping condition is met (Van Den Bergh, 2006).

In this section we will go through the essential aspects of the PSO. The complementary discussions will be introduced in Appendix B.

### 5.2.1 PSO Notations and Optimisation Process

Following the notation introduced by John et al. (2010), the elements of PSO are defined as follows:

- $S$: the underlying n-dimensional search space, where $S \subseteq R^n$.
- $N$: number of particles, each particle represents a candidate solution in the search space.
- $f$: the cost function, $f : S \to R$.
- $\mathbf{X}^i$: the position of $i^{th}$ candidate, where $\mathbf{X}^i = (x_1^i, x_2^i, \ldots, x_n^i) \in S$.
- $\mathbf{a}, \mathbf{b}$: constraint vectors, where $\mathbf{a} \leq \mathbf{X}^i \leq \mathbf{b}$ and $\mathbf{a}, \mathbf{b} \subseteq R^n$.
- $\mathbf{V}^i$: the velocity of $i^{th}$ candidate, where $\mathbf{V}^i = (v_1^i, v_2^i, \ldots, v_n^i) \in S$.
- $\mathbf{P}^i$: the personal best position of $i^{th}$ candidate, where $\mathbf{P}^i = (P_1^i, P_2^i, \ldots, P_n^i) \in S$.
- $p_{best}^i$: the personal best cost of $i^{th}$ candidate, $p_{best}^i = f(\mathbf{P}^i)$.
- $\mathbf{g}$: the global best position among all the particles.
- $g_{best}$: the global best cost, $g_{best} = f(\mathbf{g})$.

On this basis, the PSO goes through the following two steps to solve the optimisation problem:

1. **Initialization**: This step initializes the PSO by generating a random population of $N$ particles $\mathbf{X}^i$ within the search space and satisfying the constraint vectors. The velocities $\mathbf{V}^i$ of the particles are selected randomly so that every $v_k^i \in [-1,1]$. Then, the cost function $f$ evaluates the particles and sets the $p_{best}^i = f(\mathbf{P}^i)$. In addition, the particle with the minimum cost value determines the global best position $\mathbf{g}$ and the global best cost $g_{best}$.

2. **Repeat**: This is the process which is repeated until a criterion is met, i.e., either the number of maximum iterations is achieved or the global best cost $g_{best}$ satisfies a minimum threshold. Given the status of the particles in iteration $t$, i.e., $\mathbf{X}^i$, $\mathbf{V}^i$, $\mathbf{P}^i$, and $p_{best}^i$, as well as the global best cost and position at this iteration, the following sub-steps are repeated:

    (a) ***Velocity update***: the velocity vector is updated based on three different vectors according to the following formula (***magenta*** arrow in Fig. 5.2).

    $$\mathbf{V}_{t+1}^i = \omega \mathbf{V}_t^i + c_1 rand_1()(\mathbf{P}^i - \mathbf{X}_t^i) + c_2 rand_2()(\mathbf{g} - \mathbf{X}_t^i) \tag{5.1}$$

    parameters $\omega$, $c_1$, $rand_1()$, $c_2$, $rand_2()$ and their effect on the convergence of the algorithm will be discussed later in Subsection 5.2.2.

    (b) ***Position update***: the position of the particle $i$ is updated using the updated velocity vector by:

    $$\mathbf{X}_{t+1}^i = \mathbf{X}_t^i + \mathbf{V}_{t+1}^i \tag{5.2}$$

    (c) ***Position constraint check***: the updated position vectors $\mathbf{X}_{t+1}^i$ are checked against the constraint vectors, which are the boundaries of the search space, to ensure:

    $$\mathbf{a} \leq \mathbf{X}_{t+1}^i \leq \mathbf{b} \tag{5.3}$$

If the position vector of any particle violates the constraint vectors in some dimensions, the corresponding entries of those dimensions in the position vector are easily set to the boundary values. In addition, the velocity vector values of those dimensions are reversed. Fig. 5.1 shows the position constraint check process for particle $i$ over 4 iterations in a $2D$ search space. As can be seen, $\mathbf{X}_2^i$ is going to be outside the search space boundaries after the update. So, the position constraint check brings it back to the boundary. For the next iteration, $\mathbf{V}_3^i$ wants to follow the direction of movement in the previous iteration, which has pushed the particle outside the boundaries. To avoid this happening again, the velocity vector has to be corrected. If we decompose the velocity vector into its components in the $x$, $y$ directions (the cloud in Fig. 5.1), we can see the problem is the component in the $x$ direction. So, it is reversed and the new velocity vector $\mathbf{V}_3^i$ is formed.

(d) ***Personal and global best update***: for each particle, the personal and global best costs and positions are updated according to the following equations:

$$\mathbf{P}^i = \mathbf{X}_{t+1}^i \quad and \quad p_{best}^i = f(\mathbf{X}_{t+1}^i) \quad if \quad f(\mathbf{X}_{t+1}^i) < p_{best}^i \tag{5.4}$$

$$\mathbf{g} = \mathbf{P}^i \quad and \quad g_{best} = p_{best}^i \quad if \quad p_{best}^i < g_{best} \quad \forall i \tag{5.5}$$

### 5.2.2 PSO Parameters

To describe how the PSO parameters affect the algorithm, we need to understand two different concepts: *Exploration* and *Exploitation*. Exploration means providing the algorithm with the freedom to search and to explore wherever it likes. Random search has the highest degree of exploration. In contrast, exploitation enables the algorithm to exploit the current solutions which the algorithm has achieved so far. Local search is a good example of exploitation. The main superiority of the PSO algorithm, which is its capability in finding the global optimum solution, is due to the fact that it has a mixture of both these concepts. Therefore, it is important to tune the parameters to gain the best performance as much as possible.

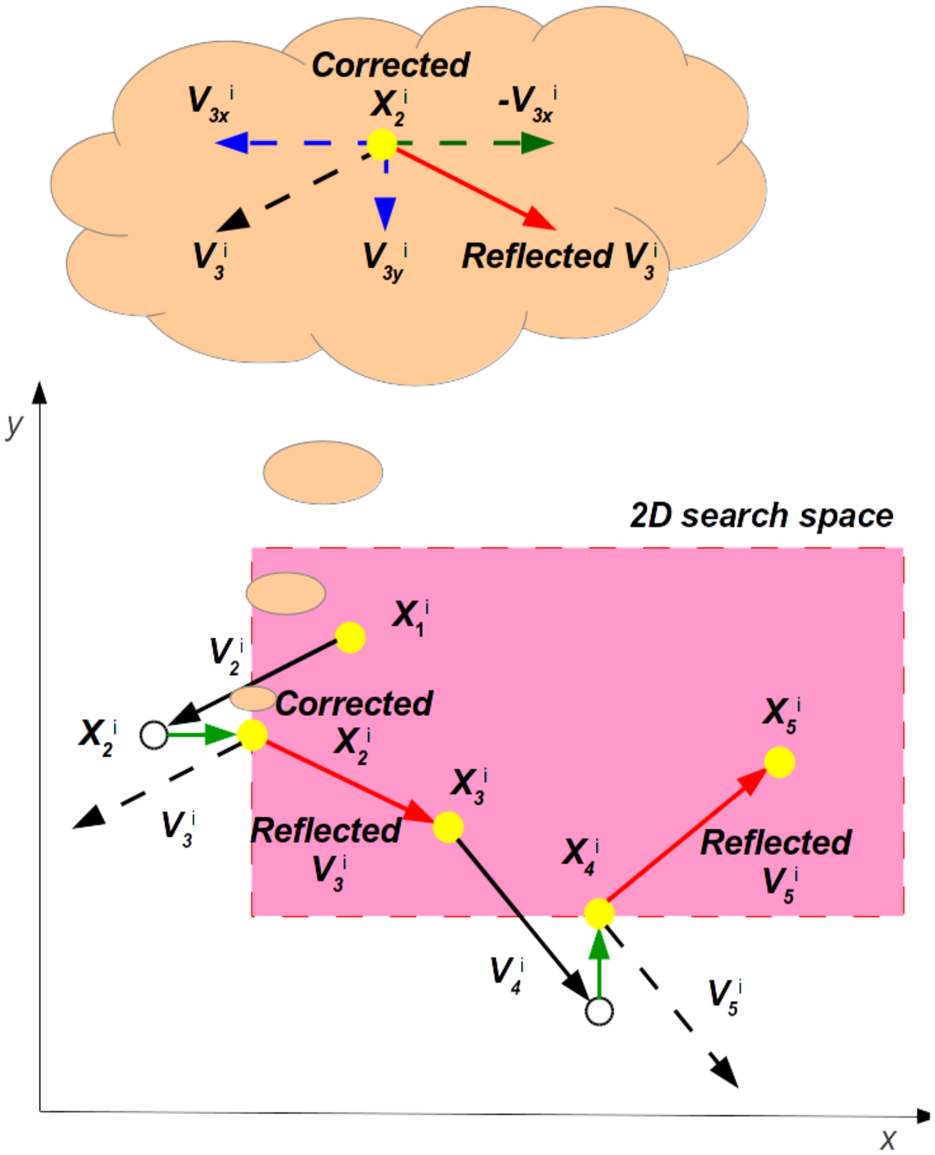


FIGURE 5.1: Position constraint process over 4 iterations for particle *i*.

The parameters $rand_1()$ and $rand_2()$ are uniform random variables, which create a stochastic influence on the velocity update. Parameters $c_1$ and $c_2$ are called the cognitive and social coefficients, respectively, and $\omega$ is called the inertia weight coefficient. These parameters are selected according the following criteria (Shi and Eberhart, 1998):

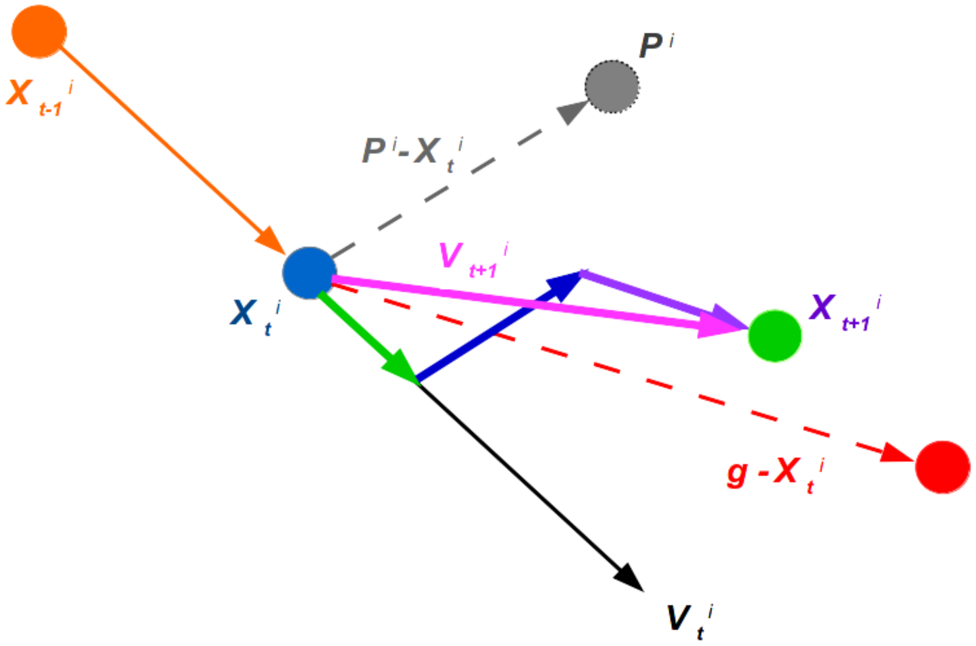


FIGURE 5.2: PSO optimisation process in iteration $t$.

$$rand_1()\&rand_2() \in [0,1] \quad , \quad \omega \in [0.4, 0.9] \quad , \quad c_1\&c_2 \in [0,2] \tag{5.6}$$

#### 5.2.2.1 Velocity Update Terms

As Fig. 5.2 shows, each of the three terms of Equation 5.1 affects the velocity update in a different way as follows:

- ***Inertia component***: The first term, $\omega\mathbf{V}_t^i$ is governed by the inertia weight parameter $\omega$, which attempts to keep the particle moving in the same direction as it was heading in the previous iteration (***green*** arrow in Fig. 5.2). Bansal et al. (2011) ran a set of experiments to evaluate 15 different inertia weights over 5 optimization test problems. As Table 5.1 shows, the constant and linear decreasing inertia weights are the best choices in terms of minimum cost value while the constant one is the worst inertia weight strategy when the number of iterations, which increases the computational cost of the algorithm, is important. As can be seen from Table 5.1, the best and worst inertia weight strategies have been found over 5 specific optimisation problems. So, they depend on the context.

TABLE 5.1: Summary of results of experiment on inertia weight over 5 optimization test problems (Bansal et al., 2011).

| **Criterion** | **Best Inertia Weight Strategy** | **Worst Inertia Weight Strategy** |
|---|---|---|
| **Average Error** | Chaotic Inertia Weight | Chaotic Random Inertia Weight |
| **Average Number of Iterations** | Random Inertia Weight | Constant Inertia Weight |
| **Minimum Error** | Constant Inertia Weight<br>Linear decreasing Inertia Weight | Chaotic Random Inertia Weight<br>Global-Local Best Inertia Weight |

In the constant inertia weight strategy, the selected value for $\omega$ can either dampen or accelerate the particle's inertia (Shi and Eberhart, 1998). Generally, the lower the inertia weight, the higher the exploitation and consequently the faster the convergence of PSO. In contrast, the higher the inertia weight, the more comprehensive the search of the entire search space, which implies a higher level of exploration. The linear decreasing strategy, on the other hand, starts with a constant value and linearly decreases it according to the following formula (Xin et al., 2009). It causes a faster convergence and a lower number of iterations to be delivered by this strategy.

$$\omega_t = \omega_{max} - \frac{\omega_{max} - \omega_{min}}{iter_{max}} \times t \tag{5.7}$$

- ***Cognitive component***: the second term, $c_1 rand_1()(\mathbf{P}^i - \mathbf{X}_t^i)$, is the particle's memory and pushes it toward the regions of the search space in which it has experienced the lowest personal cost (***blue*** arrow in Fig. 5.2). The cognitive coefficient $c_1$, which is selected usually close to 2, influences the size of the step the particles take toward their personal best position.

- ***Social component***: the last term, $c_2 rand_2()(\mathbf{g}_t - \mathbf{X}_t^i)$, makes the particle move toward the global best position, to which the swarm has approached up to iteration $t$ (***purple*** arrow in Fig. 5.2). The social coefficient $c_2$ is similarly selected close to 2, and influences the size of the step the particles take toward the global best position.

#### 5.2.2.2 Constraint Coefficients

Clerc and Kennedy (2002) presented a more sophisticated way of selecting the PSO parameters by applying some constraints on them. To do this, they defined constant parameters $\phi$, $\phi_1$, and $\phi_2$ so that:

$$\phi_1, \phi_2 > 0 \quad and \quad \phi \triangleq \phi_1 + \phi_2 > 4 \tag{5.8}$$

Accordingly, the parameter $\chi$ is defined as follows:

$$\chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}} \tag{5.9}$$

Based on these constants, the PSO parameters are defined as:

$$\begin{cases} \omega = \chi \\ c_1 = \chi\phi_1 \\ c_2 = \chi\phi_2 \end{cases} \tag{5.10}$$

The best values for the constraint parameters are gained for $\phi_1 = \phi_2 = 2.05$, which yields $\omega = 0.7298$ and $c_1 = c_2 = 1.4962$ for the PSO parameters. In this thesis, we selected this method along with the linear decreasing strategy for the inertia weight coefficient with damping factor $\omega_{damp} = 0.99$ (Shi and Eberhart, 1998).

#### 5.2.2.3 Velocity Clamping

To keep the particles from moving too far beyond the search space, a technique called *velocity clamping* is used to restrict the maximum velocity of each particle $\mathbf{V}^i_{max} = \alpha(\mathbf{b} - \mathbf{a})$. The value $\alpha$ is the velocity clamping factor so that $0.1 \leq \alpha \leq 1.0$. We use $\alpha = 0.1$ in our work.

## 5.3 PSO-Based HUB Pose Estimation Formulation and Prerequisites

In this section we formulate the HUB pose estimation problem and describe how the aforementioned PSO framework is adapted to our pose estimation and tracking problem. Moreover, we demonstrate the HUB model that represents the human upper body pose. The process of IP rendering for the pose hypotheses and the scaling factor estimation are addressed afterwards. Finally, we define and detail the cost function which we need to evaluate the hypothesised poses in the PSO framework.

### 5.3.1 Problem Formulation

Given the foreground matched IP pairs of any two consecutive frames, as well as the HUB pose of the model in the previous frame, the mission of our pose estimation algorithm in this thesis is to estimate the HUB pose in the current frame. Since a pose is represented by an $n$-dimensional vector $\mathbf{X} = (x_1, x_2, \ldots, x_n)$ (as will be described in Subsection 5.3.2), estimating the pose here is equivalent to searching the $n$-dimensional space $S$ for a point $\mathbf{X}$ which optimizes the cost function. Owing to the capability of the PSO algorithm in finding the global optimum in search problems (Saini et al., 2013), we are going to use it to solve our pose search problem.

To do this, the algorithm first hypothesizes a limited number of solutions (poses) around the potential position in the current frame. The mentioned potential position could be found simply using a Kalman filter predictor. However, it might be acceptable to consider the previous pose as the potential position under smoothness or small inter-frame motion assumptions, which are valid to assume in human body tacking (Herda et al., 2000). Then, the discrepancy between each hypothesized pose and the real observation data of the current frame, which are the IPs here, is measured. To make this comparison feasible, each pose hypothesis needs to have commensurate data. On this basis, a set of IPs, equal in number to the set of IPs in the real observation, is generated for any hypothesized pose. On one hand, the relationship between the pose hypothesis and its generated IPs should be similar to the relationship of the pose in the previous frame and the foreground IPs of the frame. On the other hand, there is presumably a deformation or transformation between the foreground matched IPs of the previous and current frames. This deformation should be applied to the generated IPs of the pose hypothesis. Otherwise, it would make no sense to compare the hypothesis and the real data fairly. Finding a way to generate these IPs, which satisfies these requisites, is a challenge and will be discussed in Subsection 5.3.3.

The assumption here is that when a hypothesized pose matches closely the pose of the person in the current frame, its generated IPs also match closely the foreground IPs of the current frame. Therefore, the process of estimating the HUB pose, which is formulated as finding the pose which minimizes discrepancy between the generated IPs of the pose hypothesis and the real IPs of the current frame, needs a discriminative discrepancy function. This challenge, which will be addressed in Subsection 5.3.5, determines the cost of estimation for each hypothesised pose. As an example, the Euclidean distance between the IPs of the current frame and the generated IPs of a pose hypothesis could be defined as a cost function.

The PSO is an iterative algorithm which starts by generating a random population of particles (pose hypotheses) in the first iteration. Then it updates the position of the particles in the space iteratively until either a minimum number of iterations is reached or a minimum cost value is gained. The particle which creates the minimum cost represents the estimated pose in the current frame. Owing to the high dimensionality of the pose space (Subsection 5.3.2) and also the iterative behaviour of the PSO, the computational expense of the pose estimation problem is another concern, which is handled in this thesis through a *divide-and-conquer* strategy by decomposing the parameters of the pose vector and estimating them separately.

For the sake of simplicity, we introduce some notation and definitions as follows:

1. $P^p$: pose in the previous frame, which is known.
2. $P^h_k$: $k^{th}$ pose hypothesis, which is generated and optimised by PSO.
3. $P^c$: pose in the current frame, which is going to be estimated.

4. $\pi_i^p$, $\pi_{ki}^h$, $\pi_i^c$: An IP of the previous frame, the pose hypothesis $P_k^h$, and the current frame respectively.
5. $\Pi^p$: the foreground IPs of the previous frame so that $\Pi^p = \{\pi_i^p\}_{i=1}^N$.
6. $\Pi_k^h$: the rendered IPs of the pose hypothesis $P_k^h$ so that $\Pi_k^h = \{\pi_{ki}^h\}_{i=1}^N$.
7. $\Pi^c$: the foreground IPs of the current frame so that $\Pi^c = \{\pi_i^c\}_{i=1}^N$.
8. $\{l_j\}_{j=0}^7$: limb $j$ of the skeletal model (Table 5.2).
9. $\{\Theta_j\}_{j=0}^7$: the limb distance thresholds.
10. $C_j^p$: cluster $j$ of $\Pi^p$, labelled to limb $l_j^p$.
11. $C_j^c$: cluster $j$ of $\Pi^c$, labelled to limb $l_j^p$.
12. $C_j^{kh}$: cluster $j$ of $\Pi_k^h$, labelled to limb $l_j$.
13. $V_j^p$: the virtual foreground IPs generated for limb $l_j^p$.
14. $V_j^c$: the virtual foreground IPs generated for limb $l_j^c$.
15. $\alpha(\pi_i)$: the matching confidence coefficient between $\pi_i^p$ and $\pi_i^c$.

### 5.3.2 Modelling the Human Upper Body

To describe the top half part of the human body for the purpose of model-based HUB pose estimation and tracking, we need to define a suitable model first. As discussed in Chapter 2, different types of models can be used in this regard. The skeletal model, which can be used in $2D$ and $3D$ spaces, is one of the most efficient because it is one of the most light-weight and yet can represent the essential information for the problem in hand.

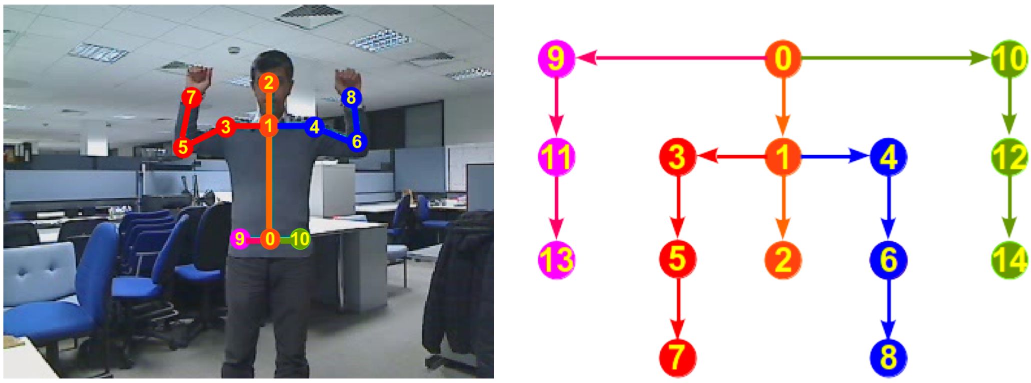
The employed HUB *skeletal model* decomposes the upper body into 7 rigid body parts, including torso, neck, head, two upper arms, and two lower arms (Fig. 5.6). Owing to the low importance of the hands in human body tracking and their complexity in implementation, we do not include them in our model. It should be noticed that our implementation of the model

(Appendix A) covers the whole body (with 11 rigid parts), of which we use only its upper section in this chapter in accordance with the purpose of this thesis. The kinematic constraints have been considered in the implementation of the model by applying constraints on the range of angles for joints of the model to prevent infeasible poses. Complementary information about the whole body model and the implementation procedures can be found in Appendix A (Kinematic Human Body Model).

TABLE 5.2: Kinematic HB model; the joints and the links.

| **Section** | **Joints** | | | **Limbs** | | |
|---|---|---|---|---|---|---|
| | **ID** | **Name** | **Acronym** | **ID** | **Name** | **Acronym** |
| | 0 | Waist | $\mathcal{WST}$ | $l_0$ | Waist → Centre of shoulder | $\mathcal{WST2CSH}$ |
| **Upper Body** | 1 | Centre of shoulder | $\mathcal{CSH}$ | $l_1$ | Centre of shoulder → Head | $\mathcal{CSH2HED}$ |
| | 2 | Head | $\mathcal{HED}$ | $l_2$ | Right shoulder → Left shoulder | $\mathcal{RSH2LSH}$ |
| | 3 | Right shoulder | $\mathcal{RSH}$ | $l_3$ | Right shoulder → Right elbow | $\mathcal{RSH2REB}$ |
| | 4 | Left shoulder | $\mathcal{LSH}$ | $l_4$ | Left shoulder → Left elbow | $\mathcal{LSH2LEB}$ |
| | 5 | Right elbow | $\mathcal{REB}$ | $l_5$ | Right elbow → Right wrist | $\mathcal{REB2RWR}$ |
| | 6 | Left elbow | $\mathcal{LEB}$ | $l_6$ | Left elbow → Left wrist | $\mathcal{LEB2LWR}$ |
| | 7 | Right wrist | $\mathcal{RWR}$ | | | |
| | 8 | Left wrist | $\mathcal{LWR}$ | | | |
| **Lower Body** | 9 | Right hip | $\mathcal{RHP}$ | $l_7$ | Right hip → Left hip | $\mathcal{RHP2LHP}$ |
| | 10 | Left hip | $\mathcal{LHP}$ | $l_8$ | Right hip → Right knee | $\mathcal{RHP2RKN}$ |
| | 11 | Right knee | $\mathcal{RKN}$ | $l_9$ | Left hip → Left knee | $\mathcal{LHP2LKN}$ |
| | 12 | Left knee | $\mathcal{LKN}$ | $l_{10}$ | Right knee → Right ankle | $\mathcal{RKN2RAN}$ |
| | 13 | Right ankle | $\mathcal{RAN}$ | $l_{11}$ | Left knee → Left ankle | $\mathcal{LKN2LAN}$ |
| | 14 | Left ankle | $\mathcal{LAN}$ | | | |

The underlying articulated structure is modelled as a kinematic tree structure (Cormen et al., 2001) containing 9 nodes (15 nodes for the full body) and each node represents a specific body joint (Fig. 5.3(a)). Nodes are connected to each other by links (Table 5.2 and Fig. 5.3(a)). In this kinematic tree, one node is the *root*, the waist (node 0); some nodes are leaf nodes (a node with no children) such as the head (node 2) and wrists (nodes 7 and 8), which are connected to only one link; and the others such as shoulders (nodes 3 and 4) are internal nodes, which are the mid-joints between the root and the ending nodes. The internal nodes play the role of parent for some nodes, while they are children for some higher nodes. In this way, a specific joint in the tree can be reached by starting from the root and passing through some mid-joints. This sequence of nodes and links connecting a joint to the root is called a *path*. For example, the path toward node 5, the right elbow, starts from the node 0, the root, and passes through the internal nodes 1 and 3. Table 5.2 summarizes the joints and links of the employed model. Fig. 5.3(b) shows the paths in the kinematic tree with the colours corresponding to Fig. 5.3(a) .



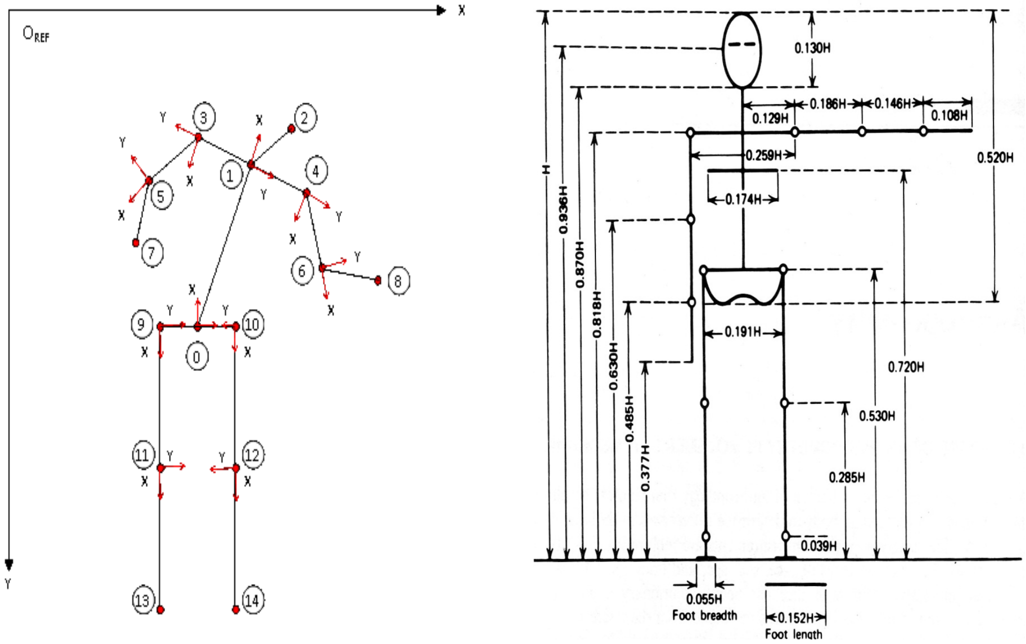
(a) The skeletal HB model.

(b) The kinematic tree and paths.

FIGURE 5.3: The skeletal HB model and its kinematic tree and paths.

#### 5.3.2.1 Pose Synthesis and Projection

For the purpose of fitting the model to the real data of the camera (the foreground interest points), we need to calculate the coordinates of the joints in the image coordinate system. This is also used to show the result of the pose estimation in the real image of the camera by projecting the model onto the image plane. Based on the tree-like structure we defined for the skeletal model, any joint is represented in its own local coordinate system with the origin at that joint. To find the coordinate of any joint in the root coordinate system, travelling through a stream of joints $s_i s_{i-1} \dots s_0$ to get to the root, it should be transferred through the systems of the intermediate joints one by one. This process is accomplished by multiplying the coordinates of that joint by the translation and rotation matrices of the intermediate joint systems (Shapiro and Stockman, 2001). For example, for each point $p$ with local coordinate $x_i$ in system $s_i$, we can find its coordinate $x_{i-1}$ in system $s_{i-1}$ using the translation and rotation matrices between the two coordinate systems. This process is repeated until we get coordinate $x_0$ for that point in the reference coordinate system. Fig. 5.4(a) shows the joints in the 2D model alongside their local coordinate systems.



(a) The implemented 2D skeletal model.

(b) The anthropometric information of the human body.

FIGURE 5.4: Skeletal HB model and its anthropometric information (Han, 2010).

This kinematic tree structure is a generic representation and should be adjusted for person-specific tracking. To do this, the model is designed based on the anthropometric information of the human body (Herman, 2007) (Fig. 5.4(b)). The limb adjustment process is performed using two different parameters $L$ and $S$, the human body height and the scale factor, respectively. The human body height $L$ allows us to define the length of each part of the body model as a fraction of $L$. Moreover, to adapt the model itself to different distances from the camera across the frames, the scale factor $S$ is used. $L$ is calculated once during the pose initialization process, whereas the scale factor $S$ is renewed for every frame during the pose estimation and tracking process.

#### 5.3.2.2 Degrees of Freedom

The other important issue about the model is the number of DoFs that each joint can have in the model as well as the total number of DoFs of the model. Generally, any rigid object can have 6 DoFs: 3 translations in the $x$, $y$, and $z$ directions; and 3 rotations roll, pitch, and yaw. Fig. 5.5, adapted from Wikipedia, shows these 6 DoFs[2] for an example rigid body. When rigid objects are connected to each other and form an articulated object, they get restricted and cannot have all of the mentioned DoFs. So, most of the joints can only have up to 3 rotational DoFs. Usually they have no DoFs for translation because of the restrictions these connections apply to each other and take their chance to be translated independently. These restrictions may also restrict some of the rotation's DoFs. Generally, the human body has around 230 joints, most of them have one DoFs and some have more than one. These joints have 244 DoFs all together (Pitman and Peterson, 1989). Owing to the application in hand, our full body model has 15 joints which have 25 DoFs in total. Fig. 5.4(a) shows the implemented 2D skeletal model with its consisting nodes. Table 5.3 summarizes the joints of the model alongside their DoFs for both the upper and lower bodies, though only the upper body part is used in this thesis.

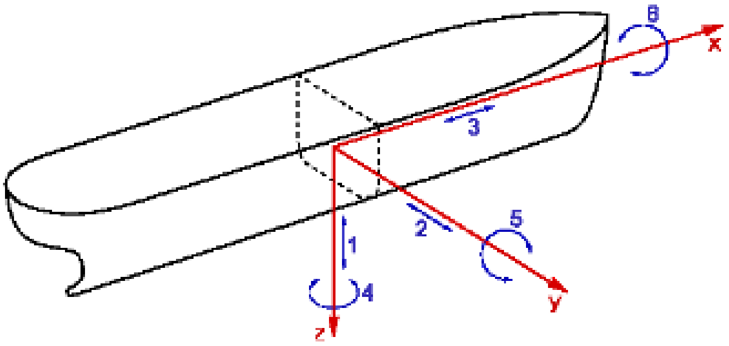


FIGURE 5.5: Six DoFs of a rigid object adapted from Wikipedia (Wikipedia).

Fig. 5.6 shows the DoFs of the joints in our employed model. Besides this, the model has 3 more DoFs for its location in the scene, which is the coordinate of the waist as the root of kinematic tree in the 3D space. So, the full body model has 25 DoFs, of which 17 DoFs belong to the upper body section. On this basis, the pose vector in 3D, which is to be estimated by the pose estimation algorithm, is defined as follows:

---

[2] Degrees of freedom (mechanics): `http://en.wikipedia.org/wiki/Degrees_of_freedom_(mechanics)`

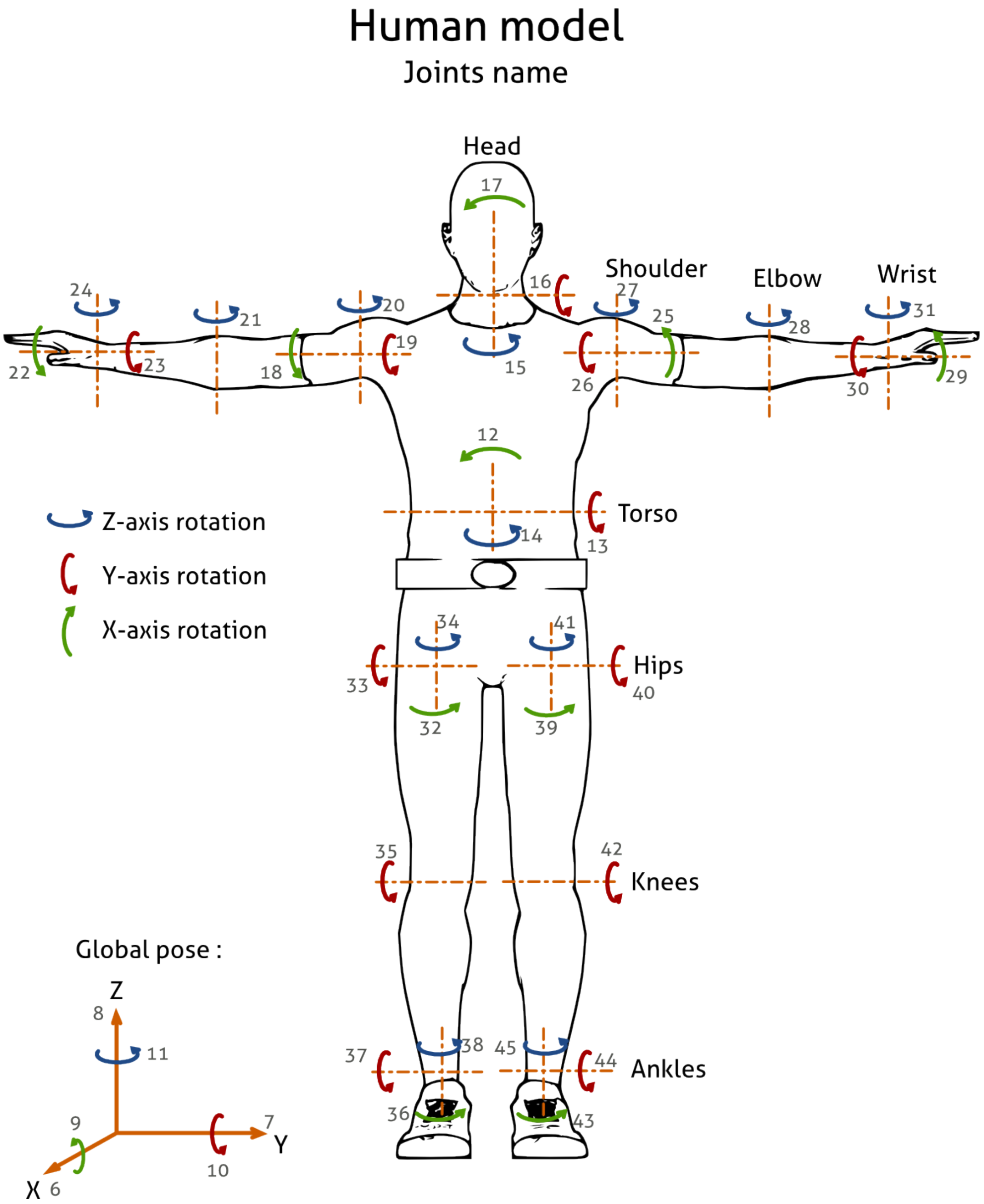


FIGURE 5.6: Human body, the joints, the DoFs of joints.

$$\mathbf{X} = [r_x, r_y, r_z, \alpha_x^{WST}, \beta_y^{WST}, \gamma_z^{WST}, \alpha_x^{NCK}, \beta_y^{NCK}, \gamma_z^{NCK}, \alpha_x^{RSH}, \beta_y^{RSH}, \gamma_z^{RSH}, \alpha_x^{LSH}, \beta_y^{LSH}, \gamma_z^{LSH}, \gamma_z^{REB}, \gamma_z^{LEB}] \tag{5.11}$$

The 2D version of the pose vector, which is used in this research, is as follows:

$$\mathbf{X} = [r_x, r_y, \theta_{WST}, \theta_{NCK}, \theta_{RSH}, \theta_{LSH}, \theta_{REB}, \theta_{LEB}] \tag{5.12}$$

TABLE 5.3: Kinematic HB model joints DoFs for both upper and lower bodies, though only the upper body part is used in this thesis.

| **Upper Body** | | | **Lower Body** | | |
|---|---|---|---|---|---|
| **ID** | **DoFs** | **#** | **ID** | **DoFs** | **#** |
| 0 | $r_x$, $r_y$, $r_z$, $\alpha_x^{WST}$, $\beta_y^{WST}$, $\gamma_z^{WST}$ | 6 | 9 | | 3 |
| 1 | $\alpha_x^{NCK}$, $\beta_y^{NCK}$, $\gamma_z^{NCK}$ | 3 | 10 | $\alpha_x^{LHP}$, $\beta_y^{LHP}$, $\gamma_z^{LHP}$ | 3 |
| 2 | - | 0 | 11 | $\alpha_x^{RKN}$ | 1 |
| 3 | $\alpha_x^{RSH}$, $\beta_y^{RSH}$, $\gamma_z^{RSH}$ | 3 | 12 | $\alpha_x^{LKN}$ | 1 |
| 4 | $\alpha_x^{LSH}$, $\beta_y^{LSH}$, $\gamma_z^{LSH}$ | 3 | 13 | - | 0 |
| 5 | $\gamma_z^{REB}$ | 1 | 14 | - | 0 |
| 6 | $\gamma_z^{LEB}$ | 1 | | | |
| 7 | - | 0 | | | |
| 8 | - | 0 | | | |
| **Total** | | **17** | **Total** | | **8** |

### 5.3.3 IP Rendering for the Pose Hypotheses

Given the $P^p$ and the IP set $\Pi^p$ from the previous frame, this section addresses the process of rendering the IP set $\Pi_k^h$ for the pose hypothesis $P_k^h$. To do this and to satisfy the previously mentioned requisites of Subsection 5.3.1, this process is conducted through the two following stages: prototyping and refinement. Algorithm 5.2 (Page 133) summarizes the whole process. Since the ”IP Rendering for the Pose Hypotheses” depends only on the IPs of the previous and current frames and also there is smoothness or small inter-frame motion between consecutive frames in human body tacking, this process is not dependent on occlusion. However, the whole process of pose estimation proposed in this thesis is valid in no-oclusion situations.

Nevertheless, using the occlusion handling procedure suggested in Chapter 6, the idea could be extended to the occlusion situations.

#### 5.3.3.1 Prototyping

This stage (line 6-11 of Algorithm 5.2 on Page 133) generates the initial prototype of $\Pi_k^h$ for the hypothesis pose $P_k^h$ according the relationship between the IP set $\Pi^p$ and pose $P^p$. Due to the articulation property of the human body, it is a true assumption that if a rigid part of the body is affected by a transformation, all the elements of that part will be transformed accordingly. On this basis, a labelling procedure is needed first to assign the $\Pi^p$ to different rigid limbs of the body (limbs in Table 5.2). This allows the algorithm to look at the IPs in rigid terms. To label each IP $\pi_i^p$, its distance to each limb is calculated using Eq 5.15. The limb with the minimum distance is considered as the assigned limb to the IP.

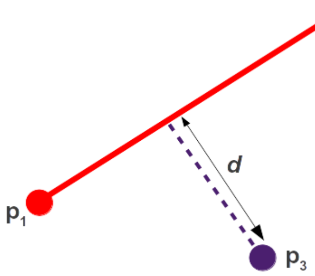


FIGURE 5.7: Shortest distance from a point to a line.

Having the line defined by two points $p_1(x_1,y_1)$ and $p_2(x_2,y_2)$, we use the method introduced by Bourke[3] to calculate the shortest distance from the point $p_3(x_3,y_3)$ to the line$p_1p_2$. The equation of the line passing through two points $p_1$ and $p_2$ is $p = p_1 + u(p_2 - p_1)$. The closest distance from point $p_3$ to the line is the length of the perpendicular line from the point to the line. Thus, the dot product of the two lines should be $(p_3 - p) \cdot (p_2 - p_1) = 0$. Substituting the equation of the line gives us $[(p_3 - p_1) - u(p_2 - p_1)] \cdot (p_2 - p_1) = 0$. Solving this gives the value of $u$ as:

---

[3] Distance from a point to a line `http://paulbourke.net/geometry/`

$$u = \frac{(x_3 - x_1)\Delta x + (y_3 - y_1)\Delta y}{\Delta x^2 + \Delta y^2} \tag{5.13}$$

where $\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$. To calculate the distance of a point to a line segment, u should lies between 0 and 1. Substituting $u$ into the equation of the line gives us the intersection point $p(x,y)$ as:

$$\begin{cases} x = x_1 + u(x_2 - x_1) \\ y = y_1 + u(y_2 - y_1) \end{cases} \tag{5.14}$$

The distance $d$, therefore between the point $P_3$ and the line is the distance between points $p(x,y)$ and $P_3$:

$$d = \sqrt{(x - x_3)^2 + (y - y_3)^2} \tag{5.15}$$

**Algorithm 5.1** IP Labelling Procedure

```
Input: IP set $\Pi^p$: $\{\pi_i^p\}_{i=1}^N$, limbs $\{l_j\}_{j=0}^7$,
and limb distance thresholds $\{\Theta_j\}_{j=0}^7$.
Output: A label $label$ for each IP $\pi_i^p$.
for each IP $\pi_i^p$ do
  $min\_distance \Leftarrow \infty$ and $label = Null$.
  for each limb $limb_j$ do
    Calculate the distance $d$ from IP to limb $limb_j$ using Eq. 5.15.
    if $d < min\_distance$ then
      $min\_distance = d$ and $label = j$.
    end if
  end for
  if $min\_distance \nleq \Theta_{label}$ then
    $label = Null$.
  end if
end for
```

Algorithm 5.1 (Page 129) outlines the IP labelling procedure. As can be seen, after finding the closest limb to an IP, that limb is considered as the label of the IP if the distance

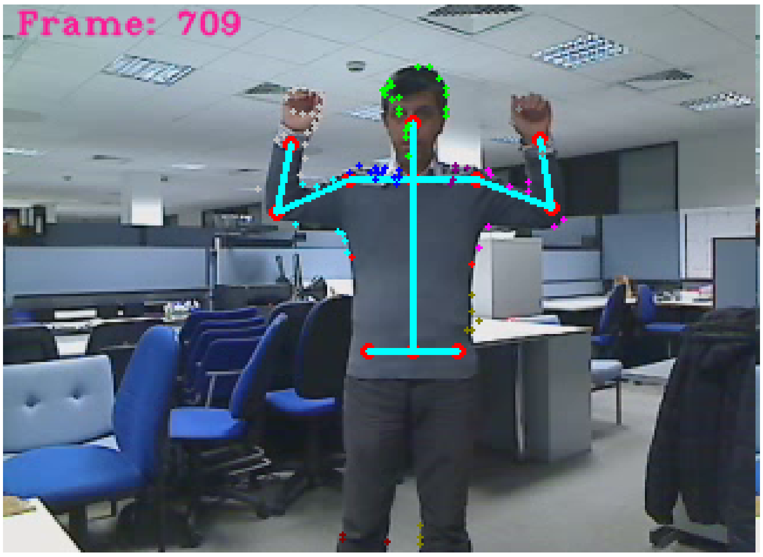


FIGURE 5.8: IP labelling with limb distance thresholds $\{\Theta_j\}_{j=0}^{7}$.

between the IP and limb satisfies a minimum specific threshold value. Otherwise the IP remains unlabelled. This distance check is performed to prevent the mislabelling of the IPs and labelling the false positive background IPs. Fig. 5.8 shows the IP labelling with limb distance thresholds $\{th_j\}_{j=0}^{7}$ on frame *709* of the test video.

To determine the limb distance thresholds, we rely on the fact that the IPs are the foreground IPs, thus they are over the human body and belong to a specific part of the body. As the pictorial-shape model of Fig. 5.9 shows, each limb skeleton passes through its corresponding pictorial part, where the size of the pictorial part is determined by anthropometric information of Fig. 5.4(b). On this basis, the distance of any IP lying inside a part to its limb skeleton should be less than a maximum value for that part. This maximum value is considered the limb distance threshold for that limb.

Having the IPs labelled to any limb of the pose $P^p$ as well as the corresponding limb of the pose hypothesis $P_k^h$, the prototyping stage renders the hypothesized IPs of that limb using the transformation between the limbs. Let us suppose we want to render IP $q_1(q_{1x}, q_{1y})$ for IP $p_1(p_{1x}, p_{1y})$ in $P^p$. If $p_1$ has been labelled as belonging to limb $l_i$ with nodes $A(A_x, A_y)$ and $B(B_x, B_y)$, and if the nodes of the corresponding limb in $P_k^h$ are $C(C_x, C_y)$ and $D(D_x, D_y)$, we

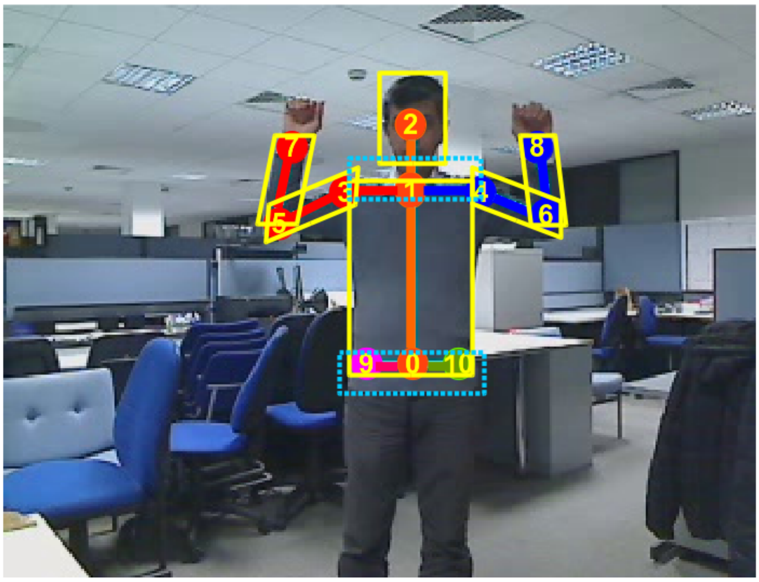


FIGURE 5.9: Limb distance thresholds $\{\Theta_j\}_{j=0}^{7}$ in pictorial model.

need the angle $\theta$ between two limbs and the scale factor $S$ between two IP sets $\Pi^p$ and $\Pi^c$ to calculate coordinates of point $q_1$ as follow:

$$\begin{cases} q_{1x} = (p_{1x} - A_x)S\cos(\theta) - (p_{1y} - A_y)S\sin(\theta) + C_x; \\ q_{1y} = (p_{1x} - A_x)S\sin(\theta) + (p_{1y} - A_y)S\cos(\theta) + C_y; \end{cases} \tag{5.16}$$

Fig. 5.10 illustrates this process for IP $p_1$, the limb $l_0$ labelled to $p_1$ and its corresponding limb in the pose hypothesis $P_k^h$ are translated first to the origin. Then, the angle $\theta$ between the translated limbs is applied to the point $p_1 - A$. The result then is translated using $C$ to create IP $q_1$. In this way, $q_1$ has the same spatial relationship with its limb that $p_1$ has with $l_0$.

#### 5.3.3.2 Refinement

In terms of the spatial relationship, the rendered IPs $\Pi_k^h$ are more similar to $\Pi^p$ than $\Pi^c$. Whereas, there is a non-linear geometric transformation between $\Pi^p$ labelled to the rigid limb $l_j$ and their matched pairs in $\Pi^c$, due to several reasons such as dealing with the 3D movements in 2D, lighting conditions, and the effect of clothing. Among these, the huge variations in the colour and texture induced by clothing, hair, and skin is one of the most important reasons

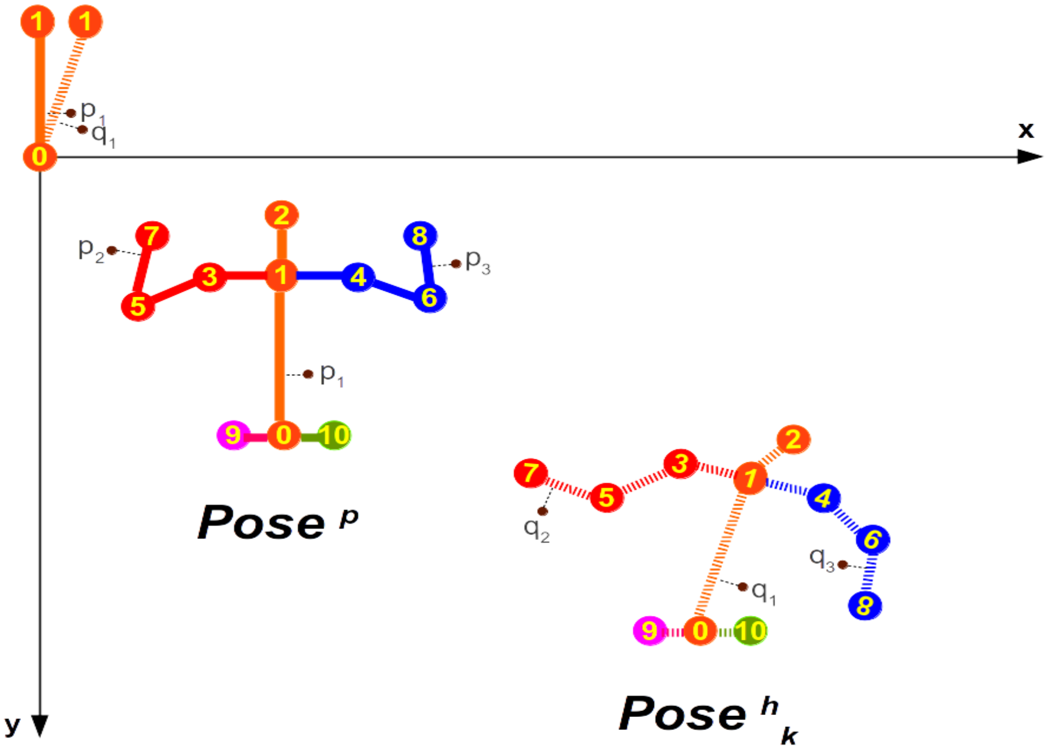


FIGURE 5.10: IP prototyping stage.

which has led to the lack of training data in pose estimation (Shotton et al., 2013). Although the depth cameras have reduced this difficulty significantly, still it is an important challenge. Thus, an appropriate transformation should be applied to the rendered IPs $\Pi_k^h$ to make it more similar to $\Pi^c$ and provide a fair comparison between the rendered IPs of the hypothesis pose and the real observation $\Pi^c$.

There are several different methods for estimating the transformation between two sets of points depending on the degree of deformation between the two sets. This fundamental problem is discussed in the context of point set registration (Brown, 1992), where it is categorized as either rigid or non-rigid. The transformation in rigid registration is handled by a small number of parameters (Brown, 1992), while non-rigid registration comprises the non-rigid transformations which are often unknown, complex, and hard to model (Chui and Rangarajan, 2003).

Since the labelling procedure of Algorithm 5.1 (Page 129) enables us to label the IPs to

**Algorithm 5.2** IP Rendering Algorithm

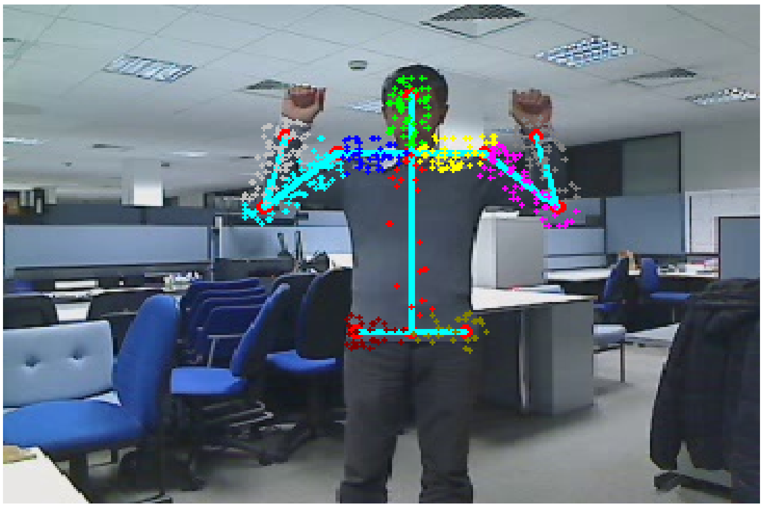
```
Input: IP sets $\Pi^p$: $\{\pi_i^p\}_{i=1}^N$ and $\Pi^c$: $\{\pi_i^c\}_{i=1}^N$, pose $P^p$,
pose hypothesis $P_k^h$, and the limb distance thresholds $\{\Theta_j\}_{j=0}^7$.
Output: The rendered IP set $\Pi_k^h$ for the pose hypothesis $P_k^h$.

IP labelling:
Extract the limbs $\{l_j\}_{j=0}^7$ for both poses $P^p$ and $P_k^h$.
Label the $\Pi^p$ using limb distance thresholds $\{\Theta_j\}_{j=0}^7$ and algorithm 5.1 (Page 129).

Prototyping:
for each $\pi_i^p$ do
    Extract the coordinates of the nodes of the labelled limb $l_j$:
    $A$, $B$ for the limb in $P^p$ and $C$, $D$ for the limb in $P_k^h$.
    Calculate the angle $\theta$ between two lines $AB$ and $CD$.
    Apply Eq. 5.16 to the coordinates of $\pi_i^p$ to get the coordinates of $\pi_{ki}^h$.
end for

Refinement:
for each limb $l_j$ do
    Compose the cluster $C_j^p$ from the $\Pi^p$ labelled to limb $l_j$.
    Compose the corresponding set $C_j^{kh}$ form the rendered IPs $\Pi_k^h$ labelled to limb $l_j$.
    Compose the corresponding set $C_j^c$ form the $\Pi^c$ using the result of matching.
    Estimate the homography matrix $H$ between two sets $C_j^p$ and $C_j^c$.
    Refine the IP set $C_j^{kh}$ by applying the estimated transformation $H$ to it.
end for
```

the limbs of the HUB and classify them into the IPs of rigid bodies, the rigid transformation estimation can be reasonably applied to our problem. Among all the most common rigid transformations (Dubrofsky, 2009), we use the perspective transformation. It is calculated basically by *Homography* estimation between two planes to handle the transformation between the two labelled IP sets. According to the definition of homography (Hartley and Zisserman, 2003), it is used in situations where IPs are coplanar. However, there are many situations in 3D computer vision such as camera calibration, 3D reconstruction, stereo vision, and scene understanding where estimating a homography may be required to solve the transformation estimation even when there are non-coplanar IPs. It doesn't matter whether the IPs of a specific limb are over a 2D or 3D model, since they belong to a rigid body, the transformation between them over any two consecutive frames can be calculated using Homography estimation.

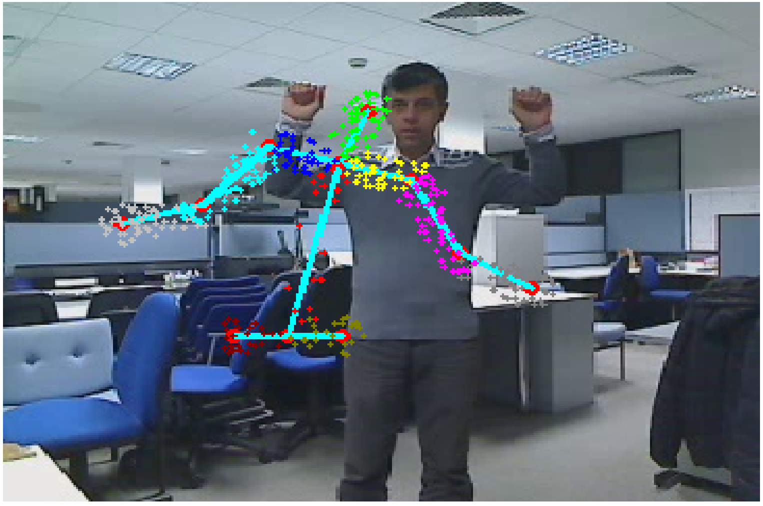
As can be seen in Appendix C, the homograhy estimation involves calculation of 8 unknown parameters in the $3 \times 3$ matrix $H$ (the Homography matrix is determined up to a scale by dividing all the entries of matrix $H$ by $h_{33}$. Thus, it is normalized so that $h_{33} = 1$ and we need to estimate only 8 parameters (Bradski, 2000)). Theoretically, 4 matched IP pairs from two IP sets are enough to estimate these parameters, but in the simplest form, where there are no outliers and the noise is rather small, all the IP pairs are used to compute an initial homography estimate using a simple least-squares scheme. In more complicated situations where not all the IP pairs fit the rigid perspective transformation, because of some outliers, the initial estimate would be poor and cannot handle the problem efficiently. In this case, robust methods such as *Random sample consensus (RANSAC)* (Fischler and Bolles, 1981b) or *least median of squares (LMeDS)* (Rousseeuw, 1984) can be used.

The RANSAC and LMeDS methods try many different random subsets of four matched pairs, estimate the homography matrix for the subset using a simple least square algorithm, and then compute the quality/goodness of the computed homography based on the number of inliers or the median re-projection error for RANSAC or LMeDs respectively. Finally, the best subset is used to produce the homography estimate and the mask of inliers/outliers. The computed homography matrix is then refined using the inliers with the Levenberg-Marquardt method (Moré, 1978) to decrease the back-projection error (Appendix C) as much as possible. RANSAC handles practically any ratio of outliers but it needs a threshold to discriminate inliers from outliers. As Torr and Zisserman (2000) mentioned, RANSAC can be sensitive to the threshold value that defines how well data fit a model with a given set of parameters. If the threshold value is too large, then all the parameter hypotheses tend to be ranked equally well. On the other hand, when it is too small, the estimated parameters tend to fluctuate when a datum is added or removed from the set of inliers. In comparison, LMeDS does not need any threshold but it works correctly only when there are less than 50% of outliers.

On this basis, the prototyped IP set $\Pi_k^h$ is refined using the estimated homography matrix $H$. Line 12-19 of Algorithm 5.2 (Page 133) summarizes this process. Fig. 5.11 shows the rendered IPs $\Pi_k^h$ for the pose hypothesis $P_k^h$. In fact, the IP rendering process translates any pose hypothesis into the IPs which can be compared with the foreground IPs of the current frame.



(a) IPs $\Pi^p$ for the pose $P^p$.



(b) The rendered IPs $\Pi_k^h$ for the pose hypothesis $P_k^h$.

FIGURE 5.11: The process of rendering IPs $\Pi_k^h$ for the pose hypothesis $P_k^h$.

### 5.3.4 Scaling Factor Estimation

As mentioned earlier in Section 5.3.2.1, the limbs of the kinematic skeletal model should be adjusted at each frame to adapt the model itself to different distances from the camera. Although different parts of the body impose constraints on each other and therefore they cannot move completely independently, they can change their distances from the camera independently. When these 3D movements are dealt with in 2D (which is the scope of this thesis), it leads to different scaling factor for different parts of the body. To handle this situation, we introduce a limb scaling factor estimation method, inspired by the work proposed by Artner et al. (2011), to calculate the scale factor $S_j$ for limbs. If the movement is in 2D, we can estimate the whole body scaling factor by averaging the estimated limbs scaling factor.

#### 5.3.4.1 Scaling of the Individual Limbs

The IP labelling process of Algorithm 5.1 (Page 129) gives us the matched IPs of individual limbs in separate clusters. This makes it possible to calculate the limb scale factor $S_j$ for the limb $l_j$ as follows. Suppose we have the corresponding clusters $C_j^p$ and $C_j^c$ of $\Pi^p$ and $\Pi^c$ respectively, labelled to limb $l_j$. The IPs of each of these clusters form in fact a fully connected graph with the IPs as the nodes and the links as the connecting edges between the nodes, denoted by $v$ and $e$ respectively. Thus, calculating the limb's scale factor $S_j$ is equivalent to the calculation of the scale factor between two graphs as follow:

$$S_j(v) = \sum_{e \in \mathbf{E}(v)} \frac{|e^c|}{|e^p|} \frac{\alpha(v_e)}{\sum_{v_e \in \mathbf{N}(v)} \alpha(v_e)} \tag{5.17}$$

where $S_j(v)$ is the estimated scale factor in the local neighbourhood $\mathbf{N}(v)$ of node $v$. $\mathbf{N}(v)$ is where all the nodes $v_e$ are connected to node $v$ by link $e$ and $\mathbf{E}(v)$ are all edges $e$ of the graph incident to node $v$. The constant weight coefficient $\alpha(v_e)$ is used to boost the influence of the most confident nodes and their associated edges. The calculated scale factor $S_j(v)$ of each each node $v$ in limb cluster $j$ is used to calculate the limb scale factor as below:

$$S_j = \sum_{v \in V_0} S_j(v) \frac{\alpha(v)}{\sum_{v \in V_0} \alpha(v)} \tag{5.18}$$

### 5.3.5 Cost (Evaluation) Function

The cost function for PSO measures how well a pose hypothesis is compatible with the visual observations (the foreground IPs of the current frame). As described earlier, a set of IPs is generated for the pose hypothesis by means of rendering. Owing to this assumption that the hypothesized pose matches closely the pose of the person in the current frame if its generated IPs match closely the foreground IPs of the frame, the discrepancy function should measure the similarity between these two IP sets in terms of the Euclidean distance as well as the local neighbourhood descriptor similarities.

The cost function $D(P_k^h, \Pi^c)$ is proposed in our work to measure the discrepancy between the pose hypothesis $P_k^h$ and the observation $\Pi^c$. This function calculates the distance between the hypothesized pose $P_k^h$ and the current IPs $\Pi^c$ as follows:

$$D(P_k^h, \Pi^c) = (1-\beta)\sum_{i=1}^{N} \frac{\alpha(\pi_i)}{N} \|\pi_i^h - \pi_i^c\| + \beta \sum_{i=1}^{N} \frac{1}{N \times 64} D_H(C(\pi_i^h), C(\pi_i^c)) \tag{5.19}$$

The first term in function $D$ calculates the Euclidean distance between $\Pi^h$ and $\Pi^c$, where $\|\pi_i^h - \pi_i^c\|$ shows the Euclidean distance between $\pi_i^h$ and $\pi_i^c$. The constant weight coefficient $\alpha(\pi_i)$, which is the matching confidence between $\pi_i^p$ and $\pi_i^c$, is used to determine the share of each IP in creating the distance measure. $\alpha_i$ is a real number $\in [0,1]$ so that the higher value implies the higher reliability in the matching.

The second term measures the distance between the local descriptors of $\Pi^h$ and $\Pi^c$. The Census Transform (CT), which is a non-parametric local transform (Zabih and Woodfill, 1996), is used as the local feature descriptor in this regard, though the other feature descriptors such as SURF (Bay et al., 2008) can be used too. It is defined as an ordered set of comparisons between the intensity of an objective pixel $(x_i, y_i)$ and the intensity of its neighbouring pixels $(x_j, y_j)$ in a local neighbourhood $\mathcal{N}(x_j, y_j)$, where $(x_j, y_j) \neq (x_i, y_i)$. Generally, a $3 \times 3$ window is considered as the local neighbourhood. We, instead, consider the larger size $16 \times 16$, while ignoring some pixels in every second row and column. In this fashion, we use a wider area around the central pixel with smaller number of neighbouring pixels, 8 × 8.

The *CT* then generates a binary bit string of 64 bits representing which pixel in $\mathcal{N}(x_j, y_j)$ has intensity lower than $I(x_i, y_i)$, the intensity of the central pixel. Having the comparison operator $\zeta(I(x_i,y_i), I(x_j,y_j))$ and the concatenation operation $\bigotimes$, the CT at a $\pi_i$ with coordinate $(x_i, y_i)$ is defined as:

$$\mathcal{C}(\pi_i) = \mathcal{C}(x_i,y_i) = \bigotimes_{(x_j,y_j)\in\mathcal{N}} \zeta(I(x_i,y_i), I(x_j,y_j)) \tag{5.20}$$

where the comparison operator is defined as:

$$\zeta(I(x_i,y_i), I(x_j,y_j)) = \begin{cases} 1 & if \quad I(x_i,y_i) < I(x_j,y_j) \\ 0 & \quad otherwise \end{cases} \tag{5.21}$$

To measures the distance between the CT descriptors of $\pi_i^p$ and $\pi_i^c$, the second term of Eq. 5.19, the Hamming distance $D_H(\mathcal{C}(\pi_i^h), \mathcal{C}(\pi_i^c))$ is used. This widely used distance measure, particularly in information theory, counts the number of positions of two strings of equal length at which the corresponding symbols are different (Hamming, 1950) (Fig. 5.12).

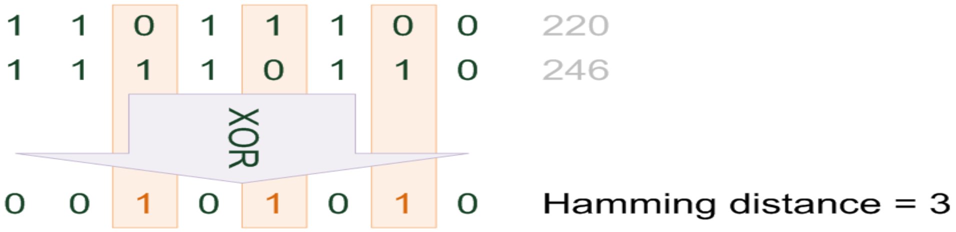


FIGURE 5.12: The Hamming distance between two strings of equal length.

The normalization factor $\beta$ in Eq. 5.19, $0 \leq \beta \leq 1$, is used to regulated the effect of the spatial and local distance between $\Pi^h$ and $\Pi^c$.

## 5.4 Implementation and Experimental Results

In this section we present the details of our two-stage strategy in implementation and using PSO for the model-based human upper body tracking. Firstly the HPSO approach, which estimates the pose hierarchically is discussed. Then, GPSO, the post-processing stage for consistency check and pose refinement, is presented. 30olorredFinally, the experimental results and performance evaluation is discussed.

### 5.4.1 Hierarchical PSO-Based HUB Pose Estimation (HPSO) Algorithm

As stated before in Section 5.3.2, the pose vector of HUB is represented by a vector of 8 parameters. Because each particle represents a potential solution in the search space, we are facing a 8-dimensional search space of all plausible skeleton configurations. Solving the optimization problem in such a high dimensional search space is a challenge for any optimisation method, including PSO. The high dimensionality:

- implies optimising a multi-modal function, which requires a complicated cost function and a larger swarm size.
- increases prohibitively the computational expense of calculating the cost function for the particles.
- involves the observation noise of all the dimensions at the same time in the optimisation process.

Taking advantage of the hierarchical nature of our tree-like kinematic model, we propose a hierarchical PSO approach to tackle the above mentioned difficulties. It solves the optimisation problem in the 8-dimensional pose space through a *divide-and-conquer* strategy by decomposing the parameters of the pose vector and estimating them separately. Owing to the constraints that the higher joints in the tree apply to the lower ones, we optimize the corresponding parameters of the joints in the pose vector hierarchically in several levels, from high to low according to the kinematic tree of Fig. 5.3. Thereby, it reduces the complexity of the search and overcomes the aforementioned drawbacks. Depending on the articulation of each joint and the connection between them, the optimisation process can be performed at each level, either on one joint only or a group of joints together. Fig. 5.13 shows the hierarchical PSO-based HUB pose estimation algorithm block diagram.

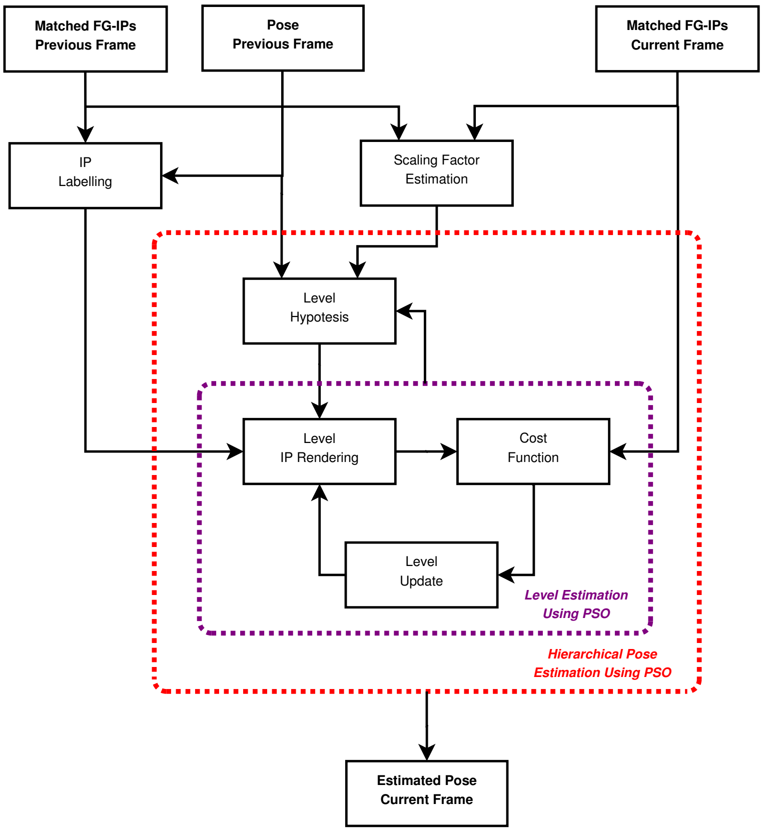

Matched FG-IPs Previous Frame
Pose Previous Frame
Matched FG-IPs Current Frame
IP Labelling
Scaling Factor Estimation
Level Hypotesis
Level IP Rendering
Cost Function
Level Update
Level Estimation Using PSO
Hierarchical Pose Estimation Using PSO
Estimated Pose Current Frame

FIGURE 5.13: The hierarchical PSO-based HUB pose estimation algorithm block diagram.

#### 5.4.1.1 Hierarchy in Estimation

According to the kinematic tree structure of Fig. 5.3, we perform the hierarchical optimisation process in *7* levels (Table 5.4). The hierarchy is started at level *0* by optimizing the parameters $r_x, r_y$ of the pose vector (2 DoFs of the model), which determine the global position of the skeletal model in the space. It is the most significant level as it is the root of the kinematic tree and the other joint parameters are greatly dependent on it. A slight error in the estimation of the global position often results in a drastic deviation of the child joints in the tree from their real values.

TABLE 5.4: The 7 hierarchical steps of our HPSO upper body pose optimisation.

| **Level** | **Parameter** | **Estimated joints** | **Involved limbs** |
|---|---|---|---|
| 0 | $r_x, r_y$ | $\mathcal{WST}$, $\mathcal{RHP}$, $\mathcal{LEP}$ | right hip, left hip |
| 1 | $\theta_{WST}$ | $\mathcal{CSH}$, $\mathcal{RSH}$, $\mathcal{LSH}$ | torso, right clavicle, left clavicle |
| 2 | $\theta_{NCK}$ | $\mathcal{HED}$ | head |
| 3 | $\theta_{RSH}$ | $\mathcal{REB}$ | right upper arm |
| 4 | $\theta_{LSH}$ | $\mathcal{LEB}$ | left upper arm |
| 5 | $\theta_{REB}$ | $\mathcal{RWR}$ | right lower arm |
| 6 | $\theta_{LEB}$ | $\mathcal{LWR}$ | left lower arm |

In each level, HPSO hypothesizes several particles for the pose vector at that level around its potential position. Having calculated the scaling factor for the limbs involved in the level, the scaling factor is applied to the limb in the previous frame to update the limb's length in the current frame. The updated length of the limb is then used to calculate the coordinates of the joints belonging to the level for each particle $k$. Then, the IPs $\Pi_k^h$ are rendered for the particle using Algorithm 5.2 (Page 133). The cost function Eq. 5.19 is then utilized to evaluate the hypothesis. Repeating this process for all the particles over several iterations will create the particle which has the lowest cost function and thus the highest similarity to the observations of the level in the current frame. Algorithm 5.3 (Page 142) outlines the hierarchical pose estimation approach.

After estimating the pose vector of a specific level (a $1 \times 2$ vector for level *0* and a $1 \times 1$ vector from level *1* onwards), the coordinates of the child joints, connected to the estimated parent joint, are calculated using the model parts length information of Fig. 5.4(b). For example, in estimating $\theta_{WST}$ in level *1*, let us calculate the position of the centre of shoulder, $\mathcal{CSH}$, the right shoulder, $\mathcal{RSH}$, and the left shoulder, $\mathcal{LSH}$. The third column of Table 5.4 shows the child joints that are calculated after estimation of each level parameter. Fig. 5.14 shows the levels in the kinematic tree.

**Algorithm 5.3** Hierarchical PSO-Based HUB Pose Estimation Algorithm

```
Input: IP sets $\Pi^p$: $\{\pi_i^p\}_{i=1}^N$ and $\Pi^c$: $\{\pi_i^c\}_{i=1}^N$, pose $P^p$,
and the limb distance thresholds $\{\Theta_j\}_{j=0}^7$.
Output: The estimated pose $P^c$.
IP labelling:
Extract the limbs $\{l_j\}_{j=0}^7$ for pose $P^p$.
Label the IPs $\Pi^p$ using limb distance thresholds $\{\Theta_j\}_{j=0}^7$ and IP-labelling Algorithm 5.1
(Page 129).
Scaling factor estimation:
for each limb $\{l_j\}_{j=0}^7$ do
    Calculate the limb scale factor $S_j$ using the labelled IP clusters $C_j^p$ and $C_j^c$ and Eq. 5.18.
end for
Calculate the whole upper body scale factor.
Hierarchical pose estimation:
for each level $i$ do
    Apply the limb's scaling factor to the limbs involved in the level
    (According the table 5.2) to update the limbs length for current frame.
    Form level pose vector: $1 \times 2$ vector for level 0 and $1 \times 1$ vector from level 1 onwards.
    Set the PSO parameters for level $i$.
    Generate $nPops$ random pose vector particles for level $i$ using the PSO parameters.
    Set the iteration counter to 1 ($It = 1$).
    while ($It < MaxIt$) do
        for each particle $k$ do
            if ($It == 1$) then
                Calculate coordinates of joints at level $i$ using the particle's pose vector and
                updated limb lengths.
                Render IPs $\Pi_k^h$ for particle $k$ using the IP-rendering Algorithm 5.2 (Page 133).
                Calculate the cost value of particle $k$ using Eq. 5.19.
                if (cost of particle $k$ < Best cost) then
                    Best cost = cost of particle $k$ and Best pose vector = pose vector of particle $k$.
                end if
            else
                Update the particle's pose vector using the PSO formulations of Section 5.2.
                Calculate coordinates of joints at level $i$ using the particle's pose vector and
                updated limb lengths.
                Render IPs $\Pi_k^h$ for particle $k$ using the IP-rendering Algorithm 5.2 (Page 133).
                Calculate the cost value of particle $k$ using Eq. 5.19.
                if (cost of particle $k$ ¡ Best cost) then
                    Best cost = cost of particle $k$ and Best pose vector = pose vector of particle $k$.
                end if
            end if
        end for
        $It = It + 1$.
    end while
    Calculate coordinates of joints at level $i$ using the Best pose vector.
end for
```

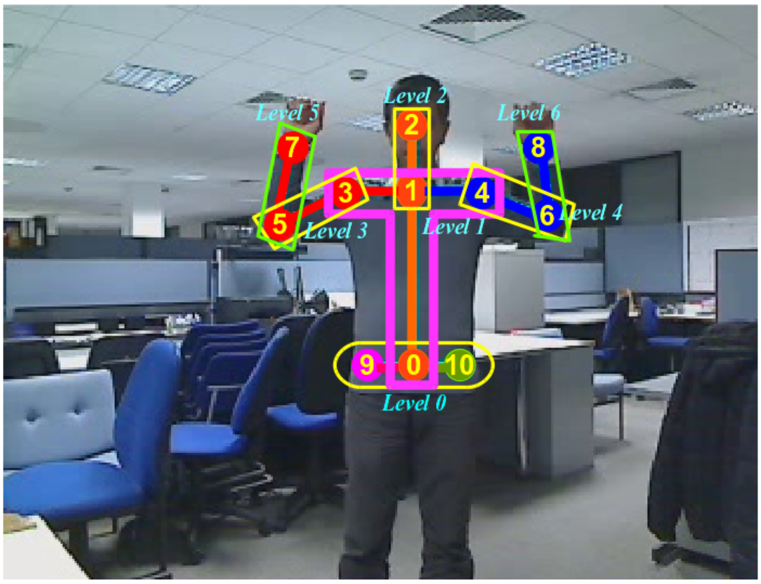


FIGURE 5.14: Levels in the kinematic tree.

As Fig. 5.14 shows, the proposed hierarchical human body pose estimation using the proposed algorithm is performed as follows:

- ***Level 0***: the goal of this level is estimation of the joint $\mathcal{WST}$, the root of kinematic tree, with coordinates $[r_x, r_y]$. Although the right and left hips belong to the lower body, they are taken into account in estimation of the root to provide more information. After running the PSO, the particle with the best cost gives the estimated joint $\mathcal{WST}$. Straightaway, the coordinates of the joints $\mathcal{RHP}$ and $\mathcal{LHP}$ are calculated using the updated limbs and the estimated root.

- ***Level 1***: Having the estimated joint $\mathcal{WST}$, this level attempts to estimate $\theta_{WST}$ as the pose vector. Estimating $\theta_{WST}$ then yields the coordinates of joints $\mathcal{CSH}$, $\mathcal{RSH}$, and $\mathcal{LSH}$. In fact, for any hypothesized particle $\theta_{WST}$, algorithm calculates coordinates of the mentioned joints using the updated limb lengths at this level.

- ***Levels 2 to 6***: in all these levels the pose vector is a $1 \times 1$ vector, which is one of the angles in the 2D upper body pose vector of Eq. 5.12 from $\theta_{NCK}$ to $\theta_{LWR}$. To estimate this angle, several hypotheses are generated for the angle. Having the estimated parent joint of the level as well as the updated length of its involved limbs, the coordinates of the child output joint is calculated and used to render the hypothesised IPs for this level. In this way, each particle in the level has the rendered IP sets which is used for the cost calculation. After several iterations, PSO estimates the best child joint as the output of the level which are used as the parent joint for the next level.

#### 5.4.1.2 HPSO Upper Body Pose Estimation Results

In this section we present the results of our proposed hierarchical PSO-based human upper body pose estimation algorithm. As mentioned in Chapter 1, the pose initialization process has not been considered in this work. Instead, we suppose that the initial pose is available as in Eq. 5.22 and we focus on estimating the pose of the subsequent frames using the IPs through a PSO-based framework. Fig. 5.15 shows the initial pose that we use in our experiments, which is frame number *709* of the test video that we recorded for the experiments.

$$\mathbf{X} = [175, 150, 0, 0, 65, 70, 125, 120] \tag{5.22}$$

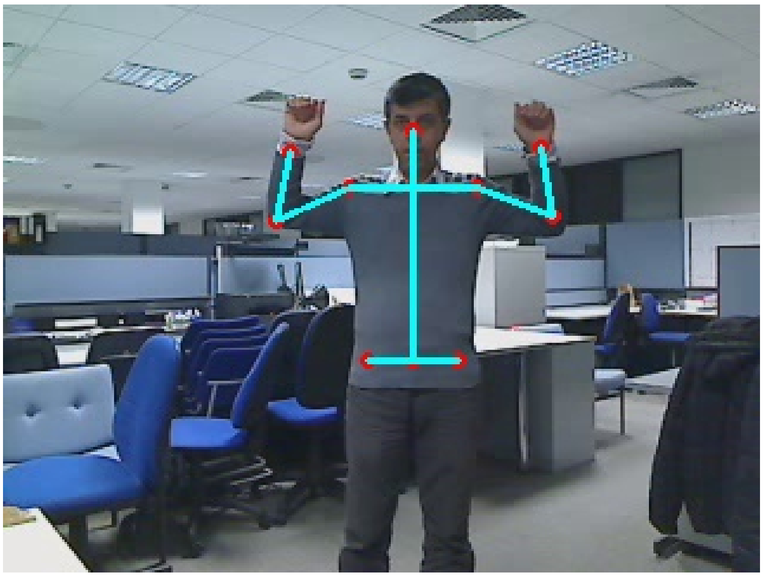
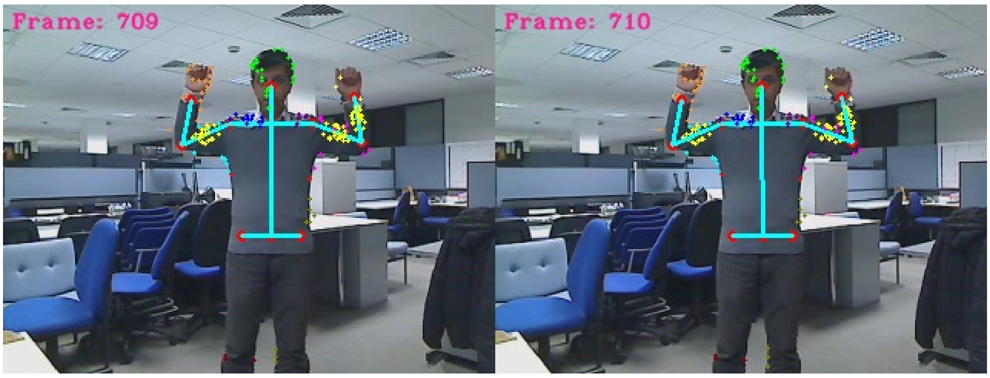
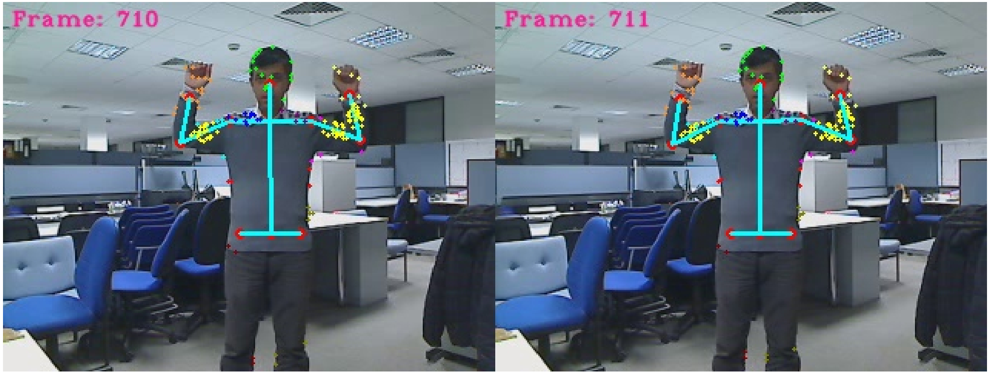


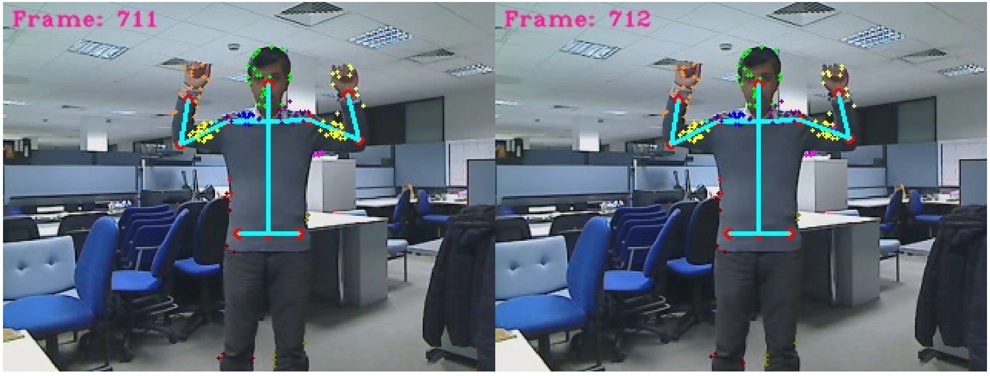
FIGURE 5.15: The initial pose $\mathbf{X} = [175, 150, 0, 0, 65, 70, 125, 120]$, which we use in our experiments.



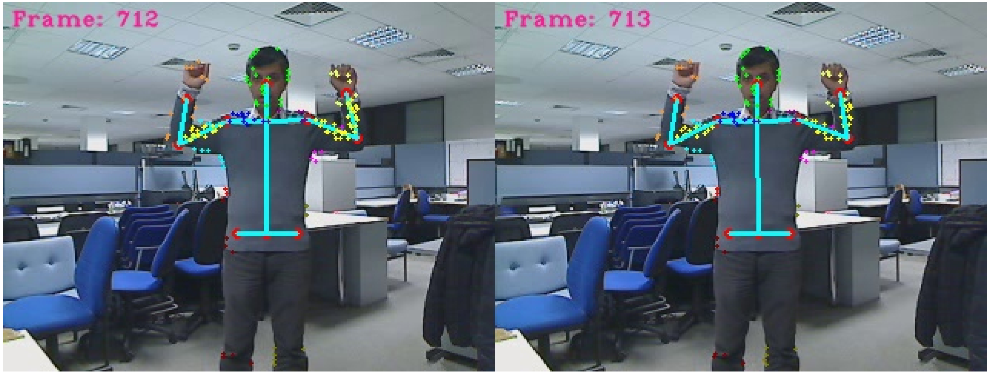
(a) $1^{st}$ *round* of estimation, frame 710.



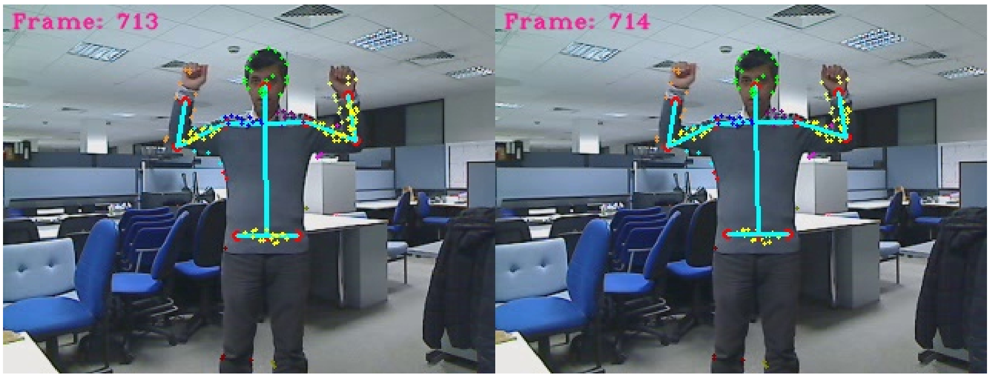
(b) $2^{nd}$ *round* of estimation, frame 711.



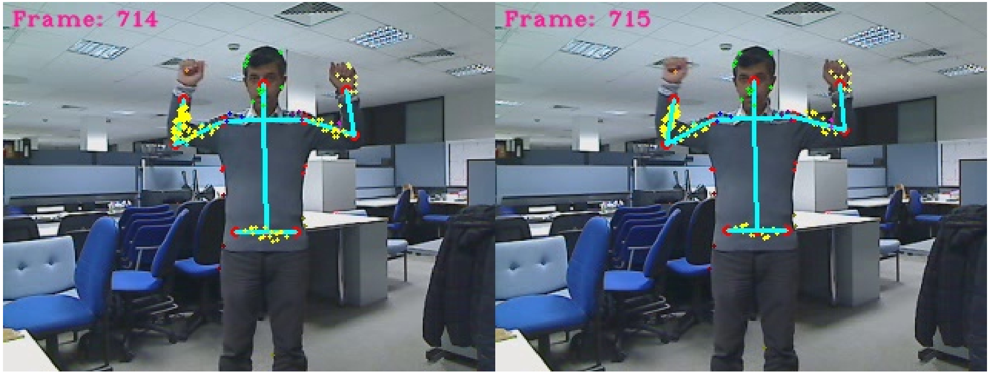
(c) $3^{rd}$ *round* of estimation, frame 712.



(d) $4^{th}$ *round* of estimation, frame 713.



(e) $5^{th}$ *round* of estimation, frame 714.



(f) $6^{th}$ *round* of estimation, frame 715.

FIGURE 5.16: 6 rounds of pose estimation from frame 710 to 715.

Given this initial pose as well as the matched foreground IP pairs between the frame of the initial pose and its subsequent frame (which is called the current frame), the proposed hierarchical PSO-based pose estimation approach performs the first round of the pose estimation and estimates the pose in the current frame using Algorithm 5.3 (Page 142). This process is repeated for any two consecutive frames. Fig. 5.16 shows both the frames and the results of the pose estimation over 6 consecutive rounds of pose estimation. In each round, the left image shows the previous frame along with its labelled foreground IPs and its pose. Similarly, the right image shows the same information for the current frame except for the pose which is the estimated pose in this case. The estimated pose in each round appears as the previous pose for the next round of pose estimation.

Focusing on the limb involved in level *5* (the right forearm) in Fig. 5.16(f), it can be seen that the limb of this level has only one foreground matched IP (the IP with colour orange.

Please ignore the yellow IPs for the moment), which presents the estimation in this level with a great deal of ambiguity because there is not enough information available for the cost function to find the particle with the lowest cost. So, the pose estimation process in this situation would be erroneous. To tackle situations like this and provide more information, a set of *virtual matched IP pairs* is generated for the limb (limbs) involved in the level. This process is illustrated in the following subsection. All the bold yellow IPs in Figs. 5.16, 5.18, and 5.19 are virtual IPs generated using the proposed method.

#### 5.4.1.3 Virtual Matched IP Pair Rendering

The procedure to render the virtual matched IP pairs for the limbs with a small number of foreground matched IP pairs is similar to the IP-rendering approach of Algorithm 5.2 (Page 133) with some little differences as follows:

- The IP-rendering algorithm is used to generate the hypothesized IPs for the hypothesized pose, while the virtual IP rendering procedure is utilized to generate virtual matched IP sets for the corresponding limbs of a level in the previous and the current frames. This is the main difference between these algorithms.
- In the IP rendering process, we have the IPs of the first set. So, we render the corresponding IPs for the second set based on that, whereas here both sets should be generated. Thus, the first step here is to generate some random IPs around the limb for the first set. To do this, we find the point $p$ on a line perpendicular to line $SE$ (the limb) at its midpoint. Having the coordinates of the vertices of triangle $SpE$, we generate a random set of IPs inside the triangle. This process will give us the first set of IPs. Fig. 5.17 shows a typical output of this process. In this figure we have virtual IPs on both sides of the limb $SE$, while only one side could be enough and the IPs of the other side could be omitted.
- In the IP rendering process, we have the coordinates of both the limbs in the previous pose and hypothesized pose. However, here we don't have the pose and thus the limb

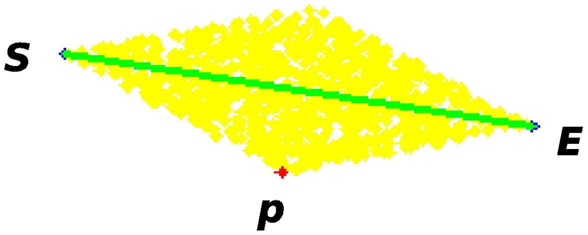


FIGURE 5.17: Generating the random virtual IPs for a limb.

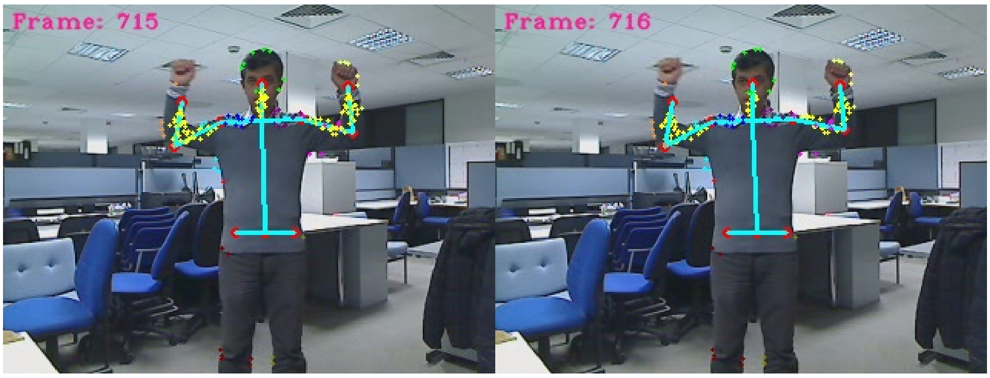
(limbs) of the level. Now, the question is how to render the virtual IPs for the second set when there is no corresponding baseline (limb) for both sets. To overcome this problem we use 2 matched IP pairs of the limbs. These IPs form the baselines for limb in the previous and current frame. Now, the IP-rendering process of Algorithm 5.2 (Page 133) can be used to render the IPs of the limb in the current frame. Algorithm 5.4 (Page 148) outlines this approach.

**Algorithm 5.4** Virtual-Matched-IP-Pair-Rendering Algorithm

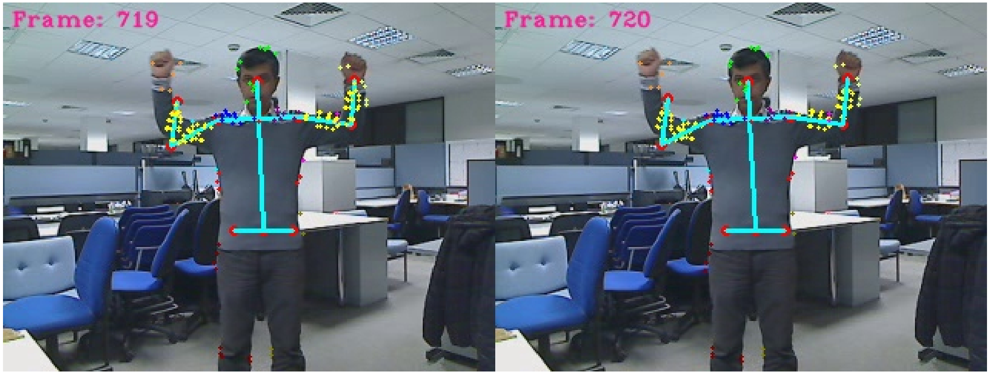
```
Input: The labelled IP clusters $C_j^p$ and $C_j^c$.
Output: The virtual foreground IPs generated for limbs $l_j^p$ and $l_j^c$.

Generating random virtual IPs for limb $l_j^p$:
Find the point $p$ on a line perpendicular to line $SE$ (the limb) at its midpoint.
Generate a random set of IPs inside the triangle $SpE$ using the coordinates of the vertices of triangle.

Rendering the virtual IPs for limb $l_j^c$:
Create the baseline for the clusters $C_j^p$ and $C_j^c$ using 2 matched IP pairs of the limbs.
Render virtual IPs for limb $l_j^c$ having the baselines and the virtual IPs of limb $l_j^p$ using IP-rendering process of Algorithm 5.2 (Page 133).
```

#### 5.4.1.4 Error Propagation in Pose Estimation

Regardless of the mentioned advantages and benefits of hierarchical pose estimation approach, it has the disadvantage of propagating the estimation error down the hierarchy. Since the estimated joint (joints) of a specific level acts (act) as the parent joint (joints) for the subsequent levels connected to the level, the inaccurate estimation of the joint (joints) of this level influences (influence) the estimation of the connected child joints in the subsequent levels. As experimental results show in Fig. 5.18, this problem happens mostly in levels *3* and *5*, which causes the erroneous estimation of joints $\mathcal{REB}$, $\mathcal{LEB}$. These inappropriate estimated joints affect the estimation of joints $\mathcal{RWR}$ and $\mathcal{LWR}$ in levels *4* and *6* accordingly. We call this type of error from now on as the *"spatial propagation error"* in comparison with the *"temporal propagation error"* which will be described in the next paragraph.



(a) Frame 716.



(b) Frame 720.

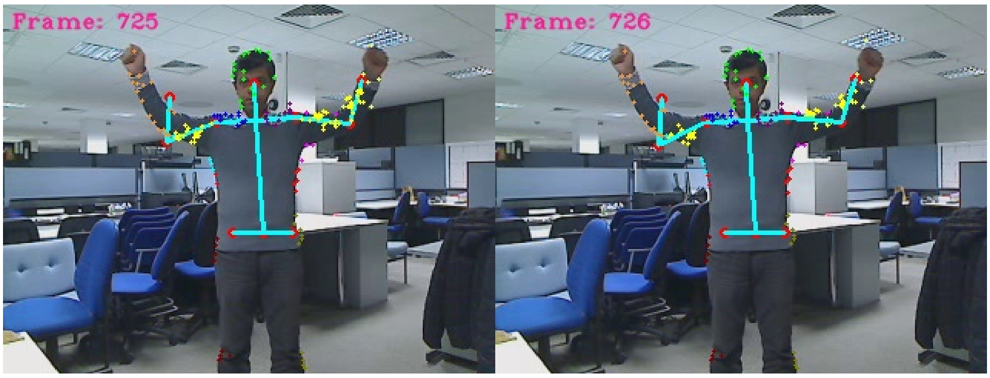
FIGURE 5.18: Spatial and temporal propagation errors over *5* rounds of pose estimation from frame 716 to 720.

Beyond the inaccurate estimation of the joints in the current frame, which is unpleasant in itself, the spatial propagation error affects the successive rounds of estimation because the estimated pose in each round appears as the previous pose for the next round of pose estimation. This causes the temporal propagation error which causes the error to grow. Fig. 5.18 shows two rounds of pose estimation where the spatial and temporal error propagation happens. As can be seen in Fig. 5.16(f), the erroneous pose estimation in level $5$ results in the inaccurate estimation for joint $\mathcal{RWR}$. This spatial error doesn't affect any subsequent level because this joint is a leaf node in the kinematic tree. However, it affects the estimation of the joint itself in the consequent frames and generates a temporal propagation error for the joint (As can be seen in Fig. 5.18).

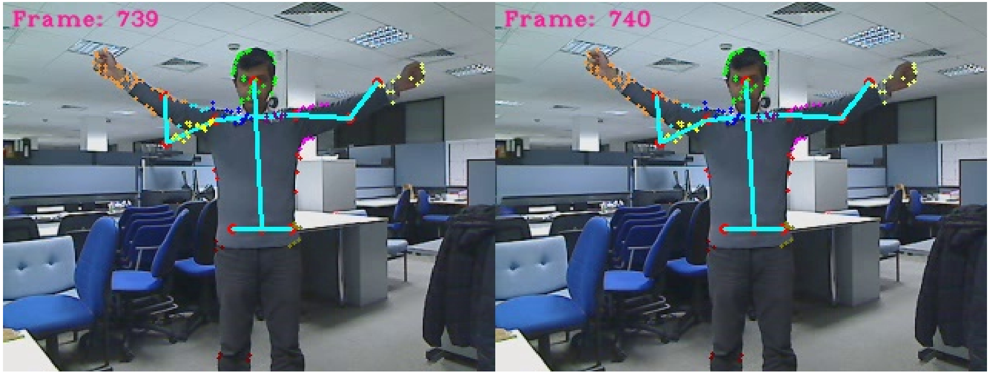
Although the virtual-matched-IP-pair-rendering process helps the HPSO algorithm to estimate the levels, it is not the optimal solution and as you can see in Figs. 5.18 and 5.19, the pose estimation algorithm faces the spatial and temporal propagation errors even when virtual-matched-IP-pair-rendering is used. To solve this problem we need a post-processing stage to perform a consistency check and pose refinement on the pose estimated by the hierarchical pose estimation algorithm. This process is described in the next section.

### 5.4.2 Global-PSO for Consistency Check and Pose Refinement

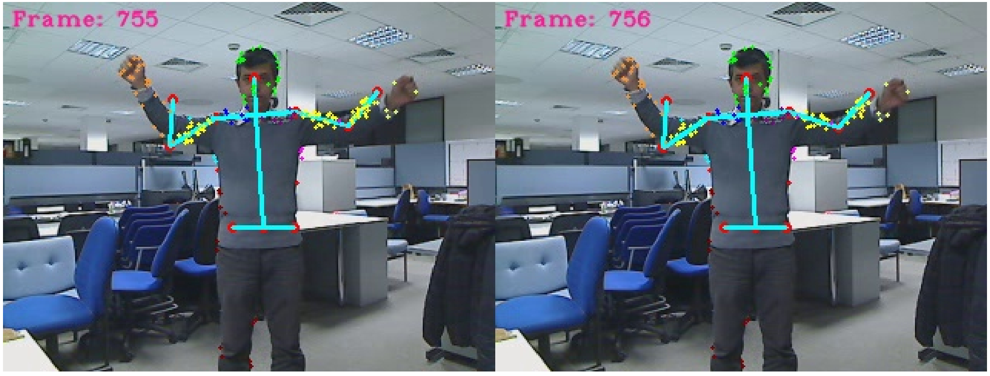
To overcome the spatial and temporal error propagation problems, we apply another PSO-based pose estimator to the pose estimated by the hierarchical pose estimation method. This estimator, which we call Global-PSO (GPSO), is different from HPSO in the sense that it searches the pose space at once for all the parameters of the pose vector, unlike the HPSO which breaks the pose space into lower dimensional spaces and estimates the pose parameters hierarchically. Since, the estimated pose is approximately correct for most of the parameters of the pose vector, as can be seen in the figures of Section 5.4.1, the mission of GPSO (as the post-processor entity in our system) is to search around the pose estimated by HPSO to correct the erroneous parameters. The reasons why GPSO could be beneficial for our problem, in comparison with when GPSO is used by itself without HPSO are:



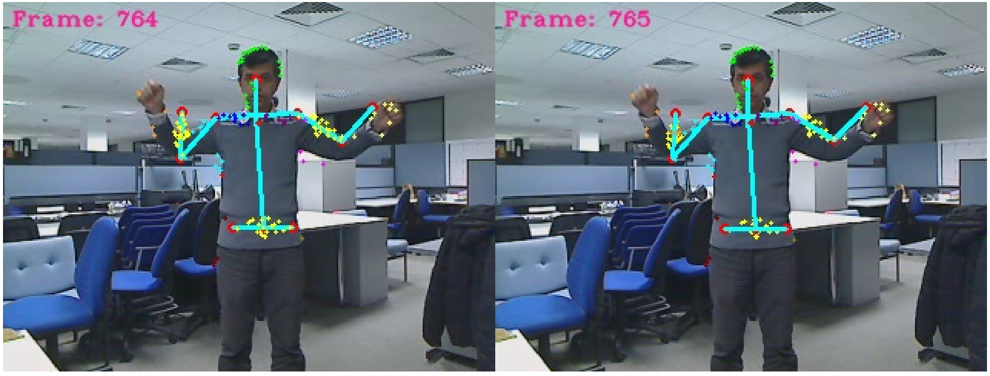
(a) Frame 726.



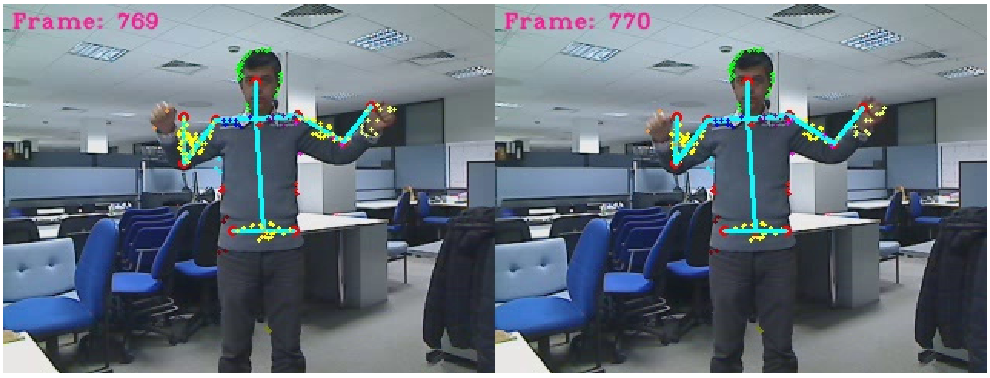
(b) Frame 740.



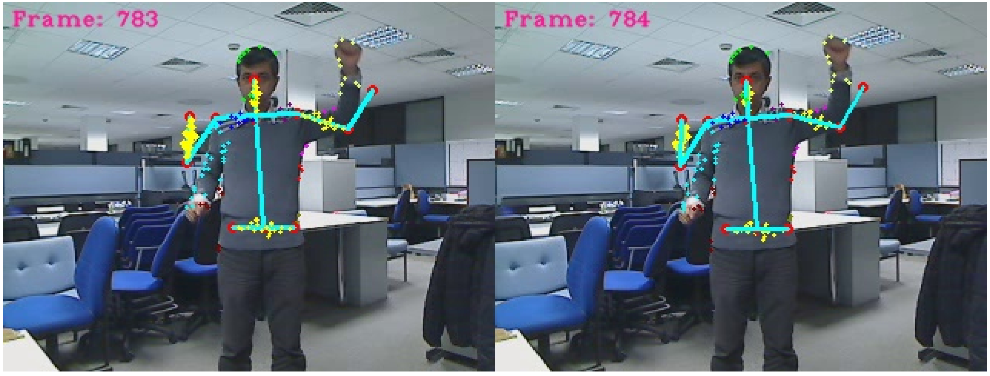
(c) Frame 756.



(d) Frame 765.



(e) Frame 770.



(f) Frame 784.

FIGURE 5.19: 6 non-consecutive rounds of pose estimation from frame 726 to 784.

- We have the estimated pose from HPSO. Thus, GPSO needs to explore a narrower region in the 8D pose search space around the estimated pose. It implies that GPSO could be handled with fewer particles and fewer iterations, In contrast, GPSO by itself would need a bigger population of particles to search a wider space in the high-dimensional pose search space. This would lead to a higher computational cost and the risk of getting stuck somewhere in the search space.

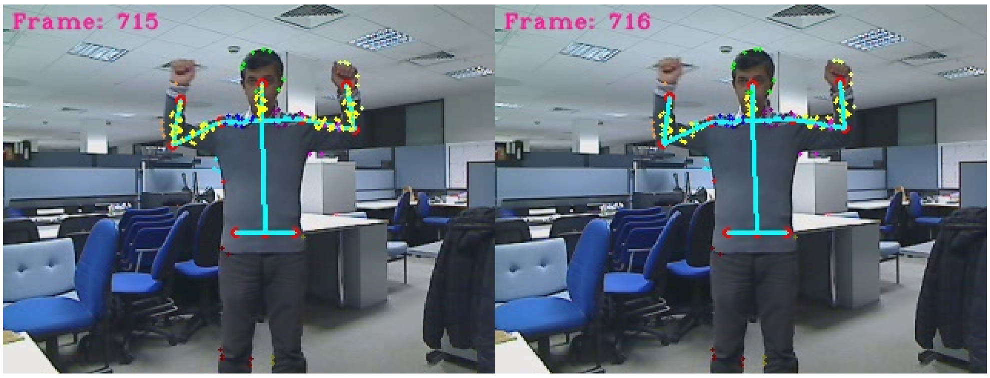
- The limbs with fewer foreground IPs (we call them *weak limbs* for the sake of simplicity), such as the right forearm in Fig. 5.16(f), could be handled more efficiently in the case of using GPSO as a complementary post-processing pose estimator rather than GPSO on its own. In fact, the children joints of a parent limb with a small number of IPs can compensate for the parent joint because the parameter, where the child joint is optimised, can be optimised only when the parent joint has its own optimised parameter.

> When GPSO is used on its own, a weak limb can cause a great amount of ambiguity in the corresponding dimension of the pose space, which causes the pose estimation to get stock in a local optimum -despite the ability of PSO in finding the global optimum. In contrast, the GPSO, after HPSO, needs to search a narrower area of the pose space to refine the pose estimated by HPSO. Therefore, it faces less ambiguity and can tackle the problem of weak limbs.

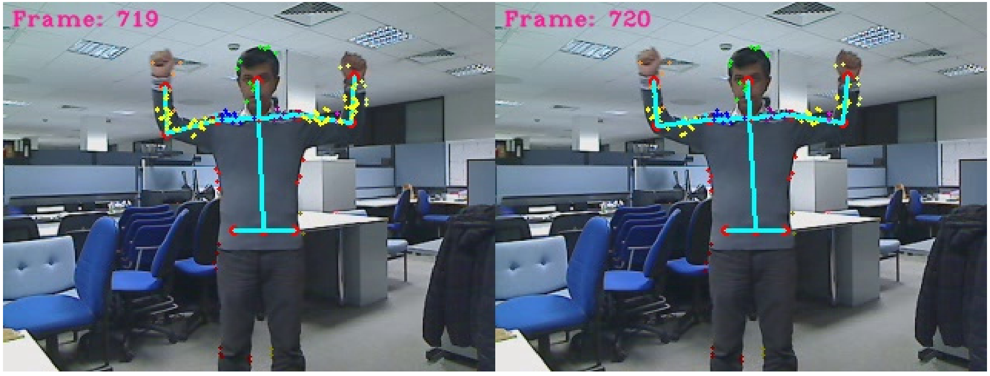
It should be mentioned that we use the same cost function as given by Eq. 5.19 for GPSO. The cost function generates a cost for each pose hypothesis and through the iterations the hypothesis with the lowest cost is selected as the estimated pose. Having the best cost, we can compare that with the best cost of HPSO for the pose. To find an inaccurately estimated parameter, we compare parameter by parameter the best cost of GPSO with the best cost of HPSO. In this way, we select the estimation with the lower best cost and provide the post processed estimated pose vector. Fig. 5.20 shows the results of GPSO on the same frames as in Figs. 5.16 and 5.19. As can be seen, GPSO can amend the results promisingly.

### 5.4.3 Experimental Results

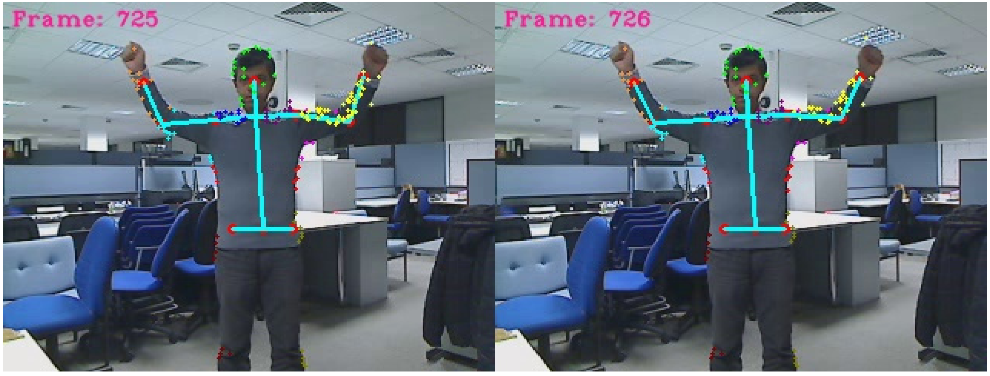
For testing the efficiency of the proposed HUBT algorithm, a test video containing a person acting in front of the camera with a range of pose changes is used (the same video as we used in the previous chapters.). According to the assumption we defined for this thesis in Chapter 1, this video was captured from a static camera. Although automatic pose initialization is an interesting issue (e.g., using the idea of Section 6.2.3) and will help the system to recover in the cases of pose estimation failure, we assume that the initial pose has been estimated manually as in Eq. 5.15. On this basis, we estimate the pose of the subsequent frames. Fig. 5.15 shows the initial pose, which is frame 709 of the test video. For the pose estimation, we have implemented an efficient original PSO library, which can be used in either a hierarchical or global manner. The number of particles and iterations were assumed to be 100 and 10, respectively, for both HPSO and GPSO, which were determined by trying a range of values and picking the best one.



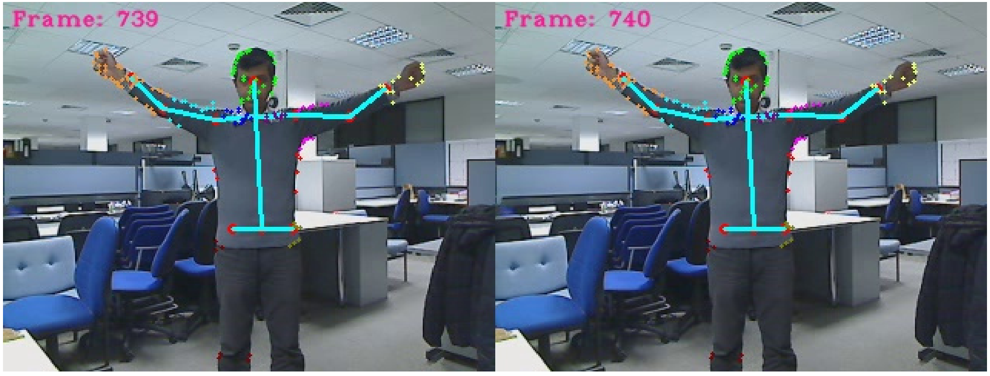
(a) Frame 716.



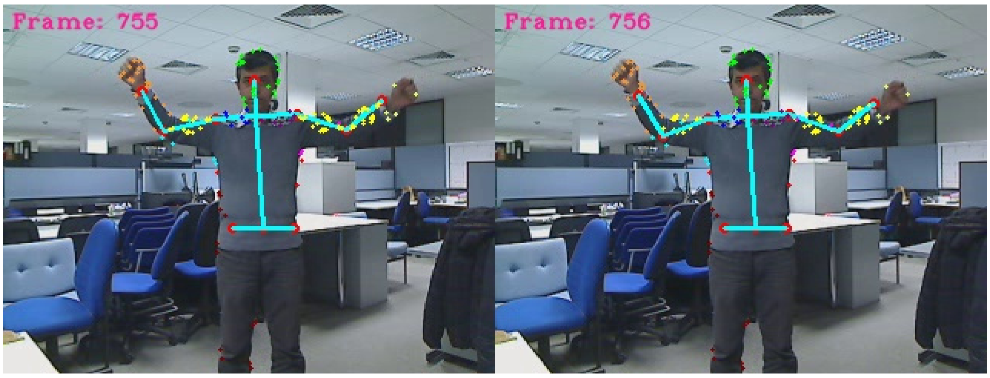
(b) Frame 720.



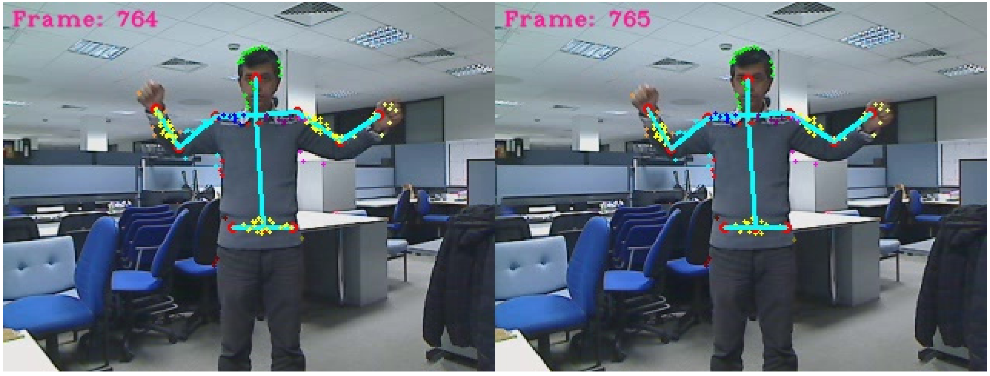
(c) Frame 726.



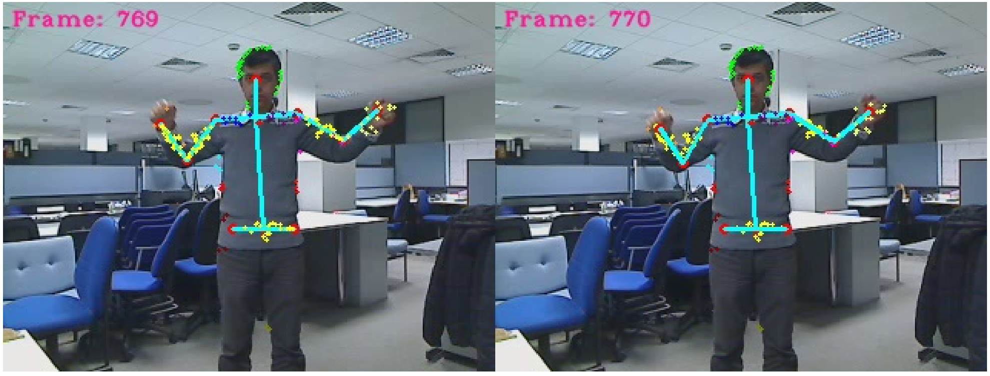
(d) Frame 740.



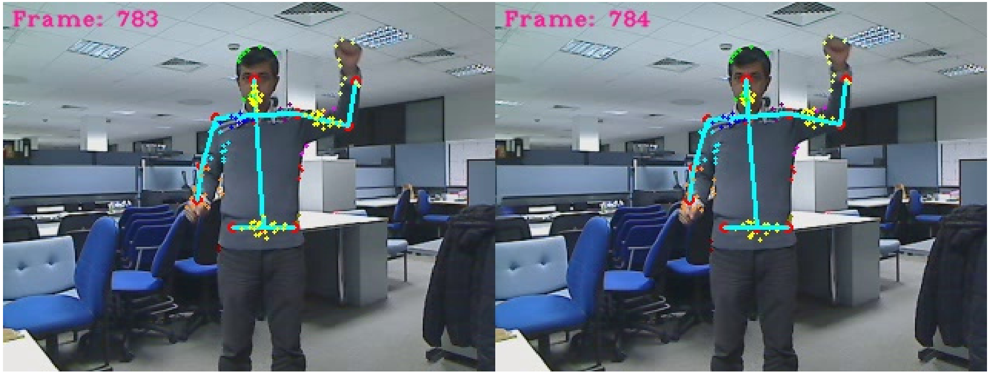
(e) Frame 756.



(f) Frame 765.



(g) Frame 770.



(h) Frame 784.

FIGURE 5.20: 8 non-consecutive rounds of pose estimation from frame 716 to 784, where GPSO amend the result of HPSO in Fig. 5.16 and 5.19.

#### 5.4.3.1 Performance Evaluation

To evaluate the performance of the proposed HUBT algorithm quantitatively, two distance measures are utilized in this section as follows:

1. **The Euclidean Distance:** This is the average Euclidean distance between the joints of the 2D skeletal model (5.4(a)) of the estimated pose and manually marked ground truth (GT) joints, as follows:

$$d = \frac{1}{J}\sum_{i=1}^{J}\sqrt{(\hat{x}_i - x_i^{GT})^2 + (\hat{y}_i - y_i^{GT})^2} \tag{5.23}$$

   where $(\hat{x}_i, \hat{y}_i)$, $(x_i^{GT}, y_i^{GT})$ are the coordinates of the joints from the estimated pose and from the manually marked GT pose and $J$ is the number of joints in the model (which is 11 in this work). Fig. 5.21 shows the resultant Euclidean distances between the estimated pose and the GT pose over several frames of the test video for the HPSO and GPSO stages of the proposed HUBT approach. As can be seen, the proposed two-stage H-G model-based HUBT approach efficiently follows the GT over the frames. Moreover, comparing the distance values of HPSO and GPSO in Fig. 5.21 shows clearly smaller Euclidean distances for GPSO for the tested video, which confirms the refinement role of GPSO after HPSO, and compensate for the error propagation we mentioned in Section 5.4.1.4.

2. **Mean Square Error (MSE):** The MSE is defined between the parameters of the estimated pose vector and the pose vector of manually marked GT pose over all frames of test video as follows:

$$MSE = \frac{1}{T}\sum_{i=1}^{T}\sqrt{(p_i^{(t)} - p_i^{(t),GT})^2} \tag{5.24}$$

   where $(p_i^{(t)}$ is the $i^{th}$ parameter of an estimated pose vector at frame $t$, $p_i^{(t),GT}$ is the $i^{th}$ parameter of GT pose vector at frame $t$, and $T$ is the total number of frames in the test

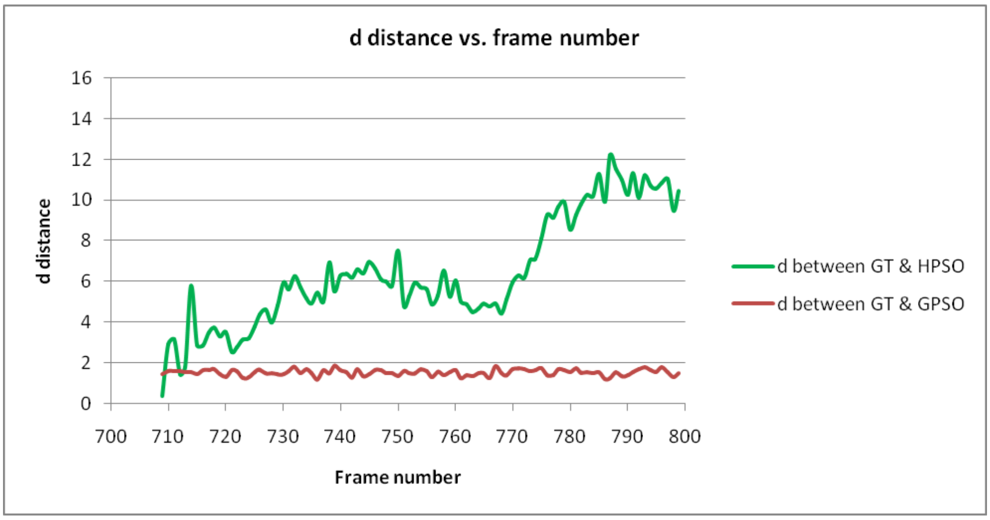


FIGURE 5.21: Euclidean distances between the estimated pose and the GT pose over *several* frames of the test video for the HPSO and GPSO stages of the proposed approach.

video. Table 5.5 shows the average of MSE over all the frames, As can be seen from Table 5.5 GPSO shows a better performance for the tested video in terms of MSE.

TABLE 5.5: Average of MSE over all the frames for HPSO and GPSO.

| **Pose parameters** | **HPSO** | **GPSO** |
|---|---|---|
| $r_x$ | 2.7758 | 1.3258 |
| $r_y$ | 1.5565 | 0.8976 |
| $\theta_{WST}$ | 5.0602 | 1.8794 |
| $\theta_{NCK}$ | 4.4319 | 0.7326 |
| $\theta_{RSH}$ | 11.076 | 2.3327 |
| $\theta_{LSH}$ | 6.128 | 1.958 |
| $\theta_{REB}$ | 50.4823 | 2.359 |
| $\theta_{REB}$ | 13.4955 | 2.125 |

## 5.5 Conclusions and Future Work

In this chapter, we presented a new two stage hierarchical-global model-based human upper body pose estimation and tracking algorithm using interest points in particle swarm optimization framework. The key characteristics of our proposed approach are as follows:

- Firstly, it lies in the way we solve the high-dimensional pose estimation optimisation problem through a *divide-and-conquer* strategy by decomposing the parameters of the pose vector and estimating them separately. Using the constraints that the higher joints in the tree apply to the lower ones, we optimize the corresponding parameters of the joints in the pose vector hierarchically in several levels, from high to low according to the kinematic tree of Fig. 5.3.

- Secondly, it is the post-processing process which is accomplished by applying another PSO-based pose estimator to the estimated pose of the first stage hierarchical pose estimation method to overcome the spatial and temporal propagation errors. This stage unlike the first stage searches the pose space at once for all the parameters of the pose vector. Since, the estimated pose of the first stage is approximately correct for most of the parameters of the pose vector, the second stage PSO searches around the estimated pose of first stage to correct the erroneous parameters.

The proposed algorithms benefit from:

- A hierarchical model-based pose estimation method (HPSO) in the first stage to cut down the complexity and computational cost of a high-dimensional pose estimation optimisation problem.

- A post-processing pose refinement method (GPSO) to compensate for the spatial and temporal propagation errors caused by the first stage.

We applied our approach to a sequence of frames with different levels of articulation and deformations. Experimental results show, promisingly, that the proposed algorithm estimates the human upper body pose using interest points. The proposed algorithm works under the assumptions of manual pose initialization and no occlusion. As we suggest in Chapter 6, the plan is to extend the current algorithm beyond these assumptions in future work.

# Chapter 6

# Overall Conclusions and Future Work

*I am not young enough to know everything.*

OSCAR WILDE, IRISH POET.

In this dissertation, we have proposed a model-based approach for human upper body pose estimation and tracking using interest points in real-time videos. The structure of the proposed system consists of three main blocks, which have been the novel ideas of the authors. Additionally, they can be used standalone in any computer vision application. The proposed algorithms in this thesis have been implemented in C++ using the OpenCV, PCL, and Boost libraries on a Linux-based machine without using GPU[1]. The final overall speed of the system for non-optimised codes is around 5 frame per second which is a promising result. It is expected to achieve higher speeds by optimizing the implementation of the algorithms.

In the following pages, for the last time, we simply review the proposed approaches of the different blocks of the system, as well as the other novel contributions. Finally, the directions for future research will be discussed.

---

[1] Graphic Processing Unit.

## 6.1 Summary of Contributions

The current thesis makes three main novel contributions to the field of computer vision. These contributions, which the proposed model-based human upper body pose estimation and tracking system has been built upon, are categorized into three main categories: (1) an IP-based background subtraction algorithm; (2) an efficient method for matching the IPs of any two consecutive frames, for the sake of tracking; (3) a two-stage hierarchical-global model-based human body tracking algorithm using the IPs in a particle swarm optimization framework. Moreover, there are some minor contributions within the categories, which are the tools for achieving the main purpose of the categories. In this section, we will review these contributions along with the minor contributions we have made in each one.

### 6.1.1 IP-based Background Subtraction (Chapter 3)

The first major contribution of this thesis is the first stage of the proposed pose estimation and tracking system (Fig. 1.2). It is a novel background subtraction algorithm based on IPs, unlike the current background subtraction algorithms, which are mostly pixel-based. The proposed algorithm, furthermore, can be used in any IP-based CV application beyond our project.

Through a block-wise processing strategy, the proposed algorithm divides the frames into blocks of the same size. IPs inside each block are grouped together as Events. Throughout the frame sequence, the Events in each block as well as the numbers of their occurrences are stored using the Repetition Index (RI) in a Binary Tree. The RI is used to classify Events as either background or foreground. The background Events appear significantly more often than foreground Events. Thus, Events with an RI greater than a certain threshold are classified as background, the rest as foreground. This Event classification is used to label IPs in each frame into foreground and background IPs. Experimental results quantitatively show that the proposed algorithm delivers a good discrimination rate in comparison with other BGS approaches. Moreover, it creates a map of the background usable for further processing, it is robust to changes in illumination and can keep itself updated to changes in the background.

### 6.1.2 IP Matching (Chapter 4)

Following the BGS algorithm, we proposed a dynamic hybrid local-spatial IP matching algorithm, which is used to match the foreground IPs of any two consecutive frames for the purpose of human upper body pose estimation. In the local stage, this algorithm finds an initial set of possible matched pairs using the local feature descriptors of the IPs. Then two filtering steps, cross-checking and displacement-checking, are applied to the initial results to increase the matching precision by deleting the mismatched pairs. As a minor contribution, we introduced a method for finding the dynamic displacement check threshold for displacement-checking, based on the kernel density estimation method. It makes the displacement-checking step more robust in cases where faster movements happen between consecutive frames.

The spatial matching stage, which uses the spatial relationship among the IPs, is applied then to the remaining unmatched IPs to improve the recall, whilst holding the precision rate at the same level. Two different spatial matching strategies, i.e. the graph-based and the shape-context-based methods, have been developed for the second stage of the IP matching algorithm.

The proposed approach benefits from:

- local-based IP matching to avoid the expense of the distance and neighbourhood comparison of the spatial-based methods, which is heavy when the matching is started initially with spatial-based method.
- spatial-based IP matching to compensate for the drawback of the local-based methods.
- An IP list scoring and refinement strategy to refine the IP lists and solve the problem of RLL. However, it can also be viewed as a form of tracking idea, which tries to track the trajectories of IPs over the frames. Nevertheless, it is different from the pose estimation and tracking algorithm we used in Chapter 5.

We applied our approach to a sequence of frames with different levels of articulation and deformations. Experimental results are promising and show that not only does the proposed algorithm increase the precision rate from 44.71% for BruteForce to 95.45% for graph-based and 97.41% for SC-based, but that improves also the recall rate from 52.33% for BruteForce to 80.15% for graph-based and 84.96% for SC-based.

### 6.1.3 Two-Stage Hierarchical-Global Model-based Human Upper Body Pose Estimation and Tracking by Means of PSO (Chapter 5)

The final major contribution of this work is a two-stage hierarchical-global model-based articulated human upper body pose estimation and tracking approach based on IPs within a particle swarm optimisation framework. This algorithm, which is a combined bottom-up top-down approach, estimates the skeletal pose of the upper body for each current frame given the pose in the previous frame and the matched foreground IP pairs between the previous and current frames.

The two PSO-based pose estimators in the proposed algorithm solve similarly the optimization problem by hypothesizing a set of uniformly random candidate poses around the predicted position for the solution in the pose search space. To make the comparison between the pose hypotheses and the real observation possible, a set of IPs is rendered for any pose hypothesis through an IP rendering algorithm (which is one of the minor contributions of this thesis). The IP rendering process conveys the spatial relationship between the IPs and the pose in the previous frame to the rendered IPs in the current frame, meantime it applies the perspective transformation between the matched IP pairs of the previous and current frame to the rendered IPs. Homography estimation using the RANSAC method calculates the underlying transformation.

The PSO-based pose estimator, then, evaluates each hypothesis based on a distance measure between the rendered IPs and the real matched IPs of the current frame. The evaluation function (which is another minor contribution) measures the weighted Euclidean distance between the IPs as well as the Hamming distance between the binary census descriptor of the local neighbourhood of IPs. A real positive number less than *1* weights the influences of these two distance measures. Using the result of the evaluation, the position of the pose hypotheses are updated iteratively to find the best pose hypothesis with the lowest distance value.

Another concern, which we solved through another minor contribution, was the scaling factor problem. The human body is changing its distance from the camera throughout the frame sequence and it causes a change in the sizes of the parts of the human body model as well as the whole body, which are not necessarily the same. The proposed scaling factor estimation method calculates the scaling factor for the limbs individually first. Then it combines these together to get the whole body scaling factor. Both the IP rendering and scaling factor estimation rely on another minor proposed contribution, the IP labelling procedure, which assigns a limb of the model to any IP.

Taking advantage of the hierarchical nature of our tree-like kinematic model, we proposed a two-stage hierarchical-global PSO approach to tackle the intrinsic difficulty of solving the optimization problem in the high dimensional pose space at once. Our approach solves this problem through a divide-and-conquer strategy in the first stage, by decomposing the parameters of the pose vector and estimating them separately. Owing to the constraints that the higher joints in the tree apply to the lower ones, our approach optimized the corresponding parameters of the joints in the pose vector hierarchically in several levels, from the higher to lower order. Thereby, it reduces the complexity of the search and overcomes the mentioned drawbacks. Depending on the articulation of the joints and the coherence between them, the optimisation process at each level of hierarchy is accomplished either for the joints individually or for a group of joints together. In contrast, the second stage operates on the pose estimated of the first stage globally at once to do a consistency check and refine the estimated pose. This stage acts as a post-processing stage on the result of the first stage to compensate for the inaccurate estimated parameters of the pose vector.

The proposed algorithms benefit from:

- A hierarchical model-based pose estimation method (HPSO) in the first stage to cut down the complexity and computational cost of a high-dimensional optimisation problem.

- A post-processing pose refinement method (GPSO) to compensate for the spatial and temporal propagation errors caused by the first stage.

## 6.2 Directions for Future Work

The ideas and concepts in this research offer interesting avenues for future research. In the sequel, we identify some of these possibilities as the most important ones:

### 6.2.1 Developing Sparse Depth Extraction (SDE)

Having the background subtraction and IP matching algorithms, we could use them to develop a sparse depth map extraction algorithm. The main ideas behind this algorithm, are:

- **Background Subtraction in Left and Right Cameras**: The BGS algorithm is applied to the $k^{th}$ frame of the left and right cameras in a stereo-rig system, no matter if the cameras are calibrated and rectified. It classifies the IPs of each camera into BG-IPs and FG-IPs.
- **Improving the BGS and IPM algorithms**: The classified IPs of the left and right images are matched to each other using the proposed IP matching algorithms. It is supposed that the left BG-IPs and FG-IPs are matched to the right BG-IPs and FG-IPs, respectively. Probable cases, where some BG-IPs of the left image are matched to the FG-IPs of right one or vice-versa, are hints to show something is going wrong in BGS and/or IPM algorithms. Thus, this information can be used to improve the BGS and IPM algorithms and get more accurate results. In this way, the False Negative (FN) and False Positive (FP) errors of the BGS and IPM algorithms will be decreased.
- **Stereo Rig Calibration and Rectification**: The improved output of the IPM algorithm, then, can be utilized by the SDE algorithm to calibrate and rectify the cameras automatically. This process has to be performed once and needs a few frames of time to be

completed. Afterwards, the rectified images can be applied to the IPM algorithm to improve the result of the matching. It makes the IPM algorithm easier and more accurate for the later frames.

- **Disparity and Depth Extraction**: Using the camera parameters (from the calibration stage) and the disparity map (from the matching and rectification stages), the depth of IPs are calculated.
- **Intelligent Depth Update**: Since we know the BG-IPs and FG-IPs maps, we need to update only the depth for the FG-IPs, unless some changes happen in the background area of the image (adding or removing something to the background). It makes the SDE algorithm faster.

### 6.2.2 Extending the BSG algorithm to Support Moving Camera Situations.

The proposed IP-based BGS algorithm works in static camera conditions. If the camera moves, IPs change their location in the blocks and so the proposed IP-based BGS algorithm would not work any more. For future work, it is planned to adapt the current algorithm to work the moving camera scenarios. To do this, we need to estimate the camera position in space, which is possible using structure from motion. The camera position update yields the IPs position update. Thus, it seems that as in the static camera situation, we can deal with the IPs blockwise to do background subtraction.

In the easiest case, where the camera moves parallel to the scene, IPs are faced with only translation which can be calculated from the new position of the camera. However, in the more complicated situations where the camera has 3D movements, this idea is not so trivial, because IPs come from different depths in the scene, therefore they will translate differently in the image depending on the movement of the camera. Although using the structure-from-motion it is possible to update the location of the camera, the structure of the location of IPs and their spatial relationship cannot be maintained because the IPs will be changed non-linearly. To make the algorithm work in these situations, we need to find a mathematical modelling strategy to compensate for this problem.

### 6.2.3 Automatic Pose Initialization

An automatic human body pose initialization algorithm can be developed based on the proposed IP-based BGS algorithm in this research. This idea should include the following stages:

- **Data Collection**: Firstly, we need to create a dataset of different poses for a range of people with a diversity of ages, genders, sizes, distances from the camera, and clothing. Since the proposed idea is built upon the FG-IPs of the images, a video with several frames is acquired for each person in the set. It permits the algorithm not only to access a range of different poses for the person, but also to do background subtraction.
- **Data Pre-processing**: The videos of the dataset would then input to the BGS algorithm to find the FG-IPs of frames. Then, the frames should be investigated to select some of them as the pre-processed images for the next step. These selected pose images would comprise the pose dataset.
- **Feature Representation**: To represent the poses based on their FG-IPs, the image of each pose would need to be divided into equal size blocks, similar to the idea we used for the BGS algorithm. Hence, the FG-IPs of each pose image fill some blocks of the image. Some attributes of these blocks can be used as features to represent the pose using its FG-IPs.
- **Pose Modelling**: The pose attributes are then could be fed into a learning process to model the pose from the FG-IPs. An improved version of Bag-of-Words such as (Zhang and Mayo, 2010) can be used in this regard.
- **Pose Recognition**: The developed pose model could be used to recognize the pose of the acting person in the image. Whenever the tracking algorithm loses track and loses the pose vector, this pose initialization algorithm can be applied to recover a new pose. This algorithm can be used also to accompany the pose estimation and tracking algorithm to reduce the search space.
- **3D and 2D**: This idea can be implemented in 2D by considering a 2D structure of blocks or a 3D cube of blocks and dealing with the 3D FG-IPs.

### 6.2.4 Occlusion Handling

The spring system idea which Artner et al. (2009) present, can be used for the occlusion handling purpose in our case. Given the labelled foreground IPs of each limb in the previous frame, a spring system can be formed for the IPs of the limb, so that IPs are the masses in the system and the links between the IPs act as the springs. Any movement for any IP change will apply a change to the springs connected to the IP. These changes in the length of the springs can be interpreted as the forces which are applied to them. Thus, the springs will apply the forces to the masses to bring them back to equilibrium. Aggregation of the changes for all the springs and masses moves the IPs toward the equilibrium. Spring systems of the limbs are also connected and apply forces to each other to maintain the kinematic structure of the human body.

In the current frame, some IPs of the limb with occlusion will have disappeared. In this situation, the spring system of the limb generates the same number of imitative IPs for the covered IPs of the limb so that the spatial relationship between the IPs of the limb is maintained. This process is continued until any of the covered IPs reappear. Now, the limb updates that IP for the onwards frames. Across the frames, the scaling factor of the limbs and whole body are calculated and applied to the links between the IPs to update the size of the spring system with the scale changes.

### 6.2.5 Extending the Whole System to 3D

Using the sparse depth extraction of Section 6.2.1, the whole system can be extended into the 3D situation.

# Bibliography

H. Aanæs, A. L. Dahl, and K. S. Pedersen. Interesting interest points. *International Journal of Computer Vision*, 97(1):18–35, 2012.

C. Abras, D. Maloney-Krichmar, and J. Preece. User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, 37(4):445–56, 2004.

A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, 2006.

J. K. Aggarwal and Q. Cai. Human motion analysis: A review. In *Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*, pages 90–102. IEEE, 1997.

J. K. Aggarwal and M. Ryoo. Human activity analysis. *ACM Computing Surveys*, 43(3):1–43, apr 2011. ISSN 03600300. doi: 10.1145/1922649.1922653. URL `http://portal.acm.org/citation.cfm?doid=1922649.1922653`.

W. Aguilar, Y. Frauel, F. Escolano, M. E. Martinez-Perez, A. Espinosa-Romero, and M. A. Lozano. A robust graph transformation matching for non-rigid registration. *Image and Vision Computing*, 27(7):897–910, 2009.

F. Alizadeh, A. Sutherland, and A. Dehghani. A simple and efficient approach for 3d model decomposition. In *8th Iranian Conference on Machine Vision and Image Processing (MVIP 2013)*, pages 1–4. IEEE, 2013.

B. Allen, B. Curless, and Z. Popović. Articulated body deformation from range scan data. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 612–619. ACM, 2002.

D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: shape completion and animation of people. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 408–416. ACM, 2005.

N. Artner, A. Ion, and W. G. Kropatsch. Tracking objects beyond rigid motion. In *Graph-Based Representations in Pattern Recognition*, pages 82–91. Springer, 2009.

N. M. Artner, A. Ion, and W. G. Kropatsch. Multi-scale 2d tracking of articulated objects using hierarchical spring systems. *Pattern Recognition*, 44(4):800–810, 2011.

M. S. Arulampalam, S. Maskell, and N. Gordon. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions ON Signal Processing*, 50: 174–188, 2002.

S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, 2004.

S.-Y. Baek and K. Lee. Parametric human body shape modelling framework for human-centered product design. *Computer-Aided Design*, 44(1):56–67, 2012.

A. O. Bălan and M. J. Black. The naked truth: Estimating body shape under clothing. In *Computer Vision–ECCV 2008*, pages 15–29. Springer, 2008.

A. O. Balan, L. Sigal, M. J. Black, J. E. Davis, and H. W. Haussecker. Detailed human shape and pose from images. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

D. Baltieri, R. Vezzani, and R. Cucchiara. Fast background initialization with recursive hadamard transform. In *7th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS 2010).*, pages 165–171. IEEE, 2010.

J. Bansal, P. Singh, M. Saraswat, A. Verma, S. Jadon, and A. Abraham. Inertia weight strategies in particle swarm optimization. In *3th World Congress on Nature and Biologically Inspired Computing (NaBIC 2011).*, pages 633–640. IEEE, 2011.

C. Barrón and I. A. Kakadiaris. Estimating anthropometry and pose from a single uncalibrated image. *Computer Vision and Image Understanding*, 81(3):269–284, 2001.

H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

Z. Beheshti and S. Shamsudding. A review of population-based meta-heuristic algorithms. *Int. J. Adv. Soft Comput. Appl*, 5:1–35, 2013.

S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *NIPS*, volume 2, page 3, 2000.

Y. Benezeth, P.-M. Jodoin, B. Emile, C. Rosenberger, and H. Laurent. Comparative study of background subtraction algorithms. *Journal of Electronic Imaging*, 19(3):033003–033003, 2010.

H. Bhaskar, L. Mihaylova, and A. Achim. Video foreground detection based on symmetric alpha-stable mixture models. *IEEE Transactions on Circuits and Systems for Video Technology.*, 20(8):1133–1138, 2010.

Z. Bing, Y. Wang, J. Hou, H. Lu, and H. Chen. Research of tracking robot based on surf features. In *Sixth International Conference on Natural Computation (ICNC 2010).*, volume 7, pages 3523–3527. IEEE, 2010.

A. Bottino and A. Laurentini. A silhouette based technique for the reconstruction of human movement. *Computer Vision and Image Understanding*, 83(1):79–95, 2001.

T. Bouwmans. Recent advanced statistical background modeling for foreground detection-a systematic survey. *Recent Patents on Computer Science*, 4(3):147–176, 2011.

T. Bouwmans. Background subtraction for visual surveillance: A fuzzy approach. *Handbook on Soft Computing for Video Surveillance*, pages 103–134, 2012.

T. Bouwmans, F. El Baf, B. Vachon, et al. Background modeling using mixture of gaussians for foreground detection-a survey. *Recent Patents on Computer Science*, 1(3):219–237, 2008.

T. Bouwmans, F. Porikli, B. Höferlin, and A. Vacavant. *Traditional and Recent Approaches in Background Modeling for Foreground Detection: An Overview*. CRC Press, 2014.

G. Bradski. Opencvlibrary. *Dr. Dobb's Journal of Software Tools*, 2000.

V. Brandou, A.-G. Allais, M. Perrier, E. Malis, P. Rives, J. Sarrazin, and P.-M. Sarradin. 3d reconstruction of natural underwater scenes using the stereovision system iris. In *OCEANS 2007 - Europe*, pages 1–6, June 2007. doi: 10.1109/OCEANSE.2007.4302315.

C. Bregler. Learning and recognizing human dynamics in video sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*, pages 568–574. IEEE, 1997.

C. Bregler, J. Malik, and K. Pullen. Twist based acquisition and tracking of animal and human kinematics. *International Journal of Computer Vision*, 56(3):179–194, 2004.

L. G. Brown. A survey of image registration techniques. *ACM computing surveys (CSUR)*, 24 (4):325–376, 1992.

S. S. Bucak, B. Günsel, and O. Gursoy. Incremental non-negative matrix factorization for dynamic background modelling. In *ICEIS 8th International Workshop on Pattern Recognition in Information Systems (PRIS)*, pages 107–116, 2007.

D. E. Butler, V. M. Bove, and S. Sridharan. Real-time adaptive foreground/background segmentation. *EURASIP Journal on Advances in Signal Processing*, 2005(14):2292–2304, 1900.

M. Callieri, P. Cignoni, M. Dellepiane, and R. Scopigno. Pushing time-of-flight scanners to the limit. In *VAST*, pages 85–92, 2009.

J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 569–577. ACM, 2003.

T.-J. Cham and J. M. Rehg. A multiple hypothesis approach to figure tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*, volume 2. IEEE, 1999.

R. Chang, T. Gandhi, and M. M. Trivedi. Vision modules for a multi-sensory bridge monitoring approach. In *The 7th International IEEE Conference on Intelligent Transportation Systems.*, pages 971–976. IEEE, 2004.

L. Chen, H. Wei, and J. Ferryman. A survey of human motion analysis using depth imagery. *Pattern Recognition Letters*, 34(15):1995–2006, 2013.

X. Cheng, N. Li, T. Zhou, L. Zhou, and Z. Wu. A particle swarm optimization algorithm with local sparse representation for visual tracking. *Journal of Computers*, 9(9):2230–2238, 2014.

K. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society*, volume 1, pages I–77. IEEE, 2003.

H. Chui and A. Rangarajan. A new algorithm for non-rigid point matching. In *IEEE Conference on Computer Vision and Pattern Recognition.*, volume 2, pages 44–51. IEEE, 2000.

H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2):114–141, 2003.

M. Clerc and J. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation,.*, 6(1): 58–73, 2002.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, et al. *Introduction to algorithms*, volume 2. MIT press Cambridge, 2001.

M. Cristani and V. Murino. A spatial sampling mechanism for effective background subtraction. In *VISAPP (2)*, pages 403–412, 2007.

M. Cristani and V. Murino. Background subtraction with adaptive spatio-temporal neighborhood analysis. In *The International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 484–489, 2008.

M. Cristani, M. Farenzena, D. Bloisi, and V. Murino. Background subtraction for automated multisensor surveillance: a comprehensive review. *EURASIP Journal on Advances in Signal Processing*, 2010:43–69, 2010.

A. Crivellaro, Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit. [demo] tracking texture-less, shiny objects with descriptor fields. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2014).*, pages 331–332. IEEE, 2014.

R. Y. Da Xu and M. Kemp. Multiple curvature based approach to human upper body parts detection with connected ellipse model fine-tuning. In *16th IEEE International Conference on Image Processing (ICIP)*, pages 2577–2580. IEEE, 2009.

A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 29(6):1052–1067, 2007.

A. Dehghani and A. Sutherland. A combined two-stage local-spatial interest point matching algorithm. In *8th Iranian Conference on Machine Vision and Image Processing (MVIP)*, pages 105–109. IEEE, 2013.

A. Dehghani and A. Sutherland. A dynamic hybrid local-spatial interest point matching algorithm for articulated human body tracking. In *International Conference on Pattern Recognition Applications and Methods (ICPRAM 2014)*, pages 536–543. SCITEPRESS, 2014a.

A. Dehghani and A. Sutherland. A novel interest-point-based background subtraction algorithm. In *ELCVIA: Electronic Letters on Computer Vision and Image Analysis*, volume 13, pages 0050–67, 2014b.

A. Dehghani and A. Sutherland. A two-stage hierarchical-global model-based human upper body pose estimation and tracking using particle swarm optimisation. In *to be submitted to 26th British Machine Vision Conference (BMVC 2015)*, 2015. Manuscript submitted for publication.

A. Dehghani, A. Sutherland, D. Moloney, and D. Pena. An improved interest point matching algorithm for human body tracking. In *1st International Image Processing, Applications and Systems Conference (IPAS 2014).*, pages 1–6. IEEE, 2014.

J. M. del Rincón, D. Makris, C. O. Uruñuela, and J.-C. Nebel. Tracking human position and lower body parts using kalman and particle filters constrained by human biomechanics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics.*, 41(1):26–37, 2011.

K. G. Derpanis. The harris corner detector. *York University*, 2004.

J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61(2):185–205, 2005.

J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *IEEE Conference on Computer Vision and Pattern Recognition.*, volume 2, pages 126–133. IEEE, 2000.

M. Domínguez-Morales, A. Jiménez-Fernández, R. Paz-Vicente, A. Linares-Barranco, and G. Jiménez-Moreno. Stereo matching: From the basis to neuromorphic engineering. *Current Advancements in Stereo Vision*, page 1, 2012.

T. Drummond and R. Cipolla. Real-time tracking of highly articulated structures in the presence of noisy measurements. In *8th IEEE International Conference on Computer Vision, (ICCV 2001).*, volume 2, pages 315–320. IEEE, 2001.

E. Dubrofsky. *Homography estimation*. PhD thesis, University of British Colombia, 2009.

A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *Computer Vision (ECCV 2000)*, pages 751–767. Springer, 2000.

A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90(7):1151–1163, 2002.

R. Fablet and M. J. Black. Automatic detection and tracking of human motion with a view-based representation. In *Computer Vision (ECCV 2002)*, pages 476–491. Springer, 2002.

X. Fang, W. Xiong, B. Hu, and L. Wang. A moving object detection algorithm based on color information. In *Journal of Physics: Conference Series*, volume 48, pages 384–387. IOP Publishing, 2006.

P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

M. Ferreira, A. P. Moreira, P. Malheiros, and N. Pires. Highly flexible robotized cells: aplication to small series painting. In *Proceedings of FAIM*, pages 244–251, 2010.

M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24 (6):381–395, 1981a.

M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24 (6):381–395, 1981b.

C. Forbes, M. Evans, N. Hastings, and B. Peacock. *Statistical distributions*. John Wiley & Sons, 2011.

T. Gao, Z.-g. Liu, W.-c. Gao, and J. Zhang. A robust technique for background subtraction in traffic video. In *Advances in Neuro-Information Processing*, pages 736–744. Springer, 2009.

S. Gauglitz, T. Höllerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International journal of computer vision*, 94(3):335–360, 2011.

D. M. Gavrila. The visual analysis of human movement: A survey. *Computer vision and image understanding*, 73(1):82–98, 1999.

D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action: A multi-view approach. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 73–80. IEEE, 1996.

M. Gevrekci and B. K. Gunturk. Illumination robust interest point detection. *Computer Vision and Image Understanding*, 113(4):565–571, 2009.

P. Gil, G. García, C. Mateo, and F. Torres. Active visual features based on events to guide robot manipulators in tracking tasks. In *19th IFAC World Congress*, pages 11890–11897, 2014.

R. F. Gilberg and B. A. Forouzan. *Data structures: a pseudocode approach with C++, Brooks*. Cole Publishing Co., Pacific Grove, CA, 2001.

G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

L. Gomes, O. R. P. Bellon, and L. Silva. 3d reconstruction methods for digital preservation of cultural heritage: A survey. *Pattern Recognition Letters*, 50:3–14, 2014.

P. E. Greenwood. *A guide to chi-squared testing*, volume 280. John Wiley & Sons, 1996.

S. Gu, Y. Zheng, and C. Tomasi. Efficient visual object tracking with online nearest neighbor classifier. In *Computer Vision (ACCV 2010)*, pages 271–282. Springer, 2011.

A. Gupta, B. Garg, C. Kumar, and D. Behera. An on-line visual human tracking algorithm using surf-based dynamic object model. In *20th IEEE International Conference on Image Processing (ICIP 2013).*, pages 3875–3879. IEEE, 2013.

R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

X. Han. Gait recognition considering walking direction. Master's thesis, University of Rochester, USA, August 2010.

S. Haner and I.-H. Gu. Combining foreground/background feature points and anisotropic mean shift for enhanced visual object tracking. In *20th International Conference on Pattern Recognition (ICPR 2010)*, pages 3488–3491. IEEE, 2010.

R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

W. He, T. Yamashita, H. Lu, and S. Lao. Surf tracking. In *IEEE 12th International Conference on Computer Vision.*, pages 1586–1592. IEEE, 2009.

B. Heisele and C. Woehler. Motion-based recognition of pedestrians. In *14th International Conference on Pattern Recognition.*, volume 2, pages 1325–1330. IEEE, 1998.

L. Herda, P. Fua, R. Plankers, R. Boulic, and D. Thalmann. Skeleton-based motion capture for robust reconstruction of human motion. In *Conference on Computer Animation.*, pages 77–83. IEEE, 2000.

I. P. Herman. *Physics of the human body*. Springer, 2007.

S. Herrero and J. Bescós. Background subtraction techniques: systematic evaluation and comparative analysis. In *Advanced Concepts for Intelligent Vision Systems*, pages 33–42. Springer, 2009.

A. Hilton, D. Beresford, T. Gentils, R. Smith, and W. Sun. Virtual people: Capturing human models to populate virtual worlds. In *Conference on Computer Animation.*, pages 174–185. IEEE, 1999.

H. Hirschmuller and D. Scharstein. Evaluation of cost functions for stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07.*, pages 1–8. IEEE, 2007.

J. Huang, T. Zhang, and D. Metaxas. Learning with structured sparsity. *The Journal of Machine Learning Research*, 12:3371–3412, 2011.

F. Huo, E. Hendriks, P. Paclik, and A. H. Oomes. Markerless human motion capture and pose recognition. In *Image Analysis for Multimedia Interactive Services, 2009. WIAMIS'09. 10th Workshop on*, pages 13–16. IEEE, 2009a.

F. Huo, E. A. Hendriks, A. Oomes, P. van Beek, and R. Veltkamp. Detection tracking and recognition of human poses for a real time spatial game. In *CASA Workshop on 3D Advanced Media In Gaming and Simulation*, pages 43–51, 2009b.

Š. Ivekovič, E. Trucco, and Y. R. Petillot. Human body pose estimation with particle swarm optimisation. *Evolutionary Computation*, 16(4):509–528, 2008.

X. Ji and H. Liu. Advances in view-invariant human motion analysis: A review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(1):13–24, 2010.

V. John, E. Trucco, and S. Ivekovic. Markerless human articulated tracking using hierarchical particle swarm optimisation. *Image and Vision Computing*, 28(11):1530–1547, 2010.

S. X. Ju, M. J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *2nd International Conference on Automatic Face and Gesture Recognition.*, pages 38–44. IEEE, 1996.

I. A. Kakadiaris and D. Metaxas. Three-dimensional human body model acquisition from multiple views. *International Journal of Computer Vision*, 30(3):191–218, 1998.

A. H. Kashan and B. Karimi. A discrete particle swarm optimization algorithm for scheduling parallel machines. *Computers & Industrial Engineering*, 56(1):216–223, 2009.

R. Kehl and L. V. Gool. Markerless tracking of complex human motions from multiple views. *Computer Vision and Image Understanding*, 104(2):190–209, 2006.

J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

J. Kim, M. Murshed, A. R. Rivera, and O. Chae. Background modelling using edge-segment distributions. *International Journal of Advanced Robotic Systems*, 10, 2013.

K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis. Background modeling and subtraction by codebook construction. In *International Conference on Image Processing (ICIP'04).*, volume 5, pages 3061–3064. IEEE, 2004.

O. King and D. A. Forsyth. How does condensation behave with a finite number of samples? In *Computer Vision (ECCV 2000)*, pages 695–709. Springer, 2000.

G. Klein and D. Murray. Improving the agility of keyframe-based slam. In *Computer Vision (ECCV 2008)*, pages 802–815. Springer, 2008.

D. Klimentjew, N. Hendrich, and J. Zhang. Multi sensor fusion of camera and 3d laser range finder for object recognition. In *IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2010).*, pages 236–241. IEEE, 2010.

T. Kobayashi, K. Nakagawa, J. Imae, and G. Zhai. Real time object tracking on video image sequence using particle swarm optimization. In *International Conference on Control, Automation and Systems (ICCAS 2007).*, pages 1773–1778. IEEE, 2007.

A. D. Kuo. A mechanical analysis of force distribution between redundant, multiple degree-of-freedom actuators in the human: Implications for the central nervous system. *Human movement science*, 13(5):635–663, 1994.

B. Kwolek, T. Krzeszowski, A. Gagalowicz, K. Wojciechowski, and H. Josinski. Real-time multi-view human motion tracking using particle swarm optimization with resampling. In *Articulated Motion and Deformable Objects*, pages 92–101. Springer, 2012.

X. Lan and D. P. Huttenlocher. Beyond trees: Common-factor models for 2d human pose recovery. In *10th IEEE International Conference on Computer Vision, (ICCV 2005).*, volume 1, pages 470–477. IEEE, 2005.

B. Lee and M. Hedley. Background estimation for video surveillance. In *Image and Vision Computing New Zealand*, pages 315–320, 2002.

N. H. Lehment, M. Kaiser, D. Arsic, and G. Rigoll. Cue-independent extending inverse kinematics for robust pose estimation in 3d point clouds. In *17th IEEE International Conference on Image Processing (ICIP)*, pages 2465–2468. IEEE, 2010.

Y. Lei, H. Pan, W. Chen, and C. Gao. A new hierarchical method for markerless human pose estimation. In *Computer Vision Systems*, pages 163–172. Springer, 2013.

C. E. Leiserson, R. L. Rivest, C. Stein, and T. H. Cormen. *Introduction to algorithms*. The MIT press, 2001.

S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *IEEE International Conference on Computer Vision (ICCV).*, pages 2548–2555. IEEE, 2011.

B. Li, Q. Meng, and H. Holstein. Point pattern matching and applications-a review. In *IEEE International Conference on Systems, Man and Cybernetics.*, volume 1, pages 729–736. IEEE, 2003.

B. Li, Q. Meng, and H. Holstein. Articulated motion reconstruction from feature points. *Pattern Recognition*, 41(1):418–431, 2008a.

X. Li, W. Hu, Z. Zhang, and X. Zhang. Robust foreground segmentation based on two effective background models. In *1st ACM international conference on Multimedia information retrieval*, pages 223–228. ACM, 2008b.

Y. Li and Z. Sun. Articulated human motion tracking using sequential immune genetic algorithm. *Mathematical Problems in Engineering*, 2013.

Y. Li, Y. Tsin, Y. Genc, and T. Kanade. Object detection using 2d spatial ordering constraints. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition(CVPR 2005).*, volume 2, pages 711–718. IEEE, 2005.

Z. Li and D. Kulic. A stereo camera based full body human motion capture system using a partitioned particle filter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*, pages 3428–3434. IEEE, 2010.

H.-H. Lin, T.-L. Liu, and J.-H. Chuang. A probabilistic svm approach for background scene initialization. In *International Conference on Image Processing.*, volume 3, pages 893–896. IEEE, 2002.

Z. Liu, J. An, and Y. Jing. A simple and robust feature point matching algorithm based on restricted spatial order constraints for aerial image registration. *IEEE Transactions on Geoscience and Remote Sensing.*, 50(2):514–527, 2012.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

J. Ma, J. Zhao, J. Tian, Z. Tu, and A. L. Yuille. Robust estimation of non-rigid transformation for point set registration. In *Proceedings of IEEE conference on Computer Vision and Pattern Recognition*. IEEE, 2013.

J. Ma, J. Zhao, J. Tian, A. L. Yuille, and Z. Tu. Robust point matching via vector field consensus. *IEEE Transactions on Image Processing*, 23(4):1706–1721, 2014.

J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *Computer Vision (ECCV 2000)*, pages 3–19. Springer, 2000.

M. Maes. On a cyclic string-to-string correction problem. *Information Processing Letters*, 35 (2):73–78, 1990.

S. Maji. A comparison of feature descriptors. *Report, University of California, Berkeley*, 2006.

A. Manzanera. Human motion analysis: Tools, models, algorithms and applications. 2009.

N. J. McFarlane and C. P. Schofield. Segmentation and tracking of piglets in images. *Machine Vision and Applications*, 8(3):187–193, 1995.

I. Mikić, M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. *International Journal of Computer Vision*, 53(3):199–223, 2003.

K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

T. B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.

T. B. Moeslund, A. Hilton, V. Krüger, and L. Sigal. *Visual Analysis of Humans*. Springer, 2011.

J. J. Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

G. Mori, S. Belongie, and J. Malik. Efficient shape matching using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1832–1837, 2005.

Q. Muhlbauer, K. Kuhnlenz, and M. Buss. A model-based algorithm to estimate body poses using stereo vision. In *The 17th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2008).*, pages 285–290. IEEE, 2008.

P. Munro and A. P. Gerdelan. Stereo vision computer depth perception. *Country United States City University Park Country Code US Post code*, 16802, 2009.

Z. Musa and J. Watada. Video tracking system: A survey. *ICIC Express Letters, An International Journal of Research and Surveys*, 2(1):65–72, 2008.

R. Navaratnam, A. Thayananthan, P. Torr, and R. Cipolla. Hierarchical part-based human body pose estimation. 2005.

H. Ning, T. Tan, L. Wang, and W. Hu. Kinematics-based tracking of human walking in monocular video sequences. *Image and Vision Computing*, 22(5):429–441, 2004.

J. F. Nunes, P. M. Moreira, and J. M. R. Tavares. A survey on human motion analysis and simulation tools. In *International Conference on Computational and Experimental Biomedical Sciences (ICCEBS 2013)*, 2013.

C.-m. Oh, M. Islam, and C.-W. Lee. Pictorial structures-based upper body tracking and gesture recognition. In *17th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV 2011).*, pages 1–6. IEEE, 2011.

I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Tracking the articulated motion of two strongly interacting hands. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012).*, pages 1862–1869. IEEE, 2012.

N. M. Oliver, B. Rosario, and A. P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 22 (8):831–843, 2000.

C. F. Olson and H. Abi-Rached. Wide-baseline stereo vision for terrain mapping. *Machine Vision and Applications*, 21(5):713–725, 2010.

D. L. Olson and D. Delen. *Advanced data mining techniques*. Springer Publishing Company, Incorporated, 2008.

M. Omran, A. P. Engelbrecht, and A. Salman. Particle swarm optimization method for image clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(03): 297–321, 2005.

J. O'Rourke and N. I. Badler. Model-based image analysis of human motion using constraint propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):522–536, 1980.

A. Pande, A. Verma, A. Mittal, and A. Agrawal. Network aware efficient resource allocation for mobile-learning video systems. *International. Conference on Mobile Learning*, pages 189–196, 2007.

N. Paragios and R. Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):266–280, 2000.

E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

F. Pascal and R. Plankers. Articulated soft objects for multiview shape and motion capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1182–1187, 2003.

H. A. Perlin, H. S. Lopes, and T. M. Centeno. Particle swarm optimization for object recognition in computer vision. In *New Frontiers in Applied Artificial Intelligence*, pages 11–21. Springer, 2008.

M. Pinto, A. P. Moreira, P. Costa, M. Ferreira, and P. Malheiros. Robotic manipulator and artificial vision system for picking cork pieces in a conveyor belt. In *10th Conference on Mobile Robots and Competitions, Robotica*, 2010.

M. I. Pitman and L. Peterson. Biomechanics of skeletal muscle. *Basic biomechanics of the musculoskeletal system*, pages 89–114, 1989.

R. Plankers and P. Fua. Articulated soft objects for multi-view shape and motion capture. *IEEE PAMI*, 25(10), 2003.

R. Poli. An analysis of publications on particle swarm optimization applications. Technical Report CSM-469, University of Essex, Department of Computer Science, 2007.

R. Poli. Mean and variance of the sampling distribution of particle swarm optimizers during stagnation. *IEEE Transactions on Evolutionary Computation.*, 13(4):712–721, 2009.

R. Poppe. Vision-based human motion analysis: An overview. *Computer vision and image understanding*, 108(1):4–18, 2007.

F. Porikli and C. Wren. Change detection by frequency decomposition: Wave-back. In *Workshop on Image Analysis for Multimedia Interactive Services*, 2005.

D. Ramanan and D. A. Forsyth. Finding and tracking people from the bottom up. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*, volume 2, pages 467–474. IEEE, 2003.

F. Reuleaux. *The kinematics of machinery: outlines of a theory of machines*. Dover, 1963.

T. J. Roberts, S. J. McKenna, and I. W. Ricketts. Human tracking using 3d surface colour distributions. *Image and Vision Computing*, 24(12):1332–1342, 2006.

C. Robertson and E. Trucco. Human body posture via hierarchical evolutionary optimization. In *British Machine Vision Conference (BMVC 2006)*, pages 999–1008, 2006.

L. F. Rocha, M. Ferreira, V. Santos, and A. P. Moreira. Object recognition and pose estimation for industrial applications: A cascade system. *Robotics and Computer-Integrated Manufacturing*, 30(6):605–621, 2014.

R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In *Computer Vision—ECCV 2002*, pages 700–714. Springer, 2002.

E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006*, pages 430–443. Springer, 2006.

P. J. Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880, 1984.

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *IEEE International Conference on Computer Vision (ICCV 2011)*, pages 2564–2571. IEEE, 2011.

S. Saini, D. R. B. A. Rambli, S. B. Sulaiman, and M. N. B. Zakaria. A study of stochastic algorithms for 3d articulated human body tracking. In *IEEE 2nd International Conference on Image Information Processing (ICIIP 2013).*, pages 84–89. IEEE, 2013.

G. Sansoni, M. Trebeschi, and F. Docchio. State-of-the-art and applications of 3d imaging sensors in industry, cultural heritage, medicine, and criminal investigation. *Sensors*, 9(1): 568–601, 2009.

N. Saunier and T. Sayed. A feature-based tracking algorithm for vehicles in intersections. In *The 3rd Canadian Conference on Computer and Robot Vision, 2006.*, pages 59–59. IEEE, 2006.

A. Saxena, J. Schulte, and A. Y. Ng. Depth estimation using monocular and stereo cues. In *IJCAI*, volume 7, pages 2197–2203, 2007.

A. Saxena, S. H. Chung, and A. Y. Ng. 3-d depth reconstruction from a single still image. *International Journal of Computer Vision*, 76(1):53–69, 2008a.

A. Saxena, M. Sun, and A. Y. Ng. Make3d: Depth perception from a single still image. In *AAAI*, pages 1571–1576, 2008b.

K. Schindler and H. Wang. Smooth foreground-background segmentation for video processing. In *Computer Vision–ACCV 2006*, pages 581–590. Springer, 2006.

N. A. Setiawan, H. Seok-Ju, K. Jang-Woon, and L. Chil-Woo. Gaussian mixture model in improved hls color space for human silhouette extraction. In *Advances in Artificial Reality and Tele-Existence*, pages 732–741. Springer, 2006.

L. Shapiro and G. C. Stockman. Computer vision. *Prentice Hall*, 2001.

J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (CVPR'94).*, pages 593–600. IEEE, 1994.

Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *IEEE World Congress on Computational Intelligence Evolutionary Computation.*, pages 69–73. IEEE, 1998.

J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *Computer Vision—ECCV 2000*, pages 702–718. Springer, 2000.

M. Sigalas, H. Baltzakis, and P. Trahanias. Visual tracking of independently moving body and arms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009).*, pages 3005–3010. IEEE, 2009.

M. Sigalas, H. Baltzakis, and P. Trahanias. Gesture recognition based on arm tracking for human-robot interaction. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5424–5429. IEEE, 2010.

D. Skocaj, M. Uray, A. Leonardis, and H. Bischof. Why to combine reconstructive and discriminative information for incremental subspace learning. In *Computer Vision Winter Workshop*, pages 1–6, 2006.

C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *The International Journal of Robotics Research*, 22(6):371–391, 2003.

J. Starck and A. Hilton. Model-based multiple view reconstruction of people. In *9th IEEE International Conference on Computer Vision.*, pages 915–922. IEEE, 2003.

J. Starck, G. Collins, R. Smith, A. Hilton, and J. Illingworth. Animated statues. *Machine Vision and Applications*, 14(4):248–259, 2003.

C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*, volume 2. IEEE, 1999.

D.-N. Ta, W.-C. Chen, N. Gelfand, and K. Pulli. Surftrac: Efficient tracking and continuous object recognition using local feature descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009).*, pages 2937–2944. IEEE, 2009.

M. F. Tasgetiren, P. N. Suganthan, and Q.-Q. Pan. A discrete particle swarm optimization algorithm for the generalized traveling salesman problem. In *9th annual conference on Genetic and evolutionary computation*, pages 158–167. ACM, 2007.

H. Tezuka and T. Nishitani. A precise and stable foreground segmentation using fine-to-coarse approach in transform domain. In *15th IEEE International Conference on Image Processing (ICIP 2008).*, pages 2732–2735. IEEE, 2008.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

C. Tomasi and T. Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991.

P. H. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *The 7th IEEE International Conference on Computer Vision.*, volume 1, pages 255–261. IEEE, 1999.

J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory.*, 53(12):4655–4666, 2007.

D.-M. Tsai and S.-C. Lai. Independent component analysis-based background subtraction for indoor surveillance. *IEEE Transactions on Image Processing*, 18(1):158–167, 2009.

F. Van Den Bergh. *An analysis of particle swarm optimizers*. PhD thesis, University of Pretoria, 2006.

P. Varcheie, M. Sills-Lavoie, and G.-A. Bilodeau. An efficient region-based background subtraction technique. In *Canadian Conference on Computer and Robot Vision CRV 2008).*, pages 71–78. IEEE, 2008.

J. Varona, J. Gonzàlez, I. Rius, and J. J. Villanueva. Importance of detection for video surveillance applications. *Optical Engineering*, 47(8):087201–087201, 2008.

A. Von Hirsch. *Censure and sanctions*. Clarendon press Oxford, 1993.

R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.

H. Wang and D. Suter. A re-evaluation of mixture of gaussian background modelling [video signal processing applications]. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05).*, volume 2, pages 1017–1020. IEEE, 2005.

J. Wang, W. Gao, S. Shan, and X. Hu. Facial feature tracking combining model-based and model-free method. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP'03).*, volume 3, pages III–205. IEEE, 2003a.

J. Wang, Z. Kuang, X. Xu, and Y. Zhou. Discrete particle swarm optimization based on estimation of distribution for polygonal approximation problems. *Expert Systems with Applications*, 36(5):9398–9408, 2009.

L. Wang, W. Hu, and T. Tan. Recent developments in human motion analysis. *Pattern recognition*, 36(3):585–601, 2003b.

Y. Wang. Feature point correspondence between consecutive frames based on genetic algorithm. *International Journal of Robotics and Automation*, 21(1):35–38, 2006.

Z. Wang, M. B. Salah, H. Zhang, and N. Ray. Shape based appearance model for kernel tracking. *Image and Vision Computing*, 30(4):332–344, 2012.

G.-J. Wen, J.-j. Lv, and W.-x. Yu. A high-performance feature-matching method for image registration by combining spatial and similarity information. *IEEE Transactions on Geoscience and Remote Sensing*, 46(4):1266–1277, 2008.

B. White and M. Shah. Automatically tuning background subtraction parameters using particle swarm optimization. In *IEEE International Conference on Multimedia and Expo.*, pages 1826–1829. IEEE, 2007.

Wikipedia. Degrees of freedom (mechanics). `http://en.wikipedia.org/wiki/Degrees_of_freedom_(mechanics)`.

C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 19(7): 780–785, 1997.

W.-Y. Wu and M.-J. J. Wang. Two-dimensional object recognition through two-stage string matching. *IEEE Transactions on Image Processing.*, 8(7):978–981, 1999.

M. Xiao, C. Han, and X. Kang. A background reconstruction for dynamic scenes. In *9th International Conference on Information Fusion.*, pages 1–7. IEEE, 2006.

J. Xin, G. Chen, and Y. Hai. A particle swarm optimizer with multi-stage linearly-decreasing inertia weight. In *The International Joint Conference on Computational Sciences and Optimization*, volume 1, pages 505–508. IEEE Computer Society, 2009.

M. Yamamoto and K. Yagishita. Scene constraints-aided tracking of human body. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 151–156. IEEE, 2000.

M. Ye, Q. Zhang, L. Wang, J. Zhu, R. Yang, and J. Gall. A survey on human motion analysis from depth data. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*, pages 149–187. Springer, 2013.

A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm Computing Surveys (CSUR)*, 38(4):13, 2006.

R. Zabih and J. Woodfill. A non-parametric approach to visual correspondence. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Citeseer, 1996.

E. Zhang and M. Mayo. Improving bag-of-words model with spatial information. In *25th International Conference of Image and Vision Computing New Zealand (IVCNZ 2010).*, pages 1–8. IEEE, 2010.

X. Zhang, W. Hu, S. Maybank, X. Li, and M. Zhu. Sequential particle swarm optimization for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008).*, pages 1–8. IEEE, 2008.

Z. Zhang, H. S. Seah, and C. K. Quah. Particle swarm optimization for markerless full body motion capture. In *Handbook of Swarm Intelligence*, pages 201–220. Springer, 2010.

H. Zheng, J. Jie, B. Hou, and Z. Fei. A multi-swarm particle swarm optimization algorithm for tracking multiple targets. In *IEEE 9th Conference on Industrial Electronics and Applications (ICIEA 2014).*, pages 1662–1665. IEEE, 2014.

J. Zheng, Y. Wang, N. L. Nihan, and M. E. Hallenbeck. Extracting roadway background image: Mode-based approach. *Transportation Research Record: Journal of the Transportation Research Board*, 1944(1):82–88, 2006.

Y. Zheng and D. Doermann. Robust point matching for nonrigid shapes by preserving local neighborhood structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 28(4):643–649, 2006.

H. Zhou, Y. Yuan, and C. Shi. Object tracking using sift features and mean shift. *Computer Vision and Image Understanding*, 113(3):345–352, 2009.

# Appendix A

# Kinematic Human Body Model

> *A room without books is like a body without a soul.*
>
> MARCUS TULLIUS CICERO

We have implemented the mentioned skeletal human body model in this thesis in an object-oriented form using a C++ class. It enables us to add the model into any tracking project without loss-of-generality. The class defines a pose vector, joints, and limbs. The model has been designed to be used in full body or upper body versions, which is controlled simply using a boolean variable. To follow the hierarchical structure of the model and transfer the coordinate of any point in the body from its local coordinate system to the reference coordinate system (REF), several translation and rotation matrices also have been considered. The pictorial model is also supported by this implementation. Different tasks related to the model and the pose estimation have been implemented using several methods started from Line 168 of implementation below. Below, the implemented model is presented as:

```
1 class model2D {
2
3 private:
4
5 public:
6   // Constants and coefficients
7   // ------------------------------------------
8   double S; // Scale Factor.
```

```
double H; // The human body height.
double x_wst, y_wst;
bool FullBody; // true: Whole Body, False: Upper Body
double Width, Height;

// Limbs L0: WST2NCK, L1: NCK2HED, L2: NCK2RSH, L3: NCK2LSH, L4: RSH2REB
//       L5: LSH2LEB, L6: REB2RWR, L7: LEB2LWR, L8: WST2RHP, L9: WST2LHP
// ----------------------------------------------------------
// Upper Body
double WST2NCK, NCK2HED, NCK2RSH, NCK2LSH, RSH2REB, LSH2LEB, REB2RWR, LEB2LWR;
// Lower Body
double WST2RHP, WST2LHP, HP2KN, KN2AN;
double ARM, FARM, WRT, THS, THL, LGS, LGL, FOOT, HEDL, HEDS;

// Joints: 0- WST 1- NCK 2- HED 3- RSH 4- LSH 5- REB 6- LEB 7- RWR 8- LWR 9- RHB 10-LHB
// ----------------------------------------------------------
// Upper Body
Point2f O_WST, O_NCK, O_HED, O_RSH, O_LSH, O_REB, O_LEB, O_RWR, O_LWR;
// Lower Body
Point2f O_RHP, O_LHP, O_RKN, O_LKN, O_RAN, O_LAN;

// Angles
// ----------------------------------------------------------
// Upper Body
double teta_WST, teta_NCK, teta_RSH, teta_LSH, teta_REB, teta_LEB;
// Lower Body
double teta_RHP, teta_LHP, teta_RKN, teta_LKN;

// Translation and Rotation Matrices
// ----------------------------------------------------------
// Upper Body
Mat_<double> WST_r; Mat_<double> WST_t;
Mat_<double> NCK_r; Mat_<double> NCK_t;
Mat_<double> RSH_r; Mat_<double> RSH_t;
Mat_<double> LSH_r; Mat_<double> LSH_t;
Mat_<double> REB_r; Mat_<double> REB_t;
Mat_<double> LEB_r; Mat_<double> LEB_t;
// Lower Body
Mat_<double> RHP_r; Mat_<double> RHP_t;
Mat_<double> LHP_r; Mat_<double> LHP_t;
Mat_<double> RKN_r; Mat_<double> RKN_t;
Mat_<double> LKN_r; Mat_<double> LKN_t;

// Pose = [x_wst, y_wst, teta_WST, teta_NCK, teta_RSH, teta_LSH, teta_REB, teta_LEB]
// ----------------------------------------------------------
Mat_<double> Pose;

vector<Point2f> Jnts_p;   // previous joints
vector<Point2f> Jnts_e;   // estimated joints

// The current frame
// ----------------------------------------------------------
```

```
  Mat Image;

  // constructor definition - Initialization of the model
  // ----------------------------------------------------
  model2D(Mat &img) {
    // prepare arguments for 2D model Constants and coefficients initialization
    Width=  img.cols; // 320;
    Height= img.rows; // 240;
    S= 1;       // The human body height = HEIGHT
    H= Height;  // The human body height = HEIGHT
    x_wst = Width/2;
    y_wst=  Height/2;
    FullBody = false; // Upper body

    // Limbs initialization
    // ----------------------------------------------------
    // Upper Body
    WST2NCK= 0.358 * H * S;
    NCK2HED= 0.116 * H * S;
    NCK2RSH= 0.130 * H * S;
    NCK2LSH= 0.130 * H * S;
    RSH2REB= 0.158 * H * S;
    LSH2LEB= 0.158 * H * S;
    REB2RWR= 0.145 * H * S;
    LEB2LWR= 0.145 * H * S;
    // Lower Body
    WST2RHP= 0.095 * H * S;
    WST2LHP= 0.095 * H * S;
    HP2KN   = 0.245 * H * S;
    KN2AN   = 0.246 * H * S;

    // Pictorial Sizes
    // ----------------------------------------------------
    ARM     = 0.070 * H * S;
    FARM    = 0.72  * ARM;
    WRT     = 0.43  * ARM;
    // Lower Body
    THS     = 0.014 * H * S;
    THL     = 0.062 * H * S;
    LGS     = 0.014 * H * S;
    LGL     = 0.035 * H * S;
    FOOT    = 0.055 * H * S;
    HEDL    = 0.100 * H * S;
    HEDS    = 0.065 * H * S;

    // Joints initialization
    // ----------------------------------------------------
    // (Upper Body)
    O_WST= Point2f(x_wst     ,  y_wst     );
    O_NCK= Point2f(WST2NCK   ,  0         );
    O_HED= Point2f(NCK2HED   ,  0         );
    O_RSH= Point2f(0         , -NCK2RSH   );
    O_LSH= Point2f(0         ,  NCK2LSH   );
    O_REB= Point2f(0         ,  RSH2REB   );
    O_LEB= Point2f(0          ,  LSH2LEB );
    O_RWR= Point2f(REB2RWR    ,   0      );
    O_LWR= Point2f(LEB2LWR    ,   0      );

    // (Lower Body)
    O_RHP= Point2f(0          , -WST2RHP );
    O_LHP= Point2f(0          ,  WST2LHP );
    O_RKN= Point2f(HP2KN      ,  0       );
    O_LKN= Point2f(HP2KN      ,  0       );
    O_RAN= Point2f(KN2AN      ,  0       );
    O_LAN= Point2f(KN2AN      ,  0       );

    // Angles initialization
    // ----------------------------------------
    // Upper Body
    teta_WST= 0; teta_NCK= 0; teta_RSH= 90; teta_LSH= 90; teta_REB= 0;
    teta_LEB= 0;
    // Lower Body
    teta_RHP= 0; teta_LHP= 0; teta_RKN= 0; teta_LKN= 0;

    // Matrices initialization
    // ----------------------------------------
    // Upper Body Rotation
    WST_r= Mat::zeros(3,3,CV_8UC1); WST_r(0,1)= 1; WST_r(1,0)=-1; WST_r
    (2,2)=1;
    NCK_r= Mat::eye(3,3,CV_8UC1)   ; // teta_WST=0
    RSH_r= Mat::eye(3,3,CV_8UC1)   ; RSH_r(0,0)=-1; RSH_r(1,1)=-1;
    LSH_r= Mat::eye(3,3,CV_8UC1)   ; LSH_r(0,0)=-1;
    REB_r= Mat::zeros(3,3,CV_8UC1); REB_r(0,1)=-1; REB_r(1,0)= 1; REB_r
    (2,2)=1; // teta_RSH=90
    LEB_r= Mat::zeros(3,3,CV_8UC1); LEB_r(0,1)=-1; LEB_r(1,0)= 1; LEB_r
    (2,2)=1; // teta_LSH=90
    // Upper Body Translation
    WST_t= Mat::eye(3,3,CV_8UC1)   ; WST_t(0,2)=O_WST.x; WST_t(1,2)=O_WST.y;
    NCK_t= Mat::eye(3,3,CV_8UC1)   ; NCK_t(0,2)=O_NCK.x; NCK_t(1,2)=O_NCK.y;
    RSH_t= Mat::eye(3,3,CV_8UC1)   ; RSH_t(0,2)=O_RSH.x; RSH_t(1,2)=O_RSH.y;
    LSH_t= Mat::eye(3,3,CV_8UC1)   ; LSH_t(0,2)=O_LSH.x; LSH_t(1,2)=O_LSH.y;
    REB_t= Mat::eye(3,3,CV_8UC1)   ; REB_t(0,2)=O_REB.x; REB_t(1,2)=O_REB.y;
    LEB_t= Mat::eye(3,3,CV_8UC1)   ; LEB_t(0,2)=O_LEB.x; LEB_t(1,2)=O_LEB.y;

    // Lower Body Rotation
    RHP_r= Mat::eye(3,3,CV_8UC1)   ; RHP_r(0,0)=-1; RHP_r(1,1)=-1;
    LHP_r= Mat::eye(3,3,CV_8UC1)   ; LHP_r(0,0)=-1;
    RKN_r= Mat::eye(3,3,CV_8UC1)   ; // teta_RKN=0
    LKN_r= Mat::eye(3,3,CV_8UC1)   ; // teta_LKN=0
    // Lower Body Translation
    RHP_t= Mat::eye(3,3,CV_8UC1)   ; RHP_t(0,2)=O_RHP.x; RHP_t(1,2)=O_RHP.y;
    LHP_t= Mat::eye(3,3,CV_8UC1)   ; LHP_t(0,2)=O_LHP.x; LHP_t(1,2)=O_LHP.y;
    RKN_t= Mat::eye(3,3,CV_8UC1)   ; RKN_t(0,2)=O_RKN.x; RKN_t(1,2)=O_RKN.y;
    LKN_t= Mat::eye(3,3,CV_8UC1)   ; LKN_t(0,2)=O_LKN.x; LKN_t(1,2)=O_LKN.y;

    // Pose Initialization
    // ----------------------------------------
    Pose = (Mat_<double>(8,1) << x_wst, y_wst, teta_WST, teta_NCK, teta_RSH
    , teta_LSH, teta_REB, teta_LEB);
    }

    // ------------------------------------------------
    // methods
    // ------------------------------------------------
    // 0- Initialize Model
    void
    InitializeModel(Mat image);

    // 1- Sets translation matrix
    void
    setTransMat(Mat_<double> &Matrix, Point2f Origin);

    // 2- Sets rotation matrix
    void
    setRotMat(Mat_<double> &Matrix, double teta);

    // 3- System 1 ------> System 2
    Point2f
    Sa2Sb(Mat_<double> Transition, Mat_<double> Rotation, Point2f I);

    // 4- Calculates the cartesian coordinates of a Point in any system using
       its r & teta.
    void
    xy(double r, double teta, Point2f &I);

    // 5- Set Scale
    void
    SetS(double scale);

    // 6- Set Model Height
    void
    SetH(double Model_H);

    // 7- Set Pose
    void
    SetPose(Mat_<double> &Pose);

    // 8- Set Limbs Size
    void
    SetLimbs();

    // 9- Set Joints x&y
    void
    SetJoints(/*double teta_WST_, double teta_NCK_, double teta_RSH_, double
      teta_LSH_, double teta_REB_,
                double teta_LEB_, double teta_RHP_, double teta_LHP_, double
      teta_RKN_, double teta_LKN_,
                int center_x_, int center_y_*/);

    // 10- Update Modle
    void
    UpdateModel(double teta_WST_, double teta_NCK_, double teta_RSH_, double
      teta_LSH_, double teta_REB_,
                double teta_LEB_, double teta_RHP_, double teta_LHP_, double teta_RKN_, double teta_LKN_,
                double Scale_, double Model_H_, int center_x_, int center_y_);

    // 11- Update Modle
    void
    UpdateModelPose(double Scale, double Model_H);

    // 12- Calculates the Cartesian coordinate of Joints in REF system.
    void
    Jnts2REF(vector<Point2f> &Jnts);

    // 13- Draw Model
    Mat
    DrwMDl_Skletal(vector<Point2f> Joints, Scalar Colour, int Tickness);

    // 13-1 Draws the Skeletal Model over image.
    void
    DrawSkeletalMoDel(vector<Point2f> Joints, Scalar Colour, int Tickness, Mat &image);

    // 13-2- Draws the Skeletal Model partialy over image.
    void
    DrawSkeletalMoDelPartialy(int level_, Scalar Colour, int Tickness, Mat &image);

    // 13-3 Draws the estimated limbs over image.
    void
    DrawEstimatedLimbs(Mat_<double> &Position_, int &level_, Scalar Colour, int Tickness, Mat &image);

    // 14- Extracting Points in a line (No= Number of points)
    vector<Point2f>
    Lin2Pnts(Point2f S, Point2f E, int Num);

    // 15- Extracting Points inside a rect.
    vector<Point2f>
    Rct2Pnts(vector<Point2f> Vertices, int lng, int lat);

    // 16- Extracting two vertices from a joint
    vector<Point2f>
    Jnt2Vers(Point2f S, Point2f E, double dl0, double dl1, double dl2, double dl3);

    // 16-1- Calculates two vertices of a line prependiculat on line J1J2.
    void
    Jnt2VertVers(Point2f &S, Point2f &E, double dl, Point2f &P);

    // 17- Creating Vertices for head.
    vector<Point2f>
    Hed2Vers(Point2f Head, double teta_head);

    // 18- Pictorial model vertices
    Pictorial
```

```
Picts2REF(vector<Point2f> Joints);

// 19- Draw Model
Mat
DrwMDl_Pictorial(Pictorial Pict_Model, Scalar Colour, int Tickness);

// 20- Extract points cloud from the pictorial model
vector<Point2f>
Pict2Cld(Pictorial pict_model, int lng, int lat);

// 21- This function set to "1" the elements related to points in PntsCld
    .
// It rounds the x,y for finding the element in matrix.
// So, the number of non-zero elements of matrix would be equal or less
    than the Vector of Points size.
Mat_<int>
Cld2Mat(vector<Point2f> PntsCld, int rows, int cols);

// 22- Converting a Pose vector to a Matrix.
// The non-zero elements of matrix show the points in Point2f cloud.
Mat_<int>
Pose2Mat(Mat_<double> Pose);

// 23- This function measures the dissimilarity between two Poses.
double
PosesDissimilarity(Mat_<double> Target, Mat_<double> Pose, int rows, int
    cols);

// 24-
double
compute_skew1(Mat img);

// 25-
void
deskew(Mat img, double angle);

// 26-
Point2f
rotatePnt(Point2f p, double teta);

// 27-
Point2f
Translation(Point2f p, int x0, int y0);

// 28- Calculates the H from the Joints coordinates
void
CalculateH(vector<Point2f> Jnts);

// 28-1 Sets the Limbs Size using the Joints.
void
SetLimbsByJoints(vector<Point2f> Jnts);

// 29- Calculates the Pose vector from the vector Joints.
void
Joints2Pose(vector<Point2f> Jnts, Mat_<double> &Pose);
```

```

// 29- 1
void
Joints2Pose1(vector<Point2f> Jnts, Mat_<double> &Pose);

// 30- Calculates the angle of a line passing two given points p1 & p2.
double
AngleOfLine(Point2f p1, Point2f p2);

// 31- Calculates the angle betwen two line cp1 & cp2.
double
Angle2Lines(Point2f c, Point2f p1, Point2f p2);

// 31-1
double
AngleOf2Lines(const Point2f &A, const Point2f &B, const Point2f &C, const
    Point2f &D);

// 32- // p1 & p2 are the ending points of line and p3 is the test point.
double
Pnt2LineDist(Point2f p1, Point2f p2, Point2f p3);

// 33- Calculates the two ending joints of a limb
void
CreateLineVertices(vector<Point2f> &Joints, vector<vector<Point2f> > &
  LineVertices);

// 34- Gets the limb
int
WhichLimb(vector<vector<Point2f> > &LineVertices, Point2f p, double &dist
  , string &Limb);

// 34-1
Scalar
WhatColour(string &Limb, int &Limb_idx);

// 34-1-1
Scalar
GetColour(int colour_idx);

// 34-2 Calculates the distance of an IP from a limb
void
Distance2Limb(vector<Point2f> &LimbVertices_, Point2f p, double &dist);

// 35- Applies transformation to a set of IPs
void
Transform_points(const Point2f &A, const Point2f &B, const Point2f &C,
  const Point2f &D,
         const vector<Point2f> &points1, vector<Point2f> &points2);

// 36- Renders a set of IPs for a pose hypothesis
void
IPs_rendering(vector<Point2f> &Pose_joints, vector<Point2f> &Pose_IPs,
         vector<Point2f> &Hypo_joints, vector<Point2f> &Hype_IPs);

  // 37- Labels the IPs to the limbs.
  void
  IPs2Limbs(const vector<Point2f> &IPs, vector<vector<Point2f> > &LineVertices,
          vector<vector<Point2f> > &LabeledIPs, vector<int> &IPsLabel);

  // 38- Gets the IPs_p & IPs_c of level.
  void
  GetIPs4Limb(vector<vector<Point2f> > &LabeledIPs_p_, vector<vector<Point2f> > &LabeledIPs_c_,
          vector<vector<double> >  &matchingDistance_, vector<Point2f> &IPs_p_, vector<Point2f> &IPs_c_,
          vector<double> &M_distance_, int &nMidIPs_, int level_);

  // 39- Creates the vertices of limb or limbs upt to the level.
  void
  CreateLimbVertices(Mat_<double> &Position_, int &level_, vector<vector<Point2f> > &LimbVertices_);

  // 39-1 Creates the vertices of limb or limbs upt to the level (for when we have Elbowroom).
  void
  CreateLimbVertices1(Mat_<double> &Position_, int &level_, vector<vector<Point2f> > &LimbVertices_);

  // 40- Creates Vertices for levels.
  void
  CreateVertices4level(vector<Point2f> &Pose_joints, vector<vector<Point2f> > &LimbVertices_p_, int level_);

  // 41- Renders the corresponding IPs_h for the hypothesised limb.
  void
  IPs_h_rendering4level(vector<vector<Point2f> > &LimbVertices_p_, vector<Point2f> &IPs_p_, vector<Point2f> &IPs_c_,
          vector<vector<Point2f> > &LimbVertices_h_, vector<Point2f> &IPs_h_, int &nMidIPs_, int level_);

  // 42- Applies the estimated Position by PSO for current level to the model.
  void
  ApplyEstimatedPosition(Mat_<double> Position_, int &level_);

  // 42-1
  void
  ApplyEstimatedPosition1(Mat_<double> Position_, int &level_);

  // 43- Gets the neighbouring level for the given level_
  int
  GetNeighbouringLevel(int &level);

};
```

# Appendix B

# Particle Swarm Optimisation Considerations

*The beauty you see in me is a reflection of you.*

VANNA BONTA.

## B.1 PSO versus Bayesian Filtering

As discussed earlier in Chapter 5, PSO uses particles to explore the search space. The phrase ”particle” causes the reader confusingly think that PSO is an implementation of the Bayesian filter, similar to the Particle Filter (PF). On this basis, one might think that particles model a probability distribution over the system states. Nevertheless, this is a misconception and the concept is completely different. PF uses the particles to estimate the system’s state probability distribution, while the particles in PSO are the potential solutions which explore the search space to find the optimum solution. Similarly, the PF particle’s weight is different from the cost function associated with the PSO particles. Each PSO particle is an individual in the search space, which has its own velocity, while there is no similar notation in PF.

The Annealed Particle Filter (APF) similarly explores the search space through several iterations (Deutscher and Reid, 2005). However, it is different from PSO substantially. PSO belongs to the swarm intelligence optimisation methods' family, where the particles have their own velocity and communicate with each other to search the solution's space at each time instant. On the other hand, APF is a particle filter improved by an additional optimisation step, where particles have no velocity and there is no communication between them.

## B.2 Convergence

Despite PSO looking very simple, it is non-trivial to analyse stochastic swarm intelligence optimisation methods. The convergence of PSO strongly depends on the cost function. Nonetheless, the research on that is ongoing. Poli (2009) analysed the behaviour of PSO in terms of the convergence under stagnation and introduced the PSO sampling distribution. Furthermore, the capability of PSO on specific problems also have been investigated by (Kobayashi et al., 2007; Perlin et al., 2008; Zhang et al., 2008).

# Appendix C

# Homography Estimation

*So many books, so little time.*

FRANK ZAPPA

Under homography, the transformation of the points in $3D$ is stated as:

$$X_2 = HX_1 \quad X_1, X_2 \in R^3 \tag{C.1}$$

Using the homogeneous coordinates in the image planes, this equation can be rewritten as:

$$\lambda_1 x_1 = X_1, \quad \lambda_2 x_2 = X_2, \rightarrow \lambda_2 x_2 = H\lambda_1 x_1 \tag{C.2}$$

By ignoring the universal scale ambiguity, we can state $x_2$ is approximately equal to $Hx_1$, i.e., $x_2 \sim Hx_1$. Starting from this equation, we can estimate $H$ as follow:

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{C.3}$$

In inhomogeneous coordinates we have $(x\prime_2 = \frac{x_2}{z_2} \quad and \quad y\prime_2 = \frac{y_2}{z_2})$. So

$$\begin{cases} x'_2 = \frac{H_{11}x_1+H_{12}y_1+H_{13}z_1}{H_{31}x_1+H_{32}y_1+H_{33}z_1} \\ y'_2 = \frac{H_{21}x_1+H_{22}y_1+H_{23}z_1}{H_{31}x_1+H_{32}y_1+H_{33}z_1} \end{cases} \tag{C.4}$$

After rearranging this equation with $z_1 = 1$ (without the loss of generality), we would have

$$\begin{cases} x'_2(H_{31}x_1+H_{32}y_1+H_{33}) = (H_{11}x_1+H_{12}y_1+H_{13}) \\ y'_2(H_{31}x_1+H_{32}y_1+H_{33}) = (H_{21}x_1+H_{22}y_1+H_{23}) \end{cases} \tag{C.5}$$

To solve for $H$, we need to solve for the above equation which can be written as follow:

$$\begin{cases} a_x^T h = 0 \\ a_y^T h = 0 \end{cases} \tag{C.6}$$

where

$$\begin{cases} h = (H_{11},H_{12},H_{13},H_{21},H_{22},H_{23},H_{31},H_{32},H_{33})^T \\ \quad a_x = (-x_1,-y_1,-1,0,0,0,x'_2x_1,x'_2y1,x'_2)^T \\ \quad a_y = (0,0,0,-x_1,-y_1,-1,y'_2x_1,y'_2y1,y'_2)^T \end{cases} \tag{C.7}$$

As mentioned in Chapter 5, to find the homography between two sets of points, we need to have at least 4 matched pairs of points. Given these IPs, we can form the following linear equation system to calculate the homography matrix $H$:

$$Ah = 0 \tag{C.8}$$

where

$$A = \begin{pmatrix} a_{x1}^T \\ a_{y1}^T \\ . \\ . \\ . \\ a_{xN}^T \\ a_{yN}^T \end{pmatrix} \tag{C.9}$$

Eq. C.9 can be solved using homogeneous linear least squares method. For the inhomogeneous linear least squares problem of $Ax = b$, we solve for $x$ using the pseudo-inverse or inverse of $A$. This does not work for the homogeneous form of our problem and so we solve it using Singular Value Decomposition (SVD) method (Golub and Reinsch, 1970). In this order, the SVD of $A$ is computed as follow:

$$A = U\Sigma V^T = \sum_{i=1}^{9} \sigma_i u_i v_i^T \tag{C.10}$$

In this way, we will have 9 singular values $\sigma_i$ sorted in descending order, from $\sigma_1$ to $\sigma_9$. According to the value of $\sigma_9$, the homography estimation falls into one of the following categories:

- if $\sigma_9 = 0$, the homography is exactly determined and the IPs are fitted exactly by the calculated homograpy.
- if $\sigma_9 \geq 0$, the homography is overdetermined and $\sigma_9$ represents the goodness of the fit.
- if $\sigma_9 < 0$, the homography is under-determined.

Using the SVD we take the singular vector (a column from V ) corresponding to the smallest singular value $\sigma_9$. This vector is the solution we are seeking for $h$ and $H$.

As stated in Chapter 5, for two set of matched IPs with a fraction of outliers, the homography estimation using RANSAC and LMeDS is performed iteratively with different group of IPs to find the best perspective transformation $H$ which minimizes the following back-projection error[1]:

$$\sum_i (x'_i - \frac{H_{11}x_1 + H_{12}y_1 + H_{13}}{H_{31}x_1 + H_{32}y_1 + H_{33}})^2 + (y'_i - \frac{H_{21}x_1 + H_{22}y_1 + H_{23}}{H_{31}x_1 + H_{32}y_1 + H_{33}})^2 \tag{C.11}$$

---

[1] findHomography function, OpenCV: `http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#MatfindHomography(InputArraysrcPoints,InputArraydstPoints,intmethod,doubleransacReprojThreshold,OutputArraymask)`

# Appendix D

# Implementation and Programming

*A person starts to live when he can live outside himself.*

ALBERT EINSTEIN

To evaluate the proposed approaches, they have to be implemented using a suitable programming language. MATLAB[1] is a numerical computing programming language, which allows easily implementation of algorithms as well as interfacing with programs written in other languages, including C, C++, Java, Fortran and Python. Nonetheless, it is not efficient in terms of speed and porting into the embedded systems.

In contrast, the high-level programming languages such as C, C++, Java, Python are not easy and user-friendly languages. Nevertheless, they are very fast, efficient, and powerful languages, which permit the user to access the hardware on a computer or even its own cards in slots. They do not provide systematic checks as they assume that what user wants to do is what he/she says to do. Errors in these languages can be very subtle, and can be hard to find. The real decision about which language is the best choice for implementation is up to the programmer and what he/she plans to do.

---

[1] Matlab (matrix laboratory): `http://www.mathworks.co.uk/products/matlab/`

There are several efficient and real-time libraries for implementation of image processing and computer vision algorithms[2]. Among them, OpenCV[3] (Open Source Computer Vision) is the most popular and mature one, which is free to use under the BSD licence and supports more than 500[4] algorithms together with good documentation[5] and sample codes. It has been developed by Intel Russia research center and is now supported by Willow Garage[6] and Itseez[7]. OpenCV was launched officially in 1999 for the C language with the C++ version issued in October 2009[8]. These interfaces make OpenCV portable to some specific platforms such as digital signal processors[9]. Wrappers for languages such as C#, Ch[10], Python[11], Ruby and Java (using JavaCV[12]) have been developed to encourage adoption by a wider audience. OpenCV runs on Windows, Android[13], Maemo[14], FreeBSD, OpenBSD, iOS[15], Linux and Mac OS.

The Point Cloud Library[16] (PCL) is another open-source, standalone, and large scale library suitable for processing of 2D/3D images and point clouds. It has been written in C++ and released under the BSD license, and is thus free for commercial and research use. The algorithms which the library supports are generally divided into: feature estimation; surface reconstruction; registration; model fitting; and segmentation. It is a good tool for visualization of the point clouds.

In this thesis, we have used a combination of three of the above mentioned languages alternatively. The C++ version of OpenCV in Eclipse[17] IDE under the Ubuntu-Linux has been

---

[2] A list of other open source computer vision codes and libraries: http://www.computervisiononline.com/software

[3] OpenCV: http://opencv.org/

[4] Computer Vision Libraries: http://digi.physic.ut.ee/mw/index.php/Computer_vision_libraries

[5] Documentation of OpenCV: http://docs.opencv.org/

[6] Willow Garage: https://www.willowgarage.com/

[7] itseez, Vision that works: http://itseez.com/

[8] http://opencv.willowgarage.com/wiki/OpenCV%20Change%20Logs

[9] OpenCV C interface: http://opencv.willowgarage.com/documentation/c/index.html.

[10] Ch OpenCV: http://www.softintegration.com/products/thirdparty/opencv/.

[11] Python Interface: http://opencv.willowgarage.com/documentation/python/index.html.

[12] JavaCV: http://code.google.com/p/javacv/.

[13] Android port: http://opencv.willowgarage.com/wiki/AndroidExperimental.

[14] Maemo port: https://garage.maemo.org/projects/opencv.

[15] IPhone port: http://www.eosgarden.com/en/opensource/opencv-ios/overview.

[16] PCL: http://pointclouds.org/

[17] Eclipse: https://www.eclipse.org/

used mainly for implementation of the ideas and algorithms. PCL also has been used sometimes, mostly for visualization purposes. On the other hand, we have used Matlab occasionally to test some ideas, owing to the ease of performing matrix manipulations in Matlab, and generating graphs and diagrams. We have employed Boost[18] open-source C++ library too for doing some mathematical operations.

Owing to our partnership with Movidius, A few months have been spentas an intern in the company to examine how the proposed algorithms can be ported into the Myriad1 and Myriad2 platforms. These have a c-based library and compiler which contains a group of the OpenCV functions and algorithms. Since this platform is a multi-core power efficient embedded system, dealing with this special programming environment for porting the algorithms has been another implementation aspect of our work.

---

[18] Boost: `http://www.boost.org/`