UNIVERSITY OF CALIFORNIA
RIVERSIDE

Individual 3D Face Modeling and Recognition in a Video Network

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Hoang Thanh Nguyen

September 2012

Dissertation Committee:

Dr. Bir Bhanu, Chairperson
Dr. Chinya Ravishankar
Dr. Amit Roy-Chowdhury
Dr. Gerardo Beni

The Dissertation of Hoang Thanh Nguyen is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am grateful to my advisor, Bir Bhanu, whose guidance and encouragement has been pivotal in helping me achieve my goals. I would like to thank my committee members for their help, insights, and for providing an endless source of ideas on which to pursue and ponder.

To my fellow labmates, all 40+ who have come and gone over the past 5 years, some from the lab and some forever, I thank you for the experiences we shared, both professionally in the lab and socially on all the trips we embarked on together. From quarterly potlucks to hiking in Yosemite, from paintball trips to piloting planes and helicopters. You have made graduate school a wonderful experience. I extend my thanks to Kevin Bash and all 60+ of my training partners at Riverside Aikido, to the members of the UCR Taekwondo Club, to the teachers of the UCR Swing Club, and to the Outdoor Excursion and personal trainers at the UCR Student Recreation Center. You helped make every day an adventure.

I would like to thank Victor Glen Hill and Giovanni Laviste Denina for their encouragement and patience while I finished my studies. I would also like to thank the Computer Science and Electrical Engineering staff for silently (but tirelessly) working behind the scenes to keep everything up and running, and to the Bourns College of Engineering Dean's Office for providing me the opportunity to work on new projects. Additionally, I would like to extend my thanks to the custodial staff, Sunday and Andy, for being there for us every day.

This work was supported in part by NSF grants 0551741, 0622176, 0727129, and 0905671 and ONR grants N00014-09-C-0388, N00014-07-C-0311, and N00014-07-1-0931.

To my family:

I dedicate my dissertation to you. To my father, Hai Thanh Nguyen, who has provided and cared for me through all my endeavors. To my mother, Tuyet Ngoc Nguyen, who has always cared for and worried about my well-being the many nights I would not return from lab. To my uncle, Tho Si Vuong, for sacrificing so much for us and always showing us generosity. To my sister, Dana Marie Morita and her husband, Albert Shigeki Morita, for taking care of me and welcoming me into their home with many a nourishing meal. To my brother, Son Thanh Nguyen, who has supported me for over a decade and motivated me to improve my health. To my sister, Lan Thanh Nguyen, for inspiring me to pursue my dreams. To my sister, Ngoc Thanh Nguyen, who has been there for me whenever I simply asked her to. To my niece, Kathlyn Sakura Morita, for giving me a reason to never give up.

I love you all.

ABSTRACT OF THE DISSERTATION

Individual 3D Face Modeling and Recognition in a Video Network

by

Hoang Thanh Nguyen

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, September 2012
Dr. Bir Bhanu, Chairperson

Given an uncalibrated network of video cameras, we are tasked with the problem of building a 3D model of every person's face as they move within the network. The process of doing so requires overcoming many challenges including person detection, tracking, cross-camera correspondence (how to determine if a person in one camera is the same person in another), and the final 3D model reconstruction from multiple views in real-time or at near real-time speeds (efficient modeling and data fusion from multiple hardware sources). Towards this goal, a wireless camera network was designed and built from the ground up, a new tracking algorithm and a cross-camera human signature method was developed, and face modeling using multiple cameras in a real-world setting was performed.

Contents

List of Figures **x**

List of Tables **xv**

1 Introduction **1**

2 Design and Optimization of the VideoWeb Wireless Camera Network **3**
2.1 Introduction 4
2.2 Related Work and Contributions 6
2.3 Building a Camera Network 7
2.3.1 Choosing the Type of Network 7
2.3.2 Choosing the Right Camera 9
2.3.3 Choosing and Configuring the Network Hardware 19
2.3.4 Building the Server Hardware 21
2.3.5 Software System 26
2.3.5.1 Sample program - head tracking 26
2.4 Experiments for Performance Characterization and Optimization of the Video Network 27
2.4.1 Measurement software 27
2.4.2 Optimizing Camera Configuration 28
2.4.3 Multi-objective Optimization Using Pareto Efficiency 31
2.4.3.1 Inferiority and Non-Inferiority 31
2.4.3.2 Data Collection 32
2.4.4 Evaluation Results 33
2.4.5 Task-based Optimization 34
2.5 Conclusions 36

3 Real-Time Pedestrian Tracking with Swarm Intelligence **38**
3.1 Abstract 38
3.2 Introduction 39
3.3 Related Work and Contributions 41
3.3.1 Tracking Approaches in the Literature 42
3.3.1.1 Optical flow 42
3.3.1.2 Template/shape-based approaches 43
3.3.1.3 Behavior modeling 44
3.3.1.4 Kernel-based approaches 45

3.3.1.5 Multi-camera approaches . . . 46
3.3.2 Swarm Intelligence . . . 46
3.3.3 Contributions . . . 47
3.4 Technical Approach . . . 49
3.4.1 Pedestrian Detection . . . 49
3.4.2 Appearance Signature . . . 50
3.4.2.1 Body segmentation . . . 50
3.4.2.2 Occlusion layers . . . 51
3.4.3 Tracking Using Swarm Intelligence . . . 52
3.4.3.1 Bacterial Foraging Optimization . . . 55
3.4.3.2 Particle Swarm Optimization . . . 57
3.4.3.3 Fitness function . . . 57
3.5 Experimental Results . . . 59
3.5.1 Datasets . . . 59
3.5.2 Effect of Parameters and their Selection . . . 60
3.5.3 Performance Metrics . . . 61
3.5.4 Tracking Results . . . 62
3.5.4.1 CAVIAR Dataset . . . 62
3.5.4.2 PETS2010 Dataset . . . 62
3.5.5 Discussion of Results . . . 62
3.6 Conclusions . . . 63

4 Zombie Survival Optimization: A Search Optimization Framework Inspired By Zombie Foraging Behavior 71
4.1 Introduction . . . 72
4.2 Related work . . . 74
4.3 Technical approach . . . 75
4.3.1 Zombie exploration mode . . . 75
4.3.2 Zombie hunter mode . . . 76
4.3.3 Human exploitation mode . . . 76
4.4 Experimental results . . . 77
4.4.1 Dataset . . . 77
4.4.2 Metrics and effect of parameters . . . 77
4.4.3 Parameters . . . 77
4.4.4 Results . . . 78
4.5 Conclusions . . . 78

5 Multi-camera Appearance Signatures 87
5.1 Introduction . . . 87
5.2 Technical Approach and Experiments . . . 88
5.3 Results . . . 89
5.4 Conclusions . . . 92

6 Individual 3D Face Modeling 93
6.1 Introduction . . . 93
6.2 Related Work . . . 94
6.3 Image Acquisition and Face Detection . . . 95
6.4 Face Alignment and Pose Estimation . . . 95
6.5 Calculating 3D Shape Parameters . . . 99

6.5.1 The 3D generic model 99
6.5.2 Fitting the 3D model 99
6.6 Extracting and Fusing Texture 102
6.7 Experimental Results 106
6.7.1 Dataset 106
6.7.2 Results 106
6.8 Conclusions 107

7 Conclusions **108**
7.1 Building a Video Network 108
7.2 Tracking in a Video Network 108
7.3 Multi-camera Recognition in a Video Network 109
7.4 3D Model Reconstruction in a Video Network 109

Bibliography **110**

List of Figures

2.1 Overall architecture. Top down: a single interface is used for direct control of any server/camera and high-level processing (e.g., user-defined face recognition). The server connects to a switch which hosts a database and joins two sets of servers: a series of mid-level (e.g., feature extraction) and low-level processors (e.g., detecting moving objects). The switch connects to routers which communicate with wireless bridges connected to the IP cameras. 5
2.2 Stream corruption caused by network congestion may manifest in different ways depending on the video format. Left: corrupted Motion JPEG stream due to partial data, right: corrupted MPEG-4 stream due to partial data. 12
2.3 37 camera locations cover the 14,300 square foot second floor of Engineering Building Unit II at the University of California, Riverside. Locations were manually selected and evaluated to ensure that usable fields of view were available for every square inch of the building from at least two viewpoints. 16
2.4 Installation of the cameras: a) Axis 214 PTZ in an outdoor-rated enclosure; only one of these were installed (high above a large open courtyard) due to size and appearance), b) wireless bridges installed in the ceilings to make the IP cameras wireless, c) flush-mounted Axis 215 PTZ cameras in sealed indoor enclosures. 18
2.5 Processing hardware and mobile cameras: a) control interface and monitors in laboratory, b) 32-server processing backend connected to the interface, c) cameras mounted on 3 robots add mobility to the network. . 24

2.6 Measurement comparison matrices for 8 cameras. While cameras may exhibit variable performance even when using the same configurations, some configurations may be inherently better than others and exhibit similar performance across the network. To discover these configurations, 100 trials are performed on each camera under a variety of parameter configurations (i.e., resolution and compression) and each recorded measurement is compared for Pareto efficiency against the other 99 trials. This results in a symmetric matrix where vertical and horizontal axes indicate the measurements M_i and M_j, respectively (i.e., the top-leftmost square in each matrix indicates the relationship of M_1 against M_{100}). Red indicates that a particular M_i is inferior to a particular M_j, green indicates superiority, and a solid horizontal yellow line denotes rows which are completely Pareto-efficient (i.e., either superior or non-inferior against all other 99 trials). 30
2.7 Probability of configuration membership in any given camera's Pareto set. 33
2.8 The top 8 dominating camera configurations as chosen by 37 cameras. Graphs are ordered by the percentage of cameras in which the particular configuration was Pareto-efficient and all metrics are normalized to 1.0 across all cameras. Clockwise from the top: resolution ranges from 176×120 to 704×480 (higher is better), JPEG compression settings range from 0 to 100 (lower is better, so inverse is shown), and frame rates range from 0 to 30 FPS (higher is better). For measuring the "smoothness" of outputted video, the standard deviation of the frame rate (recorded at 1-second intervals) and maximum lag time between any two sequential frames is recorded (lower is better, so inverse is shown). 35

3.1 System diagram of the tracking system. Parts-based appearance signatures are used to drive a swarm intelligence search algorithm for pedestrian tracking. 40
3.2 Pedestrian ROIs are segmented into multiple partitions based on statistical human body ratios to create more robust appearance signatures. . 44
3.3 Occlusion compensation using occlusion layers. Left: ROIs are sorted by the y-value of their bottom edge and placed into layers in descending order. Right: the masking of lower layers by higher layers helps minimize the influence of occluding pedestrians when computing signatures. . . . 47
3.4 How pedestrians appear to a tracker. (a) Detected pedestrians, (b) fitness space for the pedestrian using color similarity in the CIE LAB color space (brighter = higher fitness), (c) fitness space after applying the foreground mask to the input image to reduce background noise. Note that the foreground mask is not as effective in the last PETS2010 video due to sudden lighting changes in the scene. 48
3.5 Behavior of a single Bacterial Foraging Optimization swarm searching for a pedestrian. (a) Random initialization, (b) gradient-hill climbing in random directions, (c) death/rebirth of agents with poor fitness to location of agents with best fitness, (d) target location based on consensus of the best agents. 52
3.6 Performance contributions of each modification of m-BFO over traditional Bacterial Foraging Optimization (BFO) on a sample dataset [68] shown with minimum, average, and max performance measurements. 55

3.7 Sample frames from all 26 mall corridor videos of the CAVIAR dataset [32]. All videos have a resolution of 384 × 288 pixels and consist of a total of 35,913 frames and 131,288 pedestrian ROIs. The objective is to track all pedestrians entering and exiting the field of view under different shopping scenarios. 64
3.8 Sample frames of the "S2.L3" crowd scene from the PETS2010 dataset [30] (768 × 576 pixels, 240 frames, and 470 pedestrian ROIs). The objective is the track the two pedestrians labeled A and B as they join in walking with a large incoming crowd. 64
3.9 Segmenting the body into parts improves tracking performance as opposed to using traditional single blob-based histograms. Results are shown averaged with std. bars over all 26 corridor videos of the CAVIAR dataset, 30 runs for each video. 65
3.10 Average recognition accuracy of color histograms using boosted C4.5 trees on the Viewpoint Invariant Pedestrian Recognition (VIPeR) dataset [37]. 65
3.11 Average recognition accuracy of color histograms by body part on the Viewpoint Invariant Pedestrian Recognition (VIPeR) dataset [37]. . . . 66
3.12 PSO, BFO, Particle Filter, and whole image detection-based tracking perform on par with exhaustive grid search. Of these trackers, BFO performs the fastest (see Figure 3.13). Results are shown averaged with std. bars over all 26 corridor videos of the CAVIAR dataset, 30 runs for each video. 66
3.13 The entire system can be run in real-time on modest hardware (Intel Xeon E5345 2.33GHz). BFO performs the fastest amongst the trackers which perform on par with exhaustive search. Run times are averaged across all 26 corridor videos of the CAVIAR dataset and averaged over 30 runs per video. 68
3.14 Even with multi-threaded tracking of each pedestrian, the number of pedestrians in the scene (over 40) saturates all available CPU resources (numbers are averaged over 30 runs on an Intel Xeon E5345 2.33GHz quad-core CPU). 69
3.15 The swarm intelligence approaches perform slightly better than the same Particle Filter used on the CAVIAR videos, even without segmentation. Results are averaged over 30 runs on the video in Figure 3.8. 70
3.16 Comparison of tracking accuracy for each category of video of the CAVIAR dataset (see Table 3.2). Results are averaged over 30 runs for each video. 70

4.1 Agents automatically switch between 3 modes: random walking (search space exploration), survival mode (search space exploitation), and hunter mode (swarming). 72
4.2 Four of the 1,843 fitness space images generated from the dataset using HSV signatures initialized on Viola-Jones pedestrian detections. Resolution is 384×288 with higher (red) locations representing high fitness and lower locations (blue) representing low fitness. 73
4.3 1-2) Initialize and explore. 3) One or more agents turn into humans (green) due to high antidote concentrations. 4) Zombies pursue humans. 5a-5b) Humans are either defeated or survive. 79
4.4 Samples frames from the CAVIAR dataset [32]. 81

4.5 Effects of parameters N (number of agents) and G (number of generations). Adjusting these parameters can allow the number of total evaluations to remain constant (10,000 fitness evaluations in this case), while balancing the size of the swarm. Setting an even balance between the two achieves the best average performance. 82
4.6 Effects of step size parameter S. This search space-dependent parameter may vary depending on how minute or spread apart fitness value changes may be in the underlying search space. In this case, the 30 pixel step size was suitable for the 384×288-pixel images. 83
4.7 Effects of the zombie to human fitness threshold parameter A on best average fitness. Setting this parameter low results in earlier exploitation search while setting it higher favors exploration of the search space. A balanced threshold of 0.50 achieved the best performance, suggesting that exploration should be evenly balanced with exploitation. 84
4.8 Effects of variance parameter V on best average fitness. The important thing to note is that adding randomness to the agents in their direction is more beneficial than moving in a straight line, e.g., the 0 degree variance. 85
4.9 Experimental results comparing Zombie Survival Optimization with Particle Swarm Optimization, Bacterial Foraging Optimization, and Random Search. On the CAVIAR search spaces, ZSO is able to locate higher average fitness values over time. 86

5.1 Sample dataset consisting of 5 cameras (2 indoor and 3 outdoor) were used to record 5 pedestrians. The dataset consists of 4000+ pedestrian ROIs for use in testing of the discriminative properties of the appearance signature representations. 88
5.2 Recognition accuracy performance of all color and texture features. Color features far out-perform texture features when used alone. 89
5.3 Recognition accuracy performance of all color features. HSV provides the best performance using 4 bins per HSV-component while an increase to 32 bins per component favors YUV YIQ. 90
5.4 Recognition accuracy performance by body parts. Using all body parts as partitions provides improved accuracy over using all body parts as a single partition. 91
5.5 Online recognition accuracy using simple nearest neighbor comparison of histograms. 91

6.1 Overview of the proposed system. Once a face has been detected, face alignment is performed on it to recover landmark facial points. From these points, the pose is estimated and then used to morph a 3D model to fit the person's face. 94
6.2 Automatic landmark detection done in real-time using Constrained Local Models (CLMs). 96
6.3 Given a set of landmark facial points, the pose of the head is estimated by minimizing the mean square error of the landmark points with the projection of the corresponding points on the 3D generic model. Red points represent the landmark facial points of the image while blue points represent the projected 3D model points. 97

6.4 The Basel Face Model (BFM) [77]. Left: mean face of 200 subjects. Right: same face morphed using the "age" set of parameters. 98
6.5 Sample camera environment consisting of three 640×480 pixel resolution videos from three different views of the same room. 98
6.6 13 face images extracted from the 3 videos. The faces range from 20px to 90px in height. 100
6.7 The morphed 3D model improves as more frames are added to it. Landmark points in these faces are manually located. 103
6.8 Comparison of morphed 3D model with groundtruth. Left: Morphed 3D model using the proposed approach. Center: Morphed model with fused texture. Right: Groundtruth 3D face scan using a Minolta laser scanner. 104
6.9 Bicubic interpolation is used to normalize all images to the same resolution before fusing. Left: Without interpolation. Right: With interpolation. 104
6.10 Automatically reconstructed 3D faces from video. Facial landmarks from the MPEG-4 Facial Definition Parameters [79] are detected using Constrained Local Models [86] and used to morph the Basel Face Model [77] to fit the person's face. 105
6.11 Using a gender detector to add shape priors to the reconstructed models. 106

List of Tables

2.1 Camera behavior can vary radically across vendors and models. Under congested network conditions for example, cameras may permanently drop frames or attempt to resend missed frames at the expense of live data. The Panasonic camera in this case output "smoother" video (fewer frame drops between two successive frames) under heavy network congestion (until its onboard cache is exhausted) at the cost of delays in upwards of 6 seconds. 12

3.1 Summary of major approaches to the pedestrian tracking problem. . . . 41
3.2 Statistics and categories for each video of the CAVIAR dataset. 67

4.1 Definitions of symbols used in the Zombie Survival Optimization algorithm. 74

Chapter 1

Introduction

If you were asked to build a 3D model of every person who walked through a building, what would it require? Approaching such a problem requires solving a number of significant computer vision problems, some of which are still unsolved today. The challenges involved include designing a sensor system capable of monitoring the building, detecting and tracking individuals in each sensor, fusing data from separate sensors to recognize whether individuals in one sensor are the same as those in another, and finally using the sensor data to generate the 3D models.

To approach the problem of building a sensor network, we require a sensor type and a location. The location we selected is the Winston Chung Hall engineering building at the University of California, Riverside. In this building, we have designed and built from the ground up a network consisting of 37 outdoor ceiling-mounted pan/tilt/zoom video cameras, 16 indoor ceiling-mounted fixed cameras, and 3 camera-equipped mobile robots. Chapter 2 goes into detail the design and optimization of this network.

Once the video network has been built, it is necessary to identify and detect our targets of interest, namely pedestrians. Using inspiration from biological systems,

we use swarm intelligence algorithms to perform tracking of individuals in the network. Chapter 3 details in depth the algorithms we used and how we improved on them. In addition, we developed our own swarm intelligence algorithm inspired by the foraging behavior of zombies in order to perform tracking. Chapter 4 details the development and performance of this new Zombie Survival Optimization algorithm.

In order to fuse data after single-camera tracking has been performed, we devised a part-based appearance signature in order to allow us to match individuals across cameras in the network. Chapter 5 details the statistically-based partitioning scheme of the human body and optimization of the signature representation.

Finally, face alignment is performed on the faces in the video and a 2D model is fitted to the pedestrian's face. Fusing the localized landmark points of each from both single camera as well as from multiple camera views, the landmark data is used to morph a 3D model to each individual's face. Chapter 6 details the process and results and Chapter 7 offers final closing remarks.

Chapter 2

Design and Optimization of the VideoWeb Wireless Camera Network

Abstract

Sensor networks have been a very active area of research in recent years. However, most of the sensors used in the development of these networks have been local and non-imaging sensors such as acoustics, seismic, vibration, temperature, humidity, etc. The emerging development of video sensor networks poses its own set of unique challenges, including high bandwidth and low latency requirements for real-time processing and control. This article presents a systematic approach by detailing the design, implementation, and evaluation of a large-scale wireless camera network, suitable for a variety of practical real-time applications. We take into consideration issues related to hardware, software, control, architecture, network connectivity, performance evaluation,

and data processing strategies for the network. We also perform multi-objective optimization on settings such as video resolution and compression quality to provide insight into the performance trade-offs when configuring such a network and present lessons learned in the building and daily usage of the network.

2.1 Introduction

We describe the design and development of a new laboratory called VideoWeb to facilitate research in processing and understanding video in a wireless environment. While research into large-scale sensor networks has been carried out for various applications, the idea of massive video sensor networks consisting of cameras connected over a wireless network is largely new and relatively unexplored. The VideoWeb laboratory entails constructing a robust network architecture for a large number of components, including cameras, wireless routers and bridges, and video processing servers. Hardware and equipment selection needs to take into account a number of factors, including durability, performance, and cost. In addition, VideoWeb requires a number of software applications including those for data recording, video analysis, camera control, event recognition, anomaly detection, and an integrated user interface.

Challenges for the design of VideoWeb include creating a wireless network robust enough to simultaneously support dozens of high-bandwidth video cameras at their peak performance, providing power and connectivity to cameras, building a server farm capable of processing all the streaming data in real-time, implementing a low-latency control structure for camera and server control, and designing algorithms capable of real-time processing of video data.

The article is organized as follows: In Section 2.2 we cover related work and

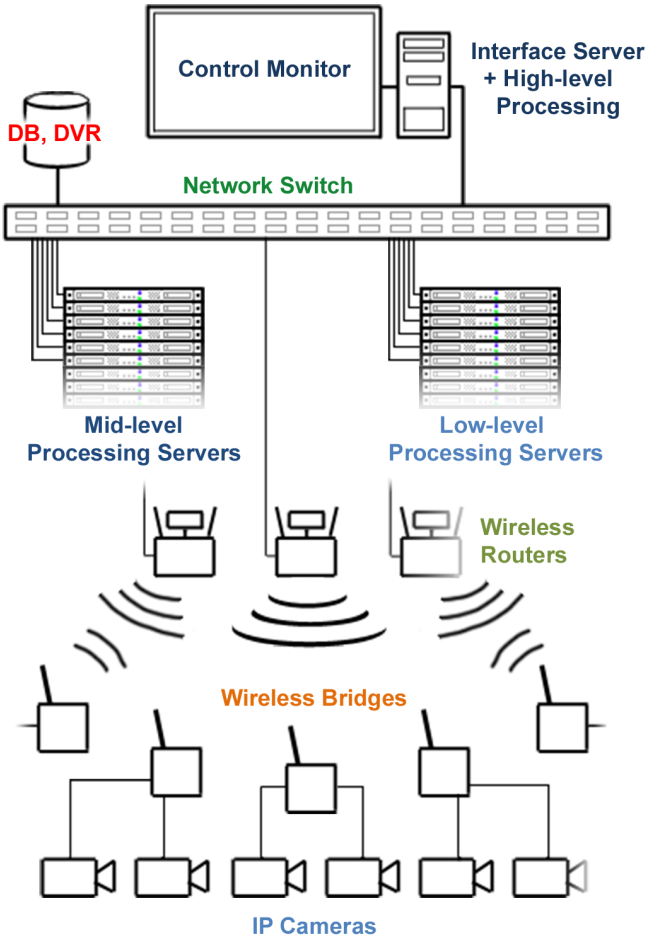


Figure 2.1: Overall architecture. Top down: a single interface is used for direct control of any server/camera and high-level processing (e.g., user-defined face recognition). The server connects to a switch which hosts a database and joins two sets of servers: a series of mid-level (e.g., feature extraction) and low-level processors (e.g., detecting moving objects). The switch connects to routers which communicate with wireless bridges connected to the IP cameras.

contributions of this article. Section 2.3 discusses the requirements and specifications used in designing the system and discusses the technical challenges and solutions for actual implementation. Section 2.4 delves into the performance metrics used to evaluate the system. Section 2.5 concludes with closing comments and lessons.

2.2 Related Work and Contributions

Many wireless camera platforms have been proposed ([1,16,75]) and emerging research in the design of wireless camera networks includes those with customized camera hardware nodes (e.g., CITRIC [15], eCAM [74]) including iMote2 and WiCa-based networks ([48,93]), as well as networks with carefully-calibrated cameras [81].

This article makes the following contributions:

1. We expand upon [70] by exhaustively detailing the design considerations made in building the VideoWeb wireless network in order to provide a general guideline for those looking to build their own network. We also discuss lessons learned from building and using the VideoWeb network so that others may benefit from our experience.

2. We make the case for IP cameras and server-side processing by designing and implementing a system utilizing network cameras running on a software-reconfigurable server and network architecture. While we use conventional IP cameras without on-board camera processing, the configuration of the server-side processing is user-configurable and allows on-the-fly changes such as going from tiered-processing (e.g., low-level processing servers do object detection and send silhouettes to mid-level processors which generate object signatures and broadcast to high-level servers) to 1-to-1 camera-to-server processing which simulates the behavior of on-camera processing networks.

3. We describe performance metrics on which to evaluate a video network's performance and show how multi-objective optimization can be used in order to discover Pareto-efficient settings for configuring the network.

2.3 Building a Camera Network

2.3.1 Choosing the Type of Network

There are many types of camera networks (e.g., wired vs. wireless, multi-hop wireless, distributed vs. central processing), but the most important factor in deciding what kind of network to build is determining the primary application. For instance, if a network's primary concern is surveillance (where reliability or maintaining uptime may be paramount), a hard-wired network may be the only way to satisfy said requirements. A wireless network, on the other hand, provides more freedom and allows cameras to go where hard-wired cameras cannot (restricted only by power source).

Our requirements and implementation

The VideoWeb network consists of a heterogeneous mixture of over 50 wireless pan/tilt/zoom (PTZ) network cameras, 3 mobile robots equipped with cameras (see Figure 2.5c), and a 128-core server rack for data processing. The network is designed to be a flexible general-purpose camera network for use as a research testbed for applications such as multi-camera tracking, scene analysis, and 3D reconstruction, as well as for research in improving robustness of wireless camera systems. Our implementation utilizes wireless cameras in order to take advantage of the flexibility in camera placement and cost savings afforded by not having to run network cable through the walls and ceilings to connect each of the cameras.

The complete architecture of the VideoWeb network (Figure 2.1) is comprised of a camera component, a wireless component, an application server component (e.g., database servers, digital video recording servers), and a processing component comprised of 3 levels: a set of servers which process camera feeds at a low level (e.g., human

detection, per-camera tracking), a set of servers which use this information for mid-level processing (e.g., feature extraction, multi-camera tracking), and a master interface server which uses this data for high-level processing and user control (e.g., task assignment, scene analysis, face recognition). The high-level server is also used as an interface for the network. Using a central switch to connect the two levels of processing servers allows data to move flexibly across servers to minimize network latency.

The server architecture is designed as a 3-level tree hierarchy: a master high-level interface server communicates with a set of mid-level processing servers, which in turn process data received from a number of low-level servers. A server architecture physically connected in this fashion would entail cameras forwarding data to one set of servers which forward low-level data to another set of servers, and once more to the high-level server. To minimize network overhead, we use a central network switch to connect the servers (as opposed to physically tiering the servers with direct connections) and implement the server hierarchy in the network's DNS configuration and in the communication strategy of our software.

An interface server is employed to allow users to view live or processed data from the cameras and to manually assign processing tasks (such as running a particular algorithm on some arbitrary number of cameras) from a central location (for VideoWeb, this location is back in the laboratory away from the noisy server room). For the wireless component of the network, cameras and servers are bridged through a single-hop wireless network using wireless routers connected to the servers to communicate with wireless bridges located throughout the building which connect to the cameras.

The following sections detail the design considerations made in building the network.

2.3.2 Choosing the Right Camera

Choosing the wrong camera can be a costly mistake when building a large video network. When selecting a camera, a number of factors should be taken into consideration. Besides cost, these may include:

- **Wired vs. Wireless cameras.** Deciding between a wired or wireless camera is often a trade off between whether or not speed and reliability can be sacrificed in order to gain flexibility and freedom in placement. Cameras which connect to a processing location (central or distributed server) with dedicated wire connections (e.g., Ethernet, audio/video cables) excel in providing improved speed and reliability. This comes at the cost of restricting installation locations to those which can be reached via physical cables and installation may prove to be very labor-intensive, expensive, or simply unfeasible. Wireless cameras on the other hand allow greater freedom in placement as well as offering the opportunity of mobility (in the case of non-stationary cameras, e.g., robots, field sensors), but may sacrifice speed, reliability, and/or security. Cameras with built-in wireless are essentially stuck with the installed protocol (though 802.11n is also backwards-compatible with 802.11g). Since the IEEE 802.11n standard supercedes 802.11g, this tends to make 802.11g-dedicated cameras feel outdated (especially if streaming requirements later exceed the bandwidth of the protocol or find that frame rates suffer from congestion and the only way to improve the situation then is by installing larger antennas, routinely changing wireless channels, and/or installing wireless repeaters). Cameras with built-in 802.11n wireless are preferred over 802.11g in almost all cases due to the increase in bandwidth, range, and potentially a less-crowded frequency range (though this may change with time). It is worth noting

that wired cameras which lack built-in wireless transmitters can easily be made wireless cameras via wireless bridges or adapters. This may be a better choice for long-lifespan camera networks which may need to be concerned with forward compatibility, for instance, as it avoids the network from being locked into any single standard. How easy it is to make this modification is affected by whether the cameras are digital or analog.

- **IP vs. Analog CCTV.** Digital vs. analog in the context of video cameras is often an issue of convenience. Traditional analog closed-circuit TV (CCTV) systems are often simpler and more cost-efficient, but search and retrieval of data is cumbersome and any applications beyond surveillance and monitoring may be awkward or require dedicated systems for each application. IP systems, on the other hand, can be more costly and/or complex, but output digital streams easily processed on computers and can even be accessed anywhere in the world simply by putting them on an Internet-accessible connection. If the video streams will be subject to constant or routine processing, analysis, or retrieval, IP cameras offer greater convenience and all the benefits of cheap digital storage, but may require additional network and software training for those only familiar with traditional CCTV systems.

- **Single-hop vs. Multi-hop wireless.** If wireless cameras are to be used, there are two primary ways they can reach their processing/storage destination: via a single-hop connection (cameras connect directly to wireless router/receivers) or via multi-hop connections (cameras connect to other cameras and pass on data before reaching the router/receiver). Multi-hop networks impose additional complexity and hardware as well as increased latency, but gain flexibility and wireless

coverage by essentially turning every camera into a repeater node; these are more-suited for cameras with on-board processing capabilities. Single-hop networks are recommended if it is viable (i.e., network routers can be installed in locations in which all cameras can reach) for purposes of lower latency and reduced hardware requirements.

- **External vs. On-camera processing.** Whether or not to perform processing on-camera or deferring processing to external computers/systems is impacted by camera capability/programmability and network latency and bandwidth. For instance, a multi-hop network may be too slow to permit active tracking if video needs to first be passed through several sensors before reaching a processor, whose control commands then need to be relayed across several more sensors before the camera ever receives the command to "pan left". Outside of basic scripting capabilities, most commercial cameras do not offer the flexibility or processing power to achieve processing tasks more complicated than basic motion detection or tracking. This issue often prompts network builders to develop custom programmable camera hardware for use in their systems [15,48,74,93]. On-camera processing can also reduce bandwidth consumption of the network (e.g, transmitting only areas of interest as opposed to full-frame video), while external processing allows a greater range of control and processing power.

- **Pan/Tilt/Zoom (PTZ) vs. Static cameras.** As the name implies, PTZ cameras offer active panning, tilting, and/or zooming capabilities whereas static cameras retain a permanent fixed field of view and orientation. PTZ cameras have the advantage of being able to cover larger areas (as a whole) and can zoom in or

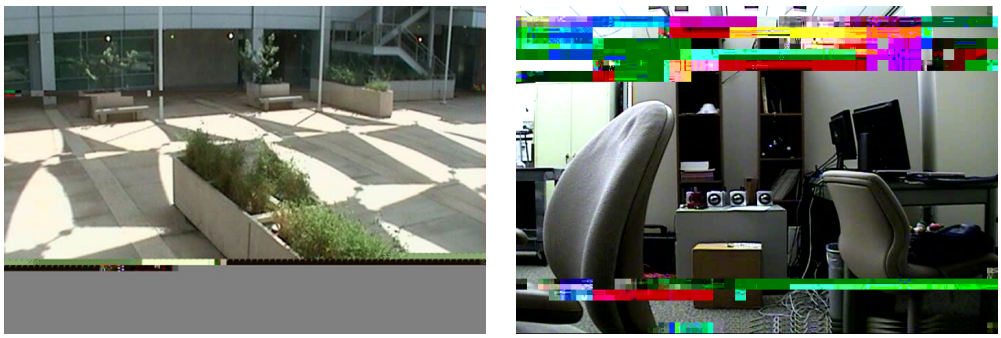


Figure 2.2: Stream corruption caused by network congestion may manifest in different ways depending on the video format. Left: corrupted Motion JPEG stream due to partial data, right: corrupted MPEG-4 stream due to partial data.

	Panasonic WVNS202		Axis 215 PTZ	
Configuration	640×480 px, 0% compress		704×480 px, 0% compress	
Cameras per bridge	2	3	2	3
Frame delay (seconds)	< 1.0	$> \mathbf{6.0}$	< 0.5	< 1.0

Table 2.1: Camera behavior can vary radically across vendors and models. Under congested network conditions for example, cameras may permanently drop frames or attempt to resend missed frames at the expense of live data. The Panasonic camera in this case output "smoother" video (fewer frame drops between two successive frames) under heavy network congestion (until its onboard cache is exhausted) at the cost of delays in upwards of 6 seconds.

out to obtain better views of a scene as appropriate. This comes at the cost of increased complexity by requiring (manual or automated) control in order to take advantage of this capability. These cameras also contain moving parts, potentially affecting long-term maintenance. Static cameras on the other hand, are often less expensive and provide consistent scene coverage. In addition, they also often allow interchangeable lenses which can mimic some versatility of PTZ cameras by allowing one to customize a camera for certain applications, e.g., installing a wide-angle lens to cover a larger area or installing a sharp telephoto lens to capture the entrance of a certain doorway (note that barrel distortion caused by using wider lenses should also be taken into account). Even with wider lenses, however, static cameras may require more installations to cover the same area as PTZ cameras

and may do so with compromised quality (camera placement is often a balance between sacrificing area coverage for close-up detail).

- **Pan/Tilt/Zoom speed and magnification.** If PTZ cameras are used, the responsiveness of such camera commands should be taken into consideration when choosing between models, as some cameras may respond or move too slowly to be useful for applications such as active tracking. Since the timing/latency specifications are often omitted by camera manufacturers, it is strongly recommended to experiment with trial cameras and testing if their PTZ speed is adequate before purchasing. In addition, the level of *optical* zoom may be important depending on the detail required for specific applications and the camera's physical distance from the scene. For most applications, digital zoom is worthless (at the raw capture stage) and should only be done in data processing.

- **Progressive vs. Interlaced cameras.** All other things equal, progressive cameras should be chosen over interlaced cameras where possible. This may not aways be the case, however, as progressive models may offer reduced frame rate, resolution, or cost substantially more. While interlaced cameras can usally perform on-camera de-interlacing to avoid the combing artifacts inherent to interlaced video, such techniques tend to wash out fine detail for static objects and result in ghosting effects on moving objects ones (the alternative, processing only every other line in the video, also effectively halves the vertical resolution). There may be some exceptions to choosing a progressive camera, such as when a CMOS-sensor progressive camera has a rolling shutter which is so slow that its video exhibits noticeable skew on moving objects (also known as the "jello effect" as often seen in handheld cameras when the camera is panned too quickly), but even this may

be preferred over the combing or ghosting artifacts from interlaced video.

- **Sensor size and CMOS vs. CCD** Sensor size is often more indicative of a camera's image quality than its stated resolution and this is true of video cameras as much as photographic cameras. Larger sensors tend to offer less image noise (especially in low light conditions) and sharper image quality. These sensors are typically either CMOS or CCD. While both sensors are used to achieve the same thing, complementary metal-oxide-semiconductor (CMOS) sensors typically use a rolling shutter (i.e., light is captured in a sweep across the sensor) whereas charge-coupled device (CCD) sensors use global shutters (i.e., light is captured simultaneously across the sensor). The two are typically characterized by different kinds of artifacts each produces. For instance, CMOS sensors may suffer from skew on moving objects or scenes if its shutter speed is too slow, while CCD sensors are vulnerable to smearing artifacts when bright light sources overload a column or row of pixels. It is recommended to consider the typical environment the cameras will be used in (e.g., low light, indoor vs. outdoor) and to trial all candidate cameras where possible.

- **Bandwidth: video format, resolution, and frame rate.** Resolution and frame rate go hand in hand as they will (in addition to video format) directly affect the bandwidth required for transmitting and storage required for archiving. Typical video cameras offer VGA resolution (640×480) at 30 frames per second, but newer high-definition (e.g., 720p or 1080p) cameras are becoming more readily available. While 640×480 resolution may be usable for many computer vision processing applications, those interested in face recognition (or better yet, face reconstruction) may find VGA to be particularly challenging to work with. Net-

works with particularly demanding requirements may want to consider specialty cameras, e.g., super high-resolution cameras, hardware-stitched 360 ° cameras, or even high-speed cameras, though these tend to demand a premium. The output format of the camera will also affect image quality; in addition to the traditional and easy-to-decode Motion JPEG codec (essentially a large series of JPEG images concatenated together), many cameras also offer MPEG-4 output for reduced bandwidth and/or higher quality using the same bandwidth via interframe compression. Decoding the video for custom-built applications may be more difficult with MPEG-4 however, and video artifacts caused by stream corruption (e.g., network congestion, dropped packets) may appear less appealing (see Figure 2.2). With either format, we recommend using the open source libavcodec [31] library to facilitate decoding in custom applications.

- **Power requirements of camera.** Depending on the power requirements, cameras may be able to draw from existing power sources or require separate power supplies. Depending on the building or location, installing power cabling to the cameras may be easier than installing cabling for the data (in the case of a wired network) since a building's electrical architecture is usually more sophisticated than its network architecture. For the most remote installations which require more permanence than battery-operated sensors, readers may want to consider solar-powered wireless cameras.

- **Physical appearance and camera enclosures.** Appearance should not to be overlooked when it comes to installing cameras. If the cameras will be installed in an outdoor environment, large outdoor enclosures may invoke a sense of intimidation (see Figure 2.4a). It is recommended to take into consideration

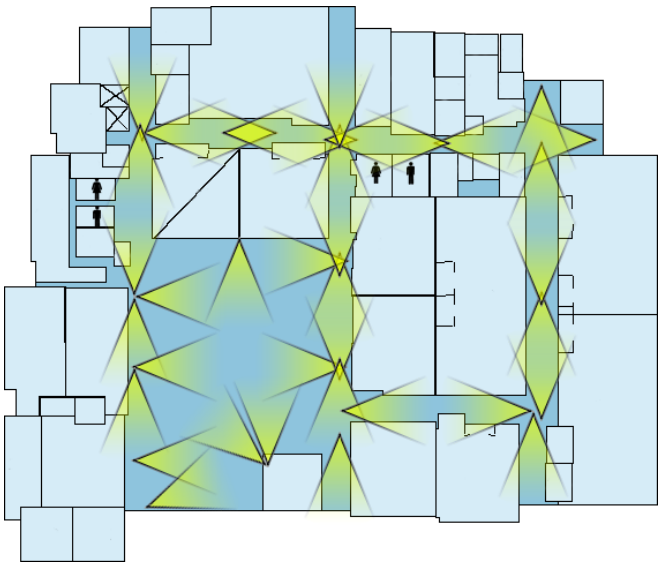


Figure 2.3: 37 camera locations cover the 14,300 square foot second floor of Engineering Building Unit II at the University of California, Riverside. Locations were manually selected and evaluated to ensure that usable fields of view were available for every square inch of the building from at least two viewpoints.

the environment the cameras will be installed to decide whether discreetness or visibility are higher concerns. As opposed to surface-mounting (installing a camera directly on a ceiling surface), flush-mounting (cutting a hole and installing a camera in the ceiling with the optics exposed) will provide a more discreet and streamlined appearance, but will require permanent alteration to the installation locations. If a network is temporary, readers are recommended to consider the life expectancy of the network before opting for flush mounting.

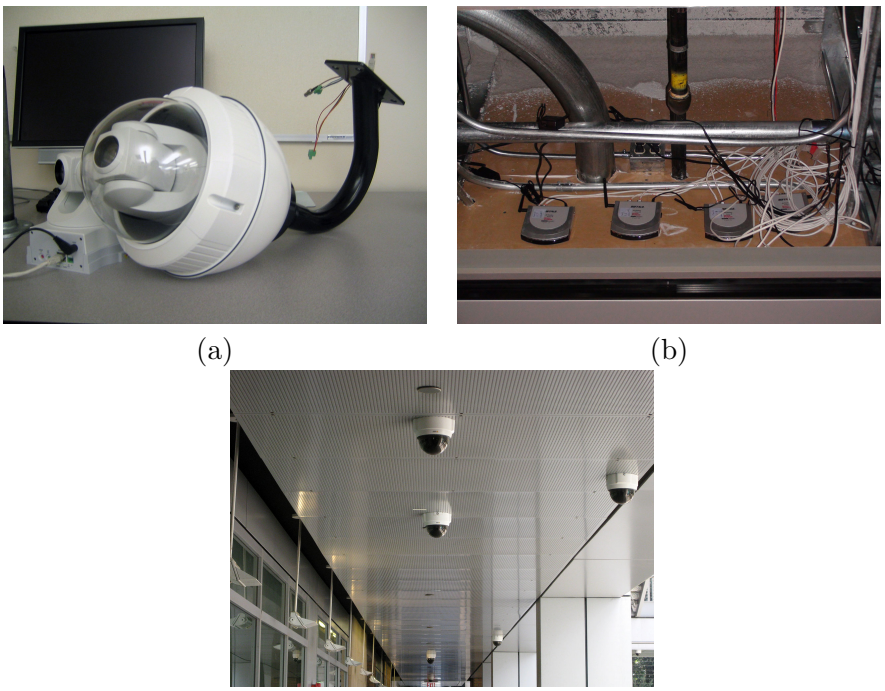
Our requirements and implementation

Initial specifications for the VideoWeb network required a minimum of VGA resolution (640×480 pixels) as well as a minimum of 20 frames per second as a threshold for acceptable real-time performance. In addition, we utilize digital IP cameras which provide a range of benefits such as streamlined processing (no digitizing required), relatively easy data storage, and simplified connectivity. The cameras are to be installed

in an indoor and outdoor building environment which includes locations such as remote open spaces exposed to rain and corridors void of sunlight. As a camera network for long-term applications with year-round use, battery-powered cameras are not sufficient and we instead use network cameras with power adapters.

Among conventional pan/tilt/zoom (PTZ) cameras considered were the Panasonic WVNS202, Axis 214, and Axis 215 cameras. Besides cost, factors influencing camera choice include performance, physical size, and availability of non-intimidating outdoor enclosures. The Panasonic cameras were deemed unsuitable after experiments which showed that the video stream begins to lag when the network becomes congested (Table 2.1). That is, in the event of network throughput issues which limit cameras to low frame rates, instead of dropping frames, the Panasonic resends cached video frames stored in its buffer. The Axis cameras on the other hand, drops frames, maintaining a relevant video stream despite low frame rates. Between the two Axis cameras, the 215 was selected as the primary camera (despite being an interlaced camera from lack of availability at the time of selection) due to lower cost and lower mechanical latency when issuing PTZ commands.

Using 45-degree fields of view for the cameras, 37 locations were selected for complete coverage of the 14,300 square foot building (Figure 2.3). As such, the network consists of 37 outdoor cameras (36 Axis 215 PTZ cameras and a larger Axis 214 PTZ camera overlooking a courtyard) and 16 indoor legacy cameras. Camera locations are selected such that every square inch of the building is viewable by at least two cameras. In testing, each camera is capable of outputting a sustained 2.65 MB/second of Motion JPEG (M-JPEG) video at a peak of 30 frames per second when set to the maximum resolution of 704×480 pixels and a minimal compression setting of 0 (out of 100). This



(a) (b)

(c)

Figure 2.4: Installation of the cameras: a) Axis 214 PTZ in an outdoor-rated enclosure; only one of these were installed (high above a large open courtyard) due to size and appearance), b) wireless bridges installed in the ceilings to make the IP cameras wireless, c) flush-mounted Axis 215 PTZ cameras in sealed indoor enclosures.

represents the maximum throughput and frame rate in an ideal environment (i.e., connecting to a camera via a direct ethernet connection and experiencing no frame drops). Other available resolutions of the cameras include 704×240, 352×240, and 176×120.

While the selected Axis 215 camera offers an outdoor dome enclosure, a dilemma was faced as there were no *discreet* outdoor enclosures for them; we had to find a way to make the cameras relatively weatherproof to withstand humidity and moisture. By choosing the Axis 215, we had to compensate for the lack of an available outdoor enclosure and improvise using the supplied flush-mount enclosures with smoked domes and surface-mount enclosures with clear domes, both designed for indoor installation. The solution was to use the surface-mount enclosures and make them weather-resistant by sealing the plastic seams with silicone sealant. In addition, the clear domes were inter-

changed with the smoked domes. The end result was a non-threatening camera dome suitable for surface-mounting at any of the 37 locations (see Figure 2.4c). Long-term effects of humidity, heat, and moisture on the cameras despite the sealed domes remains to be seen.

Electrical power was provided by installing dedicated power supplies in two of the building's electrical rooms and running conduit to the camera cluster locations where power outlets were installed. Since we had full control of the power by using our own power supplies, the cumbersome power adapters for the cameras were removed and the required power is supplied directly.

2.3.3 Choosing and Configuring the Network Hardware

The network hardware has a single purpose: to connect the cameras to the processing location(s) and to be as transparent as possible. Factors to consider when selecting network hardware include:

- **For Wired networking.** If IP cameras are being used, it is recommended to install the highest-rated network cable available (Cat-6 ethernet cable as of this writing) which can still reach its destination (generally 100 meters for gigabit ethernet or 55 meters for 10-gigabit ethernet using Cat-6a). The cost difference may be marginal (over Cat-5/5e, for instance) while providing overhead in robustness in the event that newer higher-bandwidth cameras are installed to replace aging cameras. Ethernet extenders may be required if cable lengths exceed cable specifications.

- **For Wireless networking: 802.11g vs. 802.11n vs. RF.** If wireless IP cameras are used, it will likely be a choice between 802.11g and the newer 802.11n.

If the choice is available (e.g., wireless bridges are being used to turn an ethernet camera into a wireless camera), 802.11n from our experience is a *major* upgrade from 802.11g for both increasing network throughput and signal strength. How much of an improvement may be influenced by congestion in the operating frequency range due to other wireless networks in the area. Determining a selection between analog RF transmitters, on the other hand, can be more difficult as the performance will vary more widely based on the power, frequency, and data being transmitted, as well as the environment. It is recommended to get a sample transmitter and to test each location cameras will be installed; this goes the same for wireless IP cameras, though wireless repeaters can be more-easily installed to extend ranges. In addition, selected wireless routers should offer (at minimum) gigabit capabilities, especially if a large number of cameras are expected to connect to it.

- **Wireless encryption.** Use anything besides WEP [34].

Our requirements and implementation

Since many IP cameras (the Axis 214 and 215 included) do not have built-in wireless connectivity, a wireless bridge is required to provide this functionality. As such, the wireless bridges serve a single purpose: connect the cameras to the routers. Since the camera locations are often situated in clusters, it is desirable if the bridges can support multiple clients (i.e., have more than 1 ethernet port). This quickly narrows down the selection. A conventional IEEE 802.11g bridge made by Buffalo was selected due to its support of 4 ethernet clients; IEEE 802.11n bridges were only available in 1-port versions at the time of selection. This paper does not delve into the pros and cons of individual wireless protocols, though literature on this specific topic has been recently

made available [56]. Performance testing on the Buffalo bridges revealed no outstanding issues, but prolonged testing showed that upgrading to 802.11n provides a worthwhile improvement for frame rates.

The wireless bridges were installed throughout the building in the ceilings and localized in clusters where possible to better facilitate maintenance and troubleshooting concerns (see Figure 2.4b). In total, 19 wireless bridges are used to provide connectivity for the 37 cameras. Though the bridges have 4 inputs, we only use 2; we do not take full advantage of the bridges' connectivity capabilities for a reason. We originally planned to optimistically use 3 cameras per bridge, but found 2 cameras (streaming simultaneously with maximum video settings) was the limit each bridge could support without experiencing heavy frame loss. The bridges are configured to communicate with the routers using WPA-PSK encryption.

At a maximum of 2.65 MB/s per camera (or 5.3 MB/s from each bridge), the network may be generating over 98 MB/s of data at peak performance. Gigabit routers are used to handle the amount of expected traffic and IEEE 802.11n capabilities are chosen to facilitate future upgrades. We use Linksys WRT350N routers for the first iteration of the network. Routers are split into two clusters receiving from two indoor locations. In total, 7 routers handle the traffic generated by the 19 bridges. The routers are configured to assign local addresses to the cameras and port forwarding is used to address the cameras from the servers.

2.3.4 Building the Server Hardware

Even with on-camera processing, it is still desirable to have external systems, either for data processing (due to much greater processing power) or storage. For digital networks, this system will likely be a number of computers. Whether specifying the

hardware for these machines or building from scratch, it is useful to keep in mind a number of factors:

- **Gigabit network connectivity.** When dealing with streaming video data, always opt for (at minimum) gigabit network adapters. This is especially true if a single machine is expected to process multiple camera feeds. A gigabit network switch (or higher) is almost a requirement when connecting the servers together.

- **Hard drives.** For raw data processing, hard drive speed or capacity does not matter (all image-processing can be done from memory). For long-term storage, high-capacity hard drives in a redundant configuration (e.g., RAID 5) are recommended, though it is best to store these in a central high-density storage server (as opposed to distributed across several servers) in order to facilitate easier retrieval. Depending on the expected amount of constant incoming data, expensive high-RPM drives may or may not be necessary.

- **CPU.** Depending on the multi-threaded capabilities of the processing software (either your own or vendor-supplied software), multi-core processors (and even multi-socket motherboards) may provide a significant improvement in overall system performance. This is especially true if servers expect to process feeds from multiple cameras.

- **Memory.** Images will be loaded into and read from memory constantly. Faster memory will reduce overhead, but *more* memory will likely only waste money as video images (processed on a per-frame basis) will not occupy very much space, even when uncompressed. There are exceptions, however, e.g. when using super high-resolution cameras or for database applications which will cache large quantities of images (such as a face recognition database), so it is recommended to keep

expandability in mind (i.e., 64-bit operating systems and motherboard memory capacity).

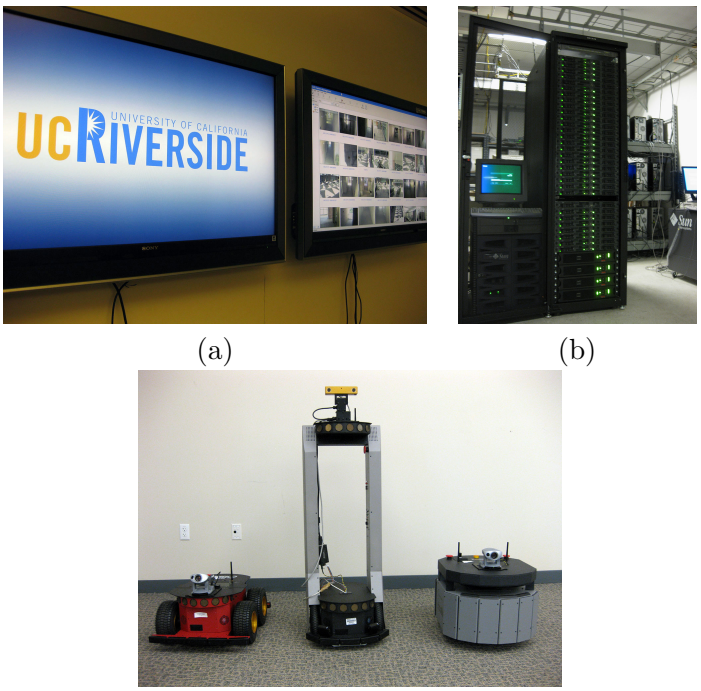
- **Operating system.** Though cross-platform code is preferred, the choice of operating system is determined mostly by the work/development environment the network operators and/or programmers feel most comfortable with. Server builders may want to take into account that most vendor-supplied software today is Windows-specific, however, but this may be irrelevant if you plan to develop your own processing software.

- **Server location.** If there are more than a few computers or servers in the system, it is recommended that they be moved to a dedicated server room with adequate cooling facilities; the heat, noise, and power consumption of all the servers can overwhelm most rooms.

- **Alternative power.** Uninterruptible power supplies (UPS) are recommended for all servers; their primary purpose is to allow the servers to gracefully shut down in the event of a power failure (or to buy time for backup generators to start up). This can be especially important for storage servers to help maintain the integrity of the servers' file systems.

Our requirements and implementation

We decided to go with a multi-core system in order to enable more streamlined parallel data processing of multiple cameras per computer. Also, with our server architecture we have 3 levels of processing. If later on this amount of processing power is insufficient, each computer should have a second vacant CPU socket for another pro-



(a) (b)

(c)

Figure 2.5: Processing hardware and mobile cameras: a) control interface and monitors in laboratory, b) 32-server processing backend connected to the interface, c) cameras mounted on 3 robots add mobility to the network.

cessor to allow doubling the processing power of the server farm if necessary without increasing the physical footprint of the system. For uniformity and to facilitate maintenance, all processing servers have the same hardware.

An idea to use conventional desktop computers for data processing was quickly discarded due to the difficulty in physically scaling ATX-sized desktop computers for a large number of cameras. Even using MicroATX cases would require a large amount of space to store the computers and would make moving the components/units particularly laborious and awkward. We instead opt for 1 height unit (1U) rack servers which can be housed in a single 42U rack enclosure with wheels for mobility.

In order to reduce contention over resources on the same machine from different camera processes, each processing server was specified with multi-core CPUs and fast memory (Intel Core 2 Quad Q6600 2.4 GHz CPUs and 2GB DDR2 800/PC 6400

memory). Though the Q6600 is not a true quad-core processor (2 dual-cores instead of 4 true cores), the support for additional threads is useful. Also, while we install 2GB for our initial setup, we also use motherboards which are expandable to 24GB of RAM for database applications in development. Gigabit ethernet cards are also selected to prevent any individual networking bottlenecks. Hard disks were given lower consideration, as most the processing nodes do mostly CPU processing and would not be storing data locally; the hard disks need only be sufficiently fast enough to run the operating system and RAM disks are setup in order to provide fast temporary storage for intermediate data. As such, conventional 80GB SATA hard drives are used. Application servers such as database or recording servers, on the other hand may emphasize larger and faster hard disks.

Thirty two identical servers were built and installed into a server rack (Figure 2.5b) and then connected to an interface server with a pair of control monitors as an interface (Figure 2.5a). The building housing the servers fortunately has a suitable server room with adequate air conditioning and power connectivity. Electricity usage monitors were used to measure power consumption of the servers. The servers mentioned, for instance, peak at 198W/1.65A when starting up, use 132W/1.14A when idle, and consume 175W/1.54A under full load on all cores and hard drives. This data was then used to specify the uninterruptible power supplies (UPS) for the servers, which consist of four 2U APC Smart-UPS 2200VA/120V batteries. Testing showed the batteries capable of supporting 8 servers each at full load for 5 minutes and 45 seconds, plenty of time to safely shut down (which can be configured automatically in software using UPS alerts) or to withstand short power outages.

2.3.5 Software System

In order to implement the tiered processing scheme of the servers, the software needs to be both clients and servers to facilitate the sending and receiving of video traffic and camera controls. Mid-level servers, for instance, may need to broadcast a stream of processed data to the high-level server for viewing by the user, while at the same time being able to download cropped object images from low-level servers.

The goal of the first software iteration was to control a networked camera using a customized program without the use of the supplied camera web interface or vendor-specific camera-management software included with most network cameras. One of the advantages of utilizing network cameras is that the camera control interface can be implemented through sending simple HTTP commands. To demonstrate this, a 10-line Python script was written for sending manual control commands to a camera. Once it was shown that it was easy to control the cameras, work started on a C++ application for the actual image processing.

2.3.5.1 Sample program - head tracking

The basic algorithm framework used for head tracking was based on a gradient and color-based tracker [7] with additional tweaks. The first implementation of this was done in C++ and MATLAB [95]. Testing showed this program to be too slow for real-time processing, so a second iteration was written in pure C++ using OpenCV [9]. The tracker began with tracking synthetic object data consisting of randomly rotated rectangles of various sizes against a white backdrop. Once this stage was satisfactory, the next task was to grab live data from the camera.

Instead of relying on vendor-supplied software development kits (SDKs) which

would have to be re-integrated into the processing software for potentially every type of camera, a generic camera controller was written. The SDK for the Axis cameras, for instance, relied on MFC-based [65] subroutines which would force development on Windows. In light of the amount of customization needed to incorporate a new SDK to do essentially the same things for different camera models, a generic cross-platform control framework was written from scratch. This control framework uses Boost.Asio [49] (a cross-platform socket wrapper) to directly send HTTP/1.1 [67] camera commands to a camera and uses the libavcodec library [31] to decode the streaming camera data. Using this approach, the software gains the benefit of being able to decode a large number of potential video streams and not just what a camera vendor has included with their SDK.

Networking communication between the three levels of servers is also implemented with Boost.Asio. For instance, processed results performed by the mid-level servers is compressed and broadcast as an M-JPEG stream, which is then parsed and displayed by the interface server.

2.4 Experiments for Performance Characterization and Optimization of the Video Network

2.4.1 Measurement software

Software that comes with most IP cameras ranges from small camera control programs to full surveillance station applications. However, even the most expensive or sophisticated of these vendor applications can be unsuitable since they are usually targeted toward security applications and recording, playback, and camera control are often their sole function. Evaluating performance using these applications is subjective

and raises the need for our own statistic-recording implementation.

A custom program was written to fulfill this function. Given an IP address and port number, the application proceeds to:

1. Establish a connection with the camera
2. Attempt to download the M-JPEG video stream
3. Parse the stream into individual JPEG frames
4. Record real-time statistics about the stream

The program records a number of statistics and measurements including bandwidth, shortest lag between two frames, and the average, minimum, and maximum amount of bandwidth required for each frame. The implementation is in C++ and uses the generic control framework written earlier.

2.4.2 Optimizing Camera Configuration

Depending on the task or application, there are numerous "optimal" ways to configure a network. For instance, maximizing video resolution and quality may be paramount for biometrics, particularly in face recognition where a large number of pixels on the face is beneficial to identifying features. Surveillance and alarm systems, on the other hand, may find reliability more important. For instance, it may be more important that every moment is recorded with minimal skipping (not only for evidence in the event of an incident, but also because security applications often employ vision-based motion detection). Object tracking in turn, may benefit most by sacrificing resolution in exchange for a high sustained frame rate.

Configuring the network may consist of changing camera parameters (e.g., resolution, compression) as well as physical network parameters (e.g., number of cameras

per bridge, number of bridges per router, number of routers per square foot). The later is helpful in introducing a metric for minimizing labor and monetary cost. We define 5 metrics for measuring camera network performance, the first two of which are used as configuration parameters.

1. *Resolution* (in pixels) - This measures the size of each video frame in pixels (the higher, the better). This parameter consists of 4 levels on the Axis cameras (704×480, 704×240, 352×240, and 176×120).

2. *Video compression* - This parameter represents the amount of *lossy* video compression applied to the video by the camera. For M-JPEG streams on the Axis cameras, this represents JPEG compression and ranges from 0 to 100 (the lower, the better). In our experiments, we test 5 of these levels (0, 20, 30, 60, and 100).

3. *Average frame rate* (in frames per second) - This measures the number of *complete* frames received per second, averaged over the duration of a measurement trial (the higher, the better). The frame rate may range from 0 to a maximum frame rate of 30 on the Axis cameras.

4. *Standard deviation of frame rate* - This measures the consistency of the video. For instance, there may be two video streams both 20 frames per second each, but the first may output a constant 20 frames per second while the second video may be sporadic and go from 30 to 0 to 10, back to 30 and so forth (but still average to 20 in the end). This metric is useful in evaluating the stability of the video (the lower the deviation, the better) and is measured by recording the delay between every two frames (in seconds with millisecond resolution) and calculating the standard deviation.

5. *Longest lag time between two complete frames* (in milliseconds) - This metric records the longest amount of time taken between any two consecutive frames (the lower, the better). This is insightful for evaluating a video stream's reliability (that is, it measures the longest amount of time a camera is "blind"). In addition to a depressed frame rate, this may be attributed to dropped/partial frames by the camera or data corruption/dropped packets undergone during transit.

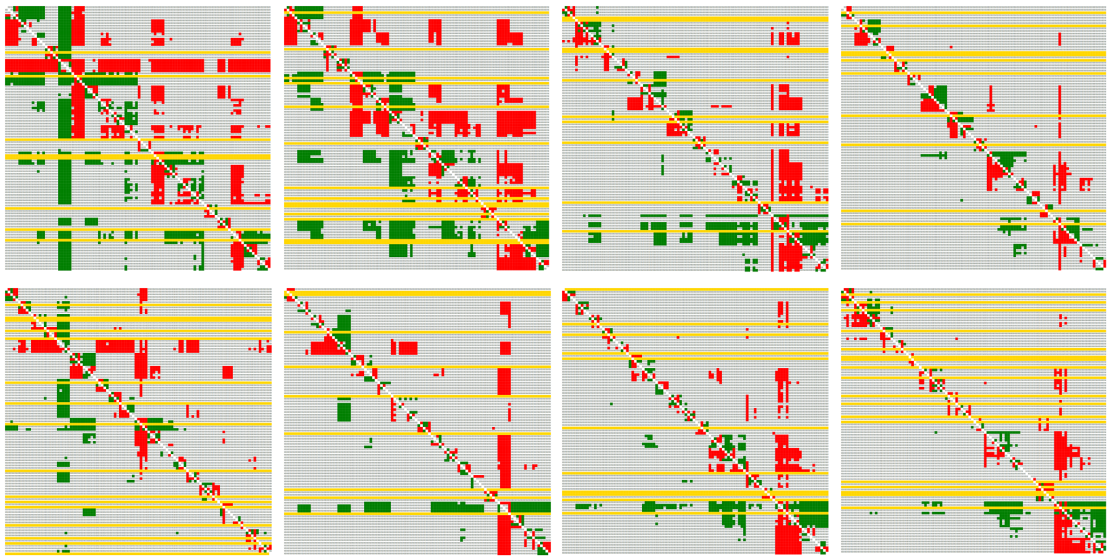


Figure 2.6: Measurement comparison matrices for 8 cameras. While cameras may exhibit variable performance even when using the same configurations, some configurations may be inherently better than others and exhibit similar performance across the network. To discover these configurations, 100 trials are performed on each camera under a variety of parameter configurations (i.e., resolution and compression) and each recorded measurement is compared for Pareto efficiency against the other 99 trials. This results in a symmetric matrix where vertical and horizontal axes indicate the measurements M_i and M_j, respectively (i.e., the top-leftmost square in each matrix indicates the relationship of M_1 against M_{100}). Red indicates that a particular M_i is inferior to a particular M_j, green indicates superiority, and a solid horizontal yellow line denotes rows which are completely Pareto-efficient (i.e., either superior or non-inferior against all other 99 trials).

2.4.3 Multi-objective Optimization Using Pareto Efficiency

We use the concept of Pareto efficiency to define which configuration of parameters is "better" than another. While this does not always tell a user which configuration should be used for a particular application, it serves to reduce the large number of possible configurations by showing which of those are usually "inferior"; a user only has to consider a configuration from the (potentially) much smaller Pareto set rather than every possible combination.

2.4.3.1 Inferiority and Non-Inferiority

Let M_1 be a vector of measurements of certain metrics for a camera and let M_2 be another trial of measurements on the same camera, but under a different parameter configuration. M_1 is said to be **inferior** to M_2 if and only if:

- every measurement in M_2 is equal to or outperforms the corresponding measurement in M_1
- one or more measurements in M_2 outperform the corresponding measurements in M_1

"Outperforms" is metric-specific and means "greater than" or "less than" depending on how the metric is defined (e.g., a *higher* frame rate outperforms a *lower* frame rate and a *lower* lag outperforms a *longer* lag). M_2 is said to be superior to or *dominates* M_1 if M_1 is inferior to M_2. Finally, M_1 and M_2 are both said to be **non-inferior** if neither is superior nor inferior to one another.

In order for a measurement M_i to be **Pareto-efficient** (amongst a set), it must be non-inferior to every other measurement in that set. That is, it possesses at least one *advantage* over every other measurement when compared one-on-one (e.g., M_1 has

higher frame rate against M_2, lower lag against M_3, ..., higher resolution than M_n). The Pareto set is the set of all Pareto-efficient measurements and ideally, allows a user to discard a large percentage of inferior parameter configurations from consideration when setting the cameras.

2.4.3.2 Data Collection

Data collection consists of varying the resolution and compression parameters and recording the measurements from 37 cameras. In total, we iterate through 4 resolutions (704×480, 704×240, 352×240, and 176×120) and 5 levels of compression (0, 20, 30, 60, and 100) each. Five measurement trials are captured for each of the 37 cameras per configuration (100 trials total per camera). Each trial consists of streaming from the camera for 600 frames or up to 2 minutes (whichever comes first).

Camera footage is tested at 5 various points in the day across all cameras. This exposes the data to a variety of video footage ranging from bright open areas with upwards of 20 moving people in the scene, to dark and grainy footage of cameras monitoring lonely halls.

After data collection is completed, each camera is optimized individually to minimize camera, bridge, or router bias. This is done in $O(n^2)$ via exhaustive search (where n is the number of trials to compare), comparing each measurement to every other measurement on the same camera. With 20 configurations and 5 trials per configuration, each camera produces a symmetric 100×100 matrix. The resolution/compression pairs which result in the Pareto-efficient measurements for each camera are later aggregated against the entire network.

2.4.4 Evaluation Results

After over 100 hours of data collection at varying times of day across two weeks, the Pareto sets for all 37 cameras are calculated (see Figure 2.6 for sample matrices of 8 cameras). Considering only configurations in the Pareto sets eliminates (on average) approximately half of the tested configurations as inferior and redundant.

Compression \ Resolution	100	60	30	20	0
176×120	46%	66%	46%	51%	74%
352×240	34%	46%	26%	34%	91%
704×240	51%	29%	17%	54%	97%
704×480	34%	31%	63%	94%	100%

Figure 2.7: Probability of configuration membership in any given camera's Pareto set.

After aggregating the resolution/compression parameters of the Pareto sets for the entire camera network, we found that, surprisingly, *every* configuration tested was in the Pareto set for at least one camera. This suggests that there is no global network-wide consensus that any camera configuration is inferior to any other; every (tested) setting was Pareto efficient for at least some camera. Calculating the percentages of the Pareto set memberships, however, reveals that the cameras tend to exhibit a "preference" for certain configurations over others (see Figure 2.7). This is in line with the previous observation that roughly half of the tested configurations are not preferred (less than a majority agreement between the cameras). It is not surprising to see higher percentages on configurations with either the maximum resolution or minimal compression since they already optimize at least one metric by definition. However, configurations such as 176×120/60% and 704×240/20% reveal local optimum which is potentially very useful for some practical applications of the video network. Using a more fine-tuned set of

compression levels, we would likely be able to find more such points, aiding in the creation of a useful set of presets for specialized applications.

In order to evaluate the relative performance of the configurations, the measurements for each camera are normalized across all measurements on the same camera and then averaged on a per-configuration basis across all cameras using the same configuration. Figure 2.8 shows the relative performance of the top 8 configurations for the entire network. Intuitively, increasing either the resolution or decreasing the compression (resulting in higher bandwidth) has the effect of a reducing the frame rate, producing a more discontinuous video stream, and increasing maximum lag time. These top configurations can then be considered as candidates for a number of applications or environments. The max resolution/0 compression configuration in Figure 2.8a, for instance, may be a good candidate for face recognition (so long as fast frame rate is not required), while face reconstruction may favor the max resolution/20% compression in Figure 2.8c due to its substantial increase in frame rate. An alternative approach to this general network optimization, however, is to optimize specifically for certain tasks.

2.4.5 Task-based Optimization

Instead of conducting exhaustive tests to find Pareto-efficient configurations, the presented multi-objective approach can also be used to optimize network parameters for specific applications or tasks. This can be done in much the same way as with the other performance metrics quantifying application-specific performance (e.g., face detection rate, smoothness of tracked objects trajectories) and adding them to the multi-objective metrics. Optimizing the network for face recognition at an airport, for instance, may be done by performing the same Pareto-efficiency tests on the precision

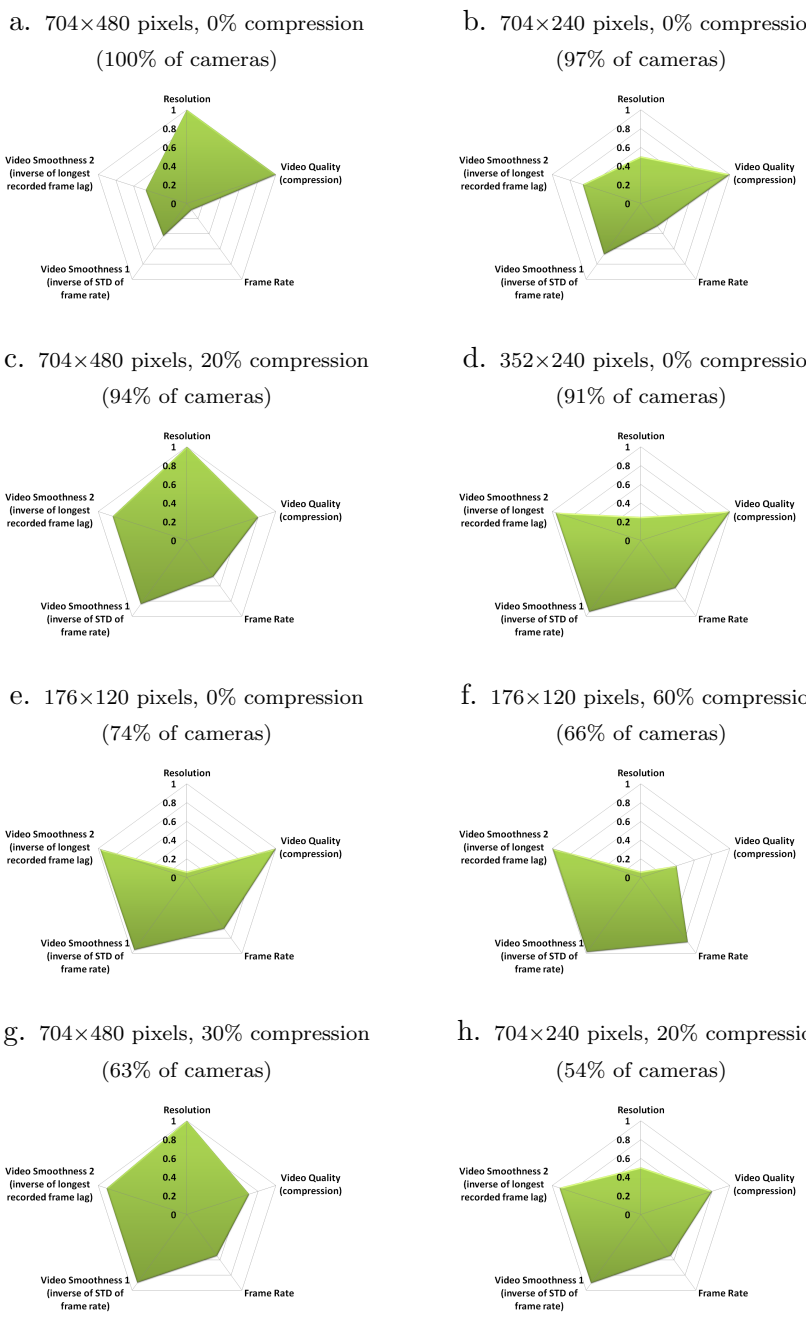


Figure 2.8: The top 8 dominating camera configurations as chosen by 37 cameras. Graphs are ordered by the percentage of cameras in which the particular configuration was Pareto-efficient and all metrics are normalized to 1.0 across all cameras. Clockwise from the top: resolution ranges from 176×120 to 704×480 (higher is better), JPEG compression settings range from 0 to 100 (lower is better, so inverse is shown), and frame rates range from 0 to 30 FPS (higher is better). For measuring the "smoothness" of outputted video, the standard deviation of the frame rate (recorded at 1-second intervals) and maximum lag time between any two sequential frames is recorded (lower is better, so inverse is shown).

and recall rates returned by a face recognition algorithm. In order to take advantage of the video produced across all the network configurations, it is recommended to record the streams during testing so that tasks which can be performed offline can be optimized with greater flexibility (e.g., a face recognition algorithm can be continuously tuned and repeatedly tested against the dataset without actually having to reconfigure the network). Tasks such as continuous PTZ tracking on the other hand, would have to be performed alongside the live data streaming.

2.5 Conclusions

We have designed an software-reconfigurable architecture for a wireless network of a large number of video cameras and implemented a working system by building the servers, installing the cameras, writing the software, and configuring the network to support it. Further, we gained insight into configuring the network's cameras by defining a set of metrics and discovering Pareto-efficient camera configurations by performing multi-objective optimization on a large volume of real data recorded by the system.

The idea persists that if one has a camera network with 30 FPS cameras, one will be able to obtain the said 30 frames per second regardless of network configuration or parameters. Though this may be true in a controlled test environment, the performance expectation should not be so optimistic for real-world wireless implementations. Even using the most preferred Pareto-efficient configurations on a non-congested network, it is shown that frame rates will most certainly suffer and that trade-offs must be made.

During a large workshop hosted in the building, however, it was observed that frame rates of the cameras would periodically drop and we later found that these drops coincided with breaks given during the workshop. Suspicious that a number of open

and local 802.11g networks may be congesting our network, a cluster of bridges were upgraded from 802.11g to 802.11n. In daily usage, frame rates were seen to reach up to 20 FPS for even the most bandwidth-intensive configurations (such as 704×480 resolution with 0% compression) where they were previously achieving typically only 3 FPS (even when other bridges in the network were not in use). While this makes a case for upgrading to 802.11n, this also suggests that network congestion from other networks may play a large role in frame rates and that networks may wish to operate in a dedicated frequency range.

In situations when even hardware upgrades can still not achieve sufficient performance, however, we would like to emphasize that partial data is still important. Rather than having algorithms which assume that the data consists entirely of complete video frames (and are only capable of processing such frames), real-time computer vision algorithms should take advantage of as much information as is available to them; the constant stream of partial frames which may only be missing the last few rows of data can still be tremendously useful for a number of applications.

This article was originally published in the EURASIP Journal on Image and Video Processing (JIVP) 2010 [71].

Chapter 3

Real-Time Pedestrian Tracking with Swarm Intelligence

3.1 Abstract

Multi-person pedestrian tracking in real-world video is a critical functionality for many applications in human-computer interaction (HCI) and security/surveillance, e.g., crowd analysis, anomaly detection, and target identification. In this paper, we present swarm intelligence algorithms for pedestrian tracking. The most widely implemented solution for tracking, particle filters, are outperformed by Particle Swarm Optimization (PSO) [106]. We present a modified Bacterial Foraging Optimization (BFO) algorithm which poses an opportunity to improve on the speed and accuracy of PSO for real-time tracking applications. We show that BFO can overcome existing limitations of currently-used evolutionary computation techniques (e.g., slow performance for real-time use) by distributing search agents more effectively. In our experiments, we show that BFO's search strategy is inherently more efficient than PSO under a range of variables with regard to the number of fitness evaluations which need to be performed

when tracking. We also compare the proposed BFO approach with other commonly-used trackers and present experimental results on all 26 corridor videos of the CAVIAR dataset as well as on difficult PETS2010 S2.L3 videos of crowd scenarios.

3.2 Introduction

Human tracking systems generally consist of three main components: 1) detection, 2) tracking, 3) track association, and analysis or the fusion of multiple trackers [57]. Since much of the computational effort is spent on tracking, improving the speed and effectiveness of the tracking component can greatly benefit many surveillance and security applications [6]. It will facilitate real-time performance and improve the accuracy of higher-level track association. A number of challenges, however, make pedestrian tracking difficult:

1) Change in appearance: The visual appearance of pedestrians may change gradually or suddenly between frames, e.g., a person may turn (changing his/her silhouette) or shrink or grow in size depending on his/her distance from the camera.

2) Non-uniform lighting and shadows: Light may not be uniform across a scene, may change across frames, and pedestrians will generally cast shadows. Non-uniform lighting results in the changed appearance of the same pedestrian depending on the time and location in the scene. Further, shadows complicate a pedestrian's appearance by altering color and size information which may not carry over into other environments (e.g., walking from a outdoor hall with concrete floors to a room with red carpet).

3) Uncalibrated cameras: Uncalibrated cameras provide no definite ground plane or distance information for a tracking algorithm to utilize; regardless of whether the cameras are fixed or non-static, e.g., movable pan/tilt/zoom (PTZ) cameras. Manually

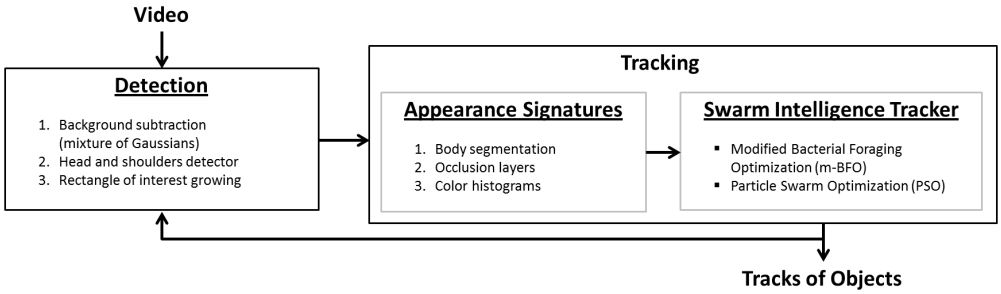


Figure 3.1: System diagram of the tracking system. Parts-based appearance signatures are used to drive a swarm intelligence search algorithm for pedestrian tracking.

calibrating cameras is a labor-intensive task which is not always easy or feasible in some cases. Since additional calibration data can only help a tracking algorithms (e.g., by providing depth information or constraints which can be taken into account into a tracker's fitness function), we focus on tracking in the more common uncalibrated camera environment.

4) Occlusion: Pedestrians may be occluded by other pedestrians, the environment, or even exhibit self-occlusion. This makes it more difficult to differentiate between two or more pedestrians involved in the occlusion as to obtain sufficient visual evidence for determining the appearance/size of a pedestrian.

5) Crowds: People often walk together in groups, complicating separation and association. Tracking in dense crowds may be infeasible, even for human interpretation.

This paper focuses on the object tracking component of a human tracking system which must handle the bulk of the above challenges. The rest of this paper is organized as follows: Section 3.3 overviews related work in pedestrian tracking and summarizes our contributions, Section 3.4 details the technical components of the proposed tracking approach, Section 3.5 presents experimental results on the CAVIAR [32] and PETS 2010 [30] datasets, and Section 3.6 offers closing remarks.

Table 3.1: Summary of major approaches to the pedestrian tracking problem.

Approach	**Principle**	**Comments**
Optical flow	Initialize and track a large number of points on an object in order to infer *flow* or a 2D vector field [64] which models an object's movement.	Produces smooth tracks of entire object surface, but computationally intensive; requires high frame rate video or the objects move slowly.
Template or shape-based	Learn visual appearance of objects [82] (e.g., texture, shape, silhouette from different angles, lighting conditions, etc.) to perform continuous object detection.	Useful against occlusion, but potentially time-consuming; appearances may be highly application-dependent and difficult to apply to other scenes.
Behavior modeling	Analyze movement of objects [54] (e.g., walking patterns) or underlying structure responsible for object's movement (e.g., muscles, skeleton) in order to predict movement or narrow search space.	Susceptible to occlusion and presence of multiple objects; often used in conjunction with other approaches.
Kernel-based	Initialize one or more particles or *kernels* (e.g., color histogram similarity metric) on image and use some search strategy (e.g., distribution-based, random walk) to dictate how to sample points and how to combine results of sampled points to estimate object location (e.g., majority voting, weighted average, probability distribution).	Occlusion and fast-moving objects addressed by increasing particle count; linear scaling as count of people or particles; performance very dependent on search strategy; most popular are particle filters [41], Mean Shift [19], and CamShift [8].
Multi-camera	Use multiple cameras to overcome occlusion [6, 33].	Greatly increases the amount of data needed to process, but effective against occlusion; often used in conjunction with other approaches.

3.3 Related Work and Contributions

While there are many individual approaches to the tracking problem [102], they can typically be broken down into five major categories: optical flow-based approaches, template/shape-based approaches, behavior modeling, kernel-based approaches, and multi-camera approaches. Each approach possesses its own strengths and weaknesses

and is usually targeted toward a specific application. Table 3.1 summaries the principle of each approach.

The most successful approaches for pedestrian tracking usually focus around Particle Filters [29, 41], Mean Shift [8], or detection-based tracking [97]. An alternative approach considers a family of biologically-inspired evolutionary computational algorithms known as swarm intelligence, a subset of kernel-based approaches. In this category, the most popular approaches are Particle Swarm Optimization (PSO) [47,105] or a combination of Ant Colony Optimization with the above approaches [38]. Of those that are available, they only show results on simple scenarios [63, 110]. These trackers are often used to generate short-term "tracklets" which are then used in methods such as Data Association Tracking (DAT) [58,91] to produce long-term inter or intra-camera tracks. The 5 major categories of tracking methods are discussed below.

3.3.1 Tracking Approaches in the Literature

3.3.1.1 Optical flow

Optical flow is an approach which selects a number of feature points or pixels on an object and attempts to track the displacement or *flow* of each individual pixel between every consecutive frame. That is, for each point, a local neighborhood of pixels is searched for the new position; this is usually performed in the intensity gradient domain [64]. The number of points is typically initialized to be very large (in order to increase overall robustness) and the displacement of these points is used to estimate a 2-dimensional vector field which models the object's movement in the 2D image.

Optical flow is popular in computer animation (e.g., for facial expression tracking [98]) or for applications in which very precise tracking is required and a controlled

environment is available (e.g., processing can be done offline, high-frame rate cameras can be used, there is cooperation with the tracked subject, and lighting conditions are constant). Due to the need of performing correspondence for each feature point in every frame, the approach exhibits an exponential runtime as the radius of the search neighborhood increases. This makes it ill-suited for real-time applications, tracking fast-moving objects, or in situations where occlusion is common, i.e., correspondence will fail between two frames when a feature point moves (or appears to move) outside of the search neighborhood. While there are efforts to make optical flow more robust to occlusion [40], research on improving performance for fast-moving objects often involves artificially increasing a video's frame rate [61,62], an approach which is not always feasible.

3.3.1.2 Template/shape-based approaches

Template or shape-based approaches involve tracking an object based on a database of learned appearances. This database may be a collection of textures (e.g., person tracking may utilize a database of learned hand, leg, and face appearances [82, 107]), shapes (e.g., using an ellipse-shaped template to track a head [7], tracking people using learned silhouette shapes [60]), or a combination of the two.

Since partial matches can be found on an object even if other parts are occluded, template/shape-based methods are often used to address occlusion. This comes at the cost requiring a suitable database of appearances to be created beforehand, in addition to the large computational overhead associated with performing matching, especially if the database is large or the templates are complex.

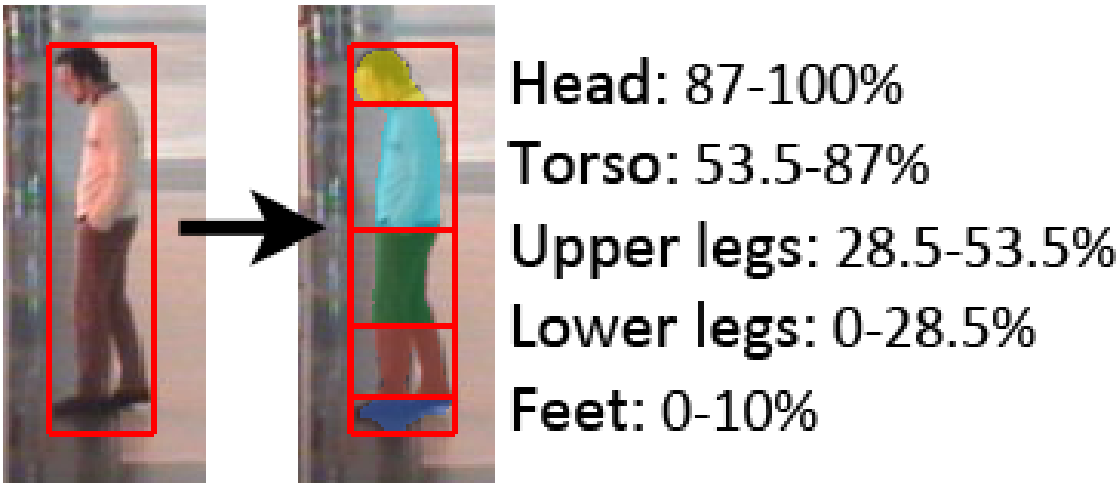


Figure 3.2: Pedestrian ROIs are segmented into multiple partitions based on statistical human body ratios to create more robust appearance signatures.

3.3.1.3 Behavior modeling

Behavior modeling involves learning either an object's movements [54] (in order to predict its next location) or learning the behavior of objects relative to a scene [33] (in order to determine areas of high/low activity, common trajectories, etc.). The former usually models single objects (e.g., humans and animals) where the later often models groups of objects (e.g., crowds of people, traffic patterns), but both methods can be used to narrow the search space when searching for an object. In fact, behavior modeling is often used as a preprocessing step to other tracking strategies in order to improve their track accuracy, but it can also be used by itself using simple correlation as its search metric (i.e., matching based on some pattern or mask appropriate to the application).

Applications include analyzing human behavior to predict movement [36] as well as using physical models (such as skeletons) to model an object's movement [53]. Behavior modeling is particularly susceptible to scenes where there are many objects, however, since object interactions may be complex or impractical to model. As such, it is better suited toward applications where the number of objects being tracked is small.

3.3.1.4 Kernel-based approaches

Kernel or particle-based approaches use a single kernel/particle or number of particles in conjunction with some search metric (such as a color [92] or texture histograms [89]) in order to perform tracking. They differentiate themselves from optical flow in that the particles are used to infer a central location, whereas optical flow aims to track the individual location of each particle. Popular single kernel-based approaches include the Mean Shift [19] and CamShift [8] algorithms and popular multiple particle-based approaches include Kalman [44] and Particle filters [41].

In the case of Mean Shift and CamShift, tracking is based on distributions created by the search metric, centering the current search window at the mean of the previous window. Though simple to implement, they are not robust to factors such as fast movement (objects easily leave the search window) or occlusion (the distribution being tracked appears to disappear). Kalman filters, on the other hand, use a linear model to estimate an object's location by minimizing squared error between particles. Since the dynamical model used in Kalman filters is assumed to be linear, it does not work well when the noise is multi-modal (e.g., when an object appears to be disjoint due to occlusion). Extended Kalman filters attempt to address this [11]. Particle filters also evolve from Kalman filters and focus on dealing with non-linear dynamical models and multi-modal densities by sampling the prior probability and weighting those samples based on observations. Thus, Extended Kalman filters and particle filters can recover from occlusion to some extent, provided that there are enough particles on the image. In practice, however, the larger the number of particles, the higher the computational cost, thereby making real-time processing challenging to implement.

3.3.1.5 Multi-camera approaches

Another method aimed at solving the occlusion problem involves using multiple cameras. Multi-camera tracking is an emerging field which handles the occlusion problem by using multiple views of the same scene in order to exploit the fact that occluded objects in one camera may be easily separable or completely visible in another. This, however, comes with an increase in the amount of data to process and often requires prior calibration of the cameras in order to ensure proper correspondence between camera views.

Aside from its popular use in computer animation for motion capture (where multiple cameras are used to track retroreflective markers placed on capture subjects), such systems are often deployed for surveillance-oriented applications with a focus on tracking multiple humans [14, 24, 33]. Multiple cameras in close proximity have also been used in a manner similar to stereo cameras (i.e., a pair of cameras which simulates binocular vision) in order to address occlusion [43], as well as multiple stereo cameras themselves being used for tracking [108].

3.3.2 Swarm Intelligence

Swarm intelligence is a family of evolutionary algorithms which are modeled after the collective behavior of biological swarms, such as ants (Ant Colony Optimization [18]), bees (Bee Colony Optimization [94]), or birds and fish (Particle Swarm Optimization [47]). In the context of tracking, swarm intelligence algorithms belong to the set of kernel-based approaches. A swarm can be represented by a number of particles initialized on an image which move in subsequent frames according to some basic rule in order to ultimately achieve a common goal. For example, in Particle Swarm Optimiza-

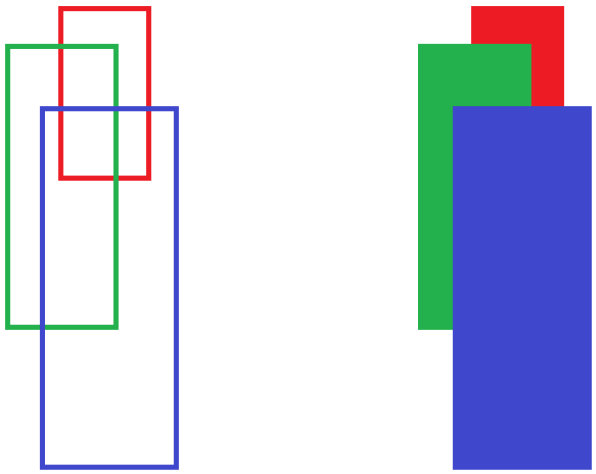


Figure 3.3: Occlusion compensation using occlusion layers. Left: ROIs are sorted by the y-value of their bottom edge and placed into layers in descending order. Right: the masking of lower layers by higher layers helps minimize the influence of occluding pedestrians when computing signatures.

tion (PSO), particles are randomly initialized with a random velocity and direction and search their current location for the object at each step. Particles determine their next search position as a function of moving towards their previous best location and moving towards the swarm's best location, resulting in swarm-like behavior where particles tend to move together, but each particle follows its own unique path depending on it's initial position.

3.3.3 Contributions

In this paper, we make the following contributions:

1. We adapt the Bacterial Foraging Optimization algorithm for real-time tracking with changes which improve its speed and accuracy. We call the new BFO algorithm as m-BFO.

2. We provide system-level performance measurements of both tracking accuracy and computational efficiency of swarm intelligence algorithms PSO [47] and BFO

(a) (b) (c)

Figure 3.4: How pedestrians appear to a tracker. (a) Detected pedestrians, (b) fitness space for the pedestrian using color similarity in the CIE LAB color space (brighter = higher fitness), (c) fitness space after applying the foreground mask to the input image to reduce background noise. Note that the foreground mask is not as effective in the last PETS2010 video due to sudden lighting changes in the scene.

[76] as well as on commonly-used CamShift and Particle Filter trackers on the difficult CAVIAR dataset and the PETS2010 *S2.L3* crowd scene. Note that of the numerous approaches to pedestrian tracking, *particle filter* is the most commonly used low-level tracker in practice. From the results of [106], however, it has been shown that the evolutionary computation algorithm Particle Swarm Optimization (PSO) is an improved and specific variation of particle filter which outperforms the traditional particle filter implementation. We show that tracking with m-BFO is better than PSO. Therefor we claim that the proposed m-BFO is better than the particle filter.

3. Show that segmenting the a pedestrian body into separate partitions improves tracking performance by making objects more distinct (as opposed to popular single ROI-based approaches).

3.4 Technical Approach

Figure 3.1 shows an overview of the system. Background subtraction is first performed on input frames. Detected blobs from the subtraction are processed with a pedestrian head-and-shoulders detector, e.g., [55, 97]. The ROI is then extended to encompass the whole body and body segmentation is performed to create five smaller ROIs. These are used to create five separate appearance signatures/histograms used by a fitness function. This fitness function can then be used by a tracker to locate the target in subsequent frames.

3.4.1 Pedestrian Detection

In order to create an online tracking system, initialization of target locations must be completely automated. Background subtraction is used to facilitate the extraction of areas of interest. The modified Gaussian mixture model (GMM) background subtractor proposed in [112] is used to dynamically learn the background as the input frames are received (with the additional benefit of removing shadows). We remove shadows in order to not confuse the the appearance signature. This creates a foreground mask which can be used to detect areas of the image to focus a more sophisticated pedestrian detector. In all the experiments, the background subtractor is run without prior training.

Labeling the connected components of the foreground mask returns a set of

blobs which may potentially contain a pedestrian. Running the pedestrian detector in this manner (as opposed to running it on the entire image as in traditional approaches) provides a significant speedup.

In order to automate pedestrian initialization, a Viola-Jones detector [13] trained to detect heads and shoulders is used on all connected components whose area is at least as large as the detector's minimum size (a 22×20-pixel rectangle in this paper) and not already covered by a tracker; this results in another significant speedup. An omega-shape head-and-shoulder detector may be used as an alternative [20, 55]. The ROIs of positive detections are extended downward to encompass an estimate of the entire body:

$$height_{body} = height_{head\_shoulders} \times R$$

where R is a fixed ratio that is dependent on the detector used ($R = 3.1$ in this paper). The full body ROI can then be segmented into multiple partitions.

3.4.2 Appearance Signature

3.4.2.1 Body segmentation

Traditional blob-based trackers use the entire ROI of a pedestrian to compute a signature for tracking. Using multiple sub-signatures localized to patches of the pedestrian, however, increases the descriptive power of a signature while maintaining its generality when the segmentation is reasonable. While the part-based appearance model has previously been proposed for fingerprinting objects for target re-identification, it is seldom used specifically for tracking; those that do are often viewpoint-dependent or require high resolution [42, 107]. We extend the approach to the lower level of tracking as opposed to just cross-camera matching.

We propose separating the pedestrian ROI into 5 separate horizontal partitions using statistical ratios of the human body [4]. These 5 partitions are as follows: 1) Head: the top $87 - 100\%$ of the full ROI, 2) Torso: $53.5 - 87\%$, 3) Upper legs: $28.5 - 53.5\%$, 4) Lower legs: $0 - 28.5\%$, 5) Feet: $0 - 10\%$.

This paper makes the reasonable assumption that the orientation of the video is level (i.e., the physical horizon would appear as a horizontal line in the video) and the image is not distorted. However, it is possible to extend the proposed approach to handle non-upright pedestrians (such as those caused by distortions of fish-eye lenses) by taking an extra step to compute the orientation of the pedestrian [25]. Figure 3.2 shows the segmentation scheme.

3.4.2.2 Occlusion layers

When initializing the signature of a pedestrian to track, it is important to minimize noise. Two sources of noise include: 1) pixels from the background and 2) pixels from other occluding pedestrians. In order to minimize background noise, the foreground mask is applied to the pedestrian's ROI. In order to handle occluding pedestrians, we use the occlusion layers scheme proposed in [59]. Given the reasonable assumption that the video is captured from a location above the physical ground plane (i.e., the camera is not positioned below the floor or looking down a steep incline), pedestrians can be sorted by the y-coordinate of the bottom edge of their ROI. Defining $(0, 0)$ to be the top-left corner of an image, ROIs whose bottom edge has a higher y-value than other ROIs are closer to the camera (assuming that the pedestrian is on the ground plane, e.g., not jumping). If two or more pedestrian ROIs intersect one another, they are sorted into layers based on their bottom y-values (higher value = higher layer); ROIs in lower layers become occluded by ROIs in higher layers (see Figure 3.3).

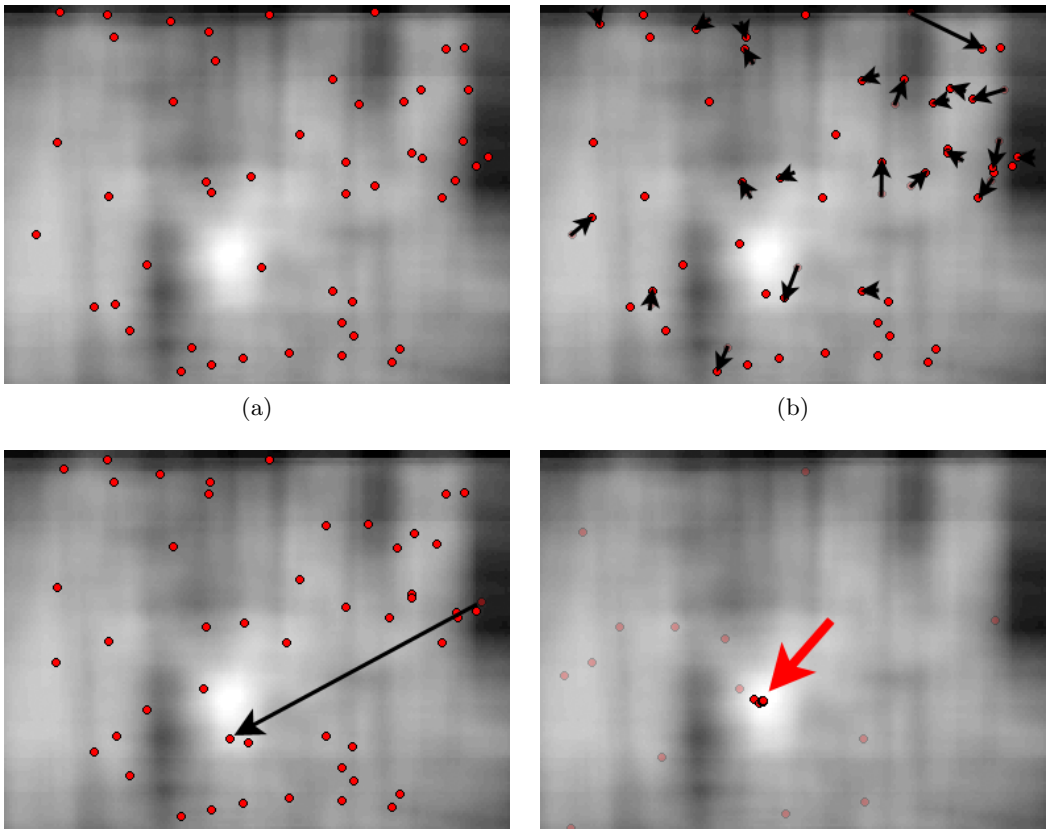


Figure 3.5: Behavior of a single Bacterial Foraging Optimization swarm searching for a pedestrian. (a) Random initialization, (b) gradient-hill climbing in random directions, (c) death/rebirth of agents with poor fitness to location of agents with best fitness, (d) target location based on consensus of the best agents.

3.4.3 Tracking Using Swarm Intelligence

Swarm intelligence is a family of evolutionary stochastic optimization algorithms modeled after biological systems. A swarm consists of a number of particles which independently follow a search strategy which allows the swarm to accomplish the common goal of finding an area of optimal fitness. Two such algorithms are Bacterial Foraging Optimization and Particle Swarm Optimization.

Algorithm 1 Modified BFO (m-BFO) algorithm for tracking

$I \leftarrow$ image to search
$Target \leftarrow$ hist and prev. location of object to search for
$R \leftarrow$ number of reproduction steps
$C \leftarrow$ number of chemotaxis steps per reproduction
$S \leftarrow$ max number of swims per chemotaxis step
$Step \leftarrow$ swim step size in pixels
$T \leftarrow$ number of agents to relocate per reproduction
$P \leftarrow$ probability a non-immune agent gets relocated
$Thresh \leftarrow$ minimum fitness to trigger early termination

procedure BacterialForaging
    if I is first frame of target **then** ▷ Init first frame
        Initialize agent locations on I
    end if ▷ Early termination?
    if $fitness(Target_{loc}, I, Target_{hist} \geq Thresh$ **then**
        return $Target_{loc}$
    end if
    for R reproduction steps **do** ▷ Begin search
        for C chemotaxis steps **do**
            for all agents A **do**
                $d \leftarrow$ random direction
                for up to S swims **do**
                    $l \leftarrow$ new location $Step$ pixels from A toward direction d
                    $f \leftarrow fitness(l, I, Target_{hist})$
                    if $f > A_{current\_fitness}$ **then** ▷ Lookahead
                        $A_{current\_fitness} \leftarrow f$
                        $A_{current\_location} \leftarrow l$
                    else
                        Break
                    end if
                end for
            end for
        end for
        for all top T agents A with best fitness **do**
            $A_{immunity} \leftarrow true$ ▷ Elitism
        end for ▷ Death/rebirth
        Move the T agents with worst fitness to
            locations of the T agents with best fitness
    end for
    for all agents A where $A_{immunity} \neq true$ **do** ▷ Elimination/dispersal
        Relocate A to random position with probability P
    end for
    $Target_{loc} \leftarrow$ best location based on all agents A ▷ Return updated location
        where $A_{immunity} = true$
    return $Target_{loc}$
end procedure

Algorithm 2 Particle Swarm Optimization (PSO) algorithm

$Image \leftarrow$ image to search
$Target \leftarrow$ hist and prev. location of object to search for
$P \leftarrow$ number of agents
$I \leftarrow$ number of iterations
$W_a \leftarrow$ weight of momentum
$W_b \leftarrow$ weight of particle best location
$W_c \leftarrow$ weight of global best location

procedure ParticleSwarm
    if $Image$ is first frame of target **then**
        Randomly initialize P swarm particles
        $Global_{fitness} \leftarrow 0$
    end if
    for I iterations **do**
        for all particles P **do**
            $fit \leftarrow fitness(P, Image, Target_{hist})$
            if $fit > Localbest_{P,fitness}$ **then**
                $Localbest_{P,fitness} \leftarrow fit$
                $Localbest_{P,location} \leftarrow P_{location}$
            end if
            if $fit > Global_{fitness}$ **then**
                $Global_{fitness} \leftarrow fit$
                $Global_{location} \leftarrow P_{location}$
            end if
            $r_0 \leftarrow rand(0, 1)$
            $r_1 \leftarrow 1 - r_0$
            $V_{t+1} = W_a V_t + W_b r_0 Localbest_{P,location}$
                $+ W_c r_1 Global_{location}$
            $P_{location} = P_{location} + V_{t+1}$
        end for
    end for
    return $Global_{location}$
end procedure

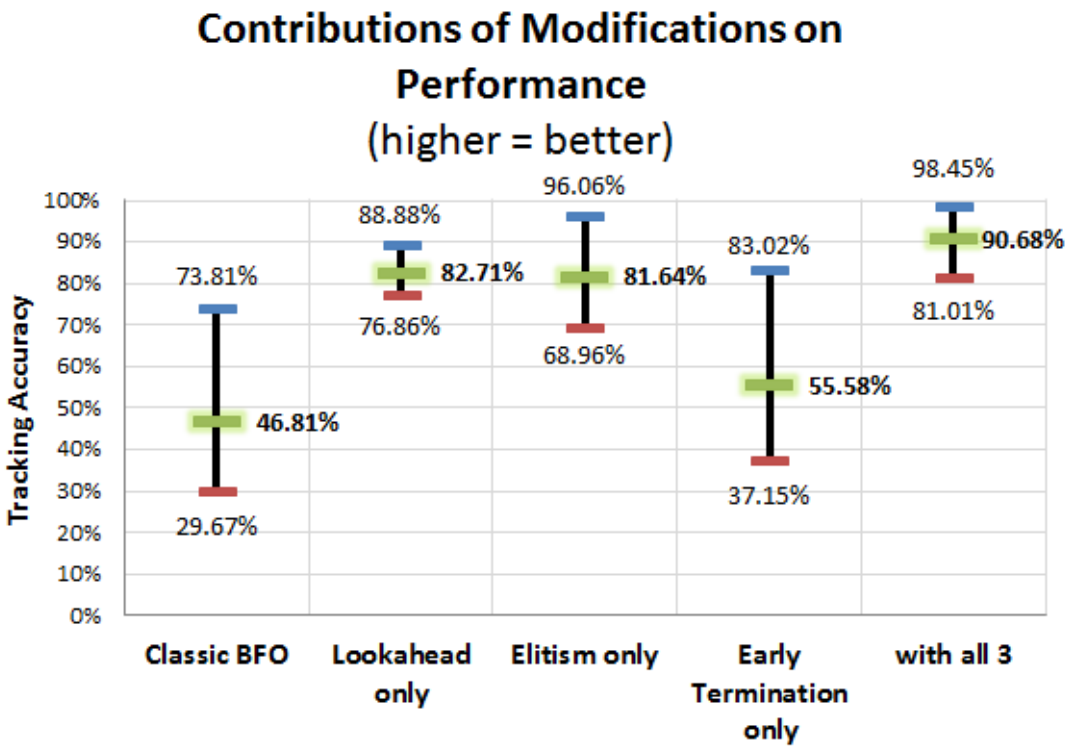


Figure 3.6: Performance contributions of each modification of m-BFO over traditional Bacterial Foraging Optimization (BFO) on a sample dataset [68] shown with minimum, average, and max performance measurements.

3.4.3.1 Bacterial Foraging Optimization

Bacterial Foraging Optimization (BFO) [76] is a stochastic evolutionary swarm intelligence search algorithm designed to model the movement and feeding behavior of *E. coli* bacteria. A swarm consists of a number of particles or "agents" which move or "swim" and "tumble" through an environment searching for concentrations of food (or regions of high fitness from a feature space point of view). Given an image, a swarm of agents is first randomly initialized on the image. The algorithm consists of R "reproduction" loops which execute a number of C "chemotaxis" or movement loops. In each chemotaxis loop, all agents "tumble" (choose a random direction) and are allowed to "swim" (or sample) up to S times in steps of size $Step$ in a gradient hill-climbing

manner. At the end of each reproduction step, the bottom T agents with the worst fitness scores die off and an equal number of agents are born at the locations of the T best agents. In this manner, resources are quickly allocated to regions of higher fitness. At the end of the algorithm, the agents undergo an elimination/dispersal step which randomly relocates agents with probability P. This step helps to simulate a changing environment such that the swarm does not fully converge and cease to track in succeeding frames. Figure 3.5 shows the behavior of a BFO swarm in a fitness space.

BFO has never been used previously for pedestrian tracking [69], yet possesses traits which make it suitable to the problem. The near-uniform coverage of the search space is useful for overcoming occlusion (whereas many other approaches lose track once they converge). In addition, the fast propagation of agents to regions of high fitness reduces overhead of having the agents gradually making their way toward global-best fitness regions. This paper utilizes BFO with the enhancements proposed in [68] for additional characteristics such as early termination, lookahead, and elitism:

Early Termination allows the algorithm to terminate early if positions of adequate fitness have been discovered early on.

Lookahead allows the algorithm to accept or reject fitness samples during the gradient hill climbing to improve local optimality.

Elitism allows the search agents of highest fitness to stop searching after each round and to select the final location based on a consensus of these agents as opposed to a single highest-fitness sample.

Algorithm 1 summarizes the full procedure and Figure 3.6 shows the performance contributions of each of the above modifications on a recorded dataset [68].

3.4.3.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) [47] is the most commonly-used swarm intelligence approach for tracking and is modeled after the social behavior of schools of fish and flocks of birds. In addition to tracking, it has also been used for a number of optimization problems [3] and has many variations [96, 105]. PSO's standard search strategy moves particles according to a linear combination of their current speed and direction, the vector to the position of their local best fitness, and the vector to the position of the swarm's best fitness (see Algorithm 2).

3.4.3.3 Fitness function

Illumination changes of a scene are handled by the choice in the color space. In order to find the best color space to represent a pedestrian signature, body segment-based signatures were extracted using 8 different color spaces from the Viewpoint Invariant Pedestrian Recognition (VIPeR) dataset [37], a popular dataset used for research in person re-identification [27]. The 8 tested color spaces are: RGB, HSV, HSL, YUV YCrCb, YUV YIQ, CIE XYZ, CIE Lab, and CIE Luv. For each pedestrian, signatures were extracted for each color space for each body part using sets of 2, 4, 8, 16, 32, and 64 bins per color space component. The signatures were then boosted using a C4.5 tree and tested for recognition with 10-fold cross validation. The color space configuration with the greatest discrimination capability from the tests is YUV YIQ with 32 bins per component. Figure 3.10 shows the boosted recognition performance of each color space as the number of histogram bins changes. In order to find the relative discrimination power of each individual body part, the recognition rates were tested on a per-body part basis. Figure 3.11 shows that the most important body parts (for YUV YIQ) are first

the torso, then the upper legs, lower legs, feet, and finally the head.

The fitness or objective function used by the two optimization algorithms is computed as follows. For each body part, a color histogram is extracted for all pixels in the YUV YIQ color space using $N = 32$ bins for each of the Y, I, and Q components and normalized to the sum of pixels in each body part. The similarity between two histograms H_0 and H_1 is computed as the histogram intersection:

$$intersect(H_0, H_1) = \sum_{i}^{N} min(H_0^i, H_1^i)$$

An appearance signature S is comprised of $M = 5$ histograms, one for each body segment. The fitness function for comparing two pedestrian signatures S_0 and S_1 is defined as the average histogram intersection between the histograms of each body part:

$$similarity(S_0, S_1) = \frac{\sum_i^M intersect(S_0^i, S_1^i)}{M}$$

This fitness function can then be used by any conventional tracker for computing the distance between a query location in an image versus a target signature. Figure 3.4 shows exhaustively-generated fitness spaces for sample pedestrians by computing the fitness at every pixel against an initialized signature.

In order to address tracking of similarly-dressed pedestrians, an additional trajectory smoothness component is useful to give preference to areas of fitness which more closely resemble the current trajectory. Trajectory smoothness at a point can be defined by both smoothness in velocity and smoothness in direction:

$$V_t = P_{t+1} - P_t$$

$$smoothness = W_d \times \frac{V_{t-1} \cdot V_t}{|V_{t-1}||V_t|} + W_v \times \frac{2\sqrt{|V_{T-1}||V_t|}}{|V_{t-1}| + |V_t|}$$

where V_t is the vector between the previous point and a proposed point and W_d, W_v are weights which sum to 1.0 and control emphasis on either direction or velocity. The final fitness function is:

$$fitness = W_s \times smoothness + (1 - W_s) similarity$$

where W_s controls the influence of smoothness.

3.5 Experimental Results

All trackers are implemented in C++ and experiments are performed using a single Intel Xeon E5345 2.33GHz quad-core CPU. Multi-target tracking is facilitated by the use of multi-threading, e.g., the tracking of each pedestrian is performed using its own thread.

3.5.1 Datasets

Experiments are performed on all 26 corridor videos of the CAVIAR dataset [32] as well as the difficult "S2.L3" crowd scenario of the PETS2010 [30] dataset. The CAVIAR videos pose a challenge for low-level trackers because: 1) they are low resolution Common Intermediate Format (CIF) videos (382×288), 2) the ROI sizes of pedestrians changes dramatically depending on pedestrian's position in the corridor (the size of the same person may vary by as much as 450% from one end of the corridor to the other), and 3) there is much occlusion between pedestrians, not only in the middle of a pedestrian's path, but also when pedestrians first enter as well as leave the field of view. We have categorized the CARVIAR videos into 4 categories to better illustrate the performance of the algorithms on different scenarios: 1) Very Few Pedestrians, 2)

Few Pedestrians, 3) Many Pedestrians, and 4) Very Many Pedestrians. Table 3.2 shows statistics and categorizations of all 26 CAVIAR videos.

While higher resolution (768×576), the PETS2010 video is difficult due to heavy occlusion, lighting changes, and the shear number of individuals within the crowd (over 40). Figure 3.7 shows sample frames from the CAVIAR videos and Figure 3.8 shows frames from the S2.L3 video. Note that official groundtruth ROIs are available for the CAVIAR dataset, but are not publicly available for the PETS2010 dataset (groundtruth tracks for the S2.L3 video were created manually for the experiments).

3.5.2 Effect of Parameters and their Selection

Bacterial Foraging Optimization. The number of agents A, reproduction steps R, chemotaxis steps C, and swims S control the runtime of the BFO algorithm: $O(A * R * C * S)$. The step size $Step$ controls how fast agents swim (higher for bigger steps, lower for finer local search). The number of agents T to relocate controls how much the algorithm balances its search; higher values increase exploitation of areas of higher fitness while lower values increase exploration of the entire search space.

Particle Swarm Optimization. The number of particles P and number of iterations I are the two primary parameters which control the runtime of the PSO algorithm: $O(P * I)$. Setting $W_a < 1.0$ makes particles tend to slow down, > 1.0 makes particles tend to speed up, and $= 1.0$ preserves the current momentum of a particle. The balance of parameters W_b and W_c affect the influence of a particle's local best location so far vs. the swarm's global best location so far; setting $W_b > W_c$ leads to higher results and increased local search while setting $W_b < W_c$ makes the swarm collapse sooner on a location (though at the risk of converging on a local minima).

Parameter Selection. Parameters are manually optimized for each individual tracking

algorithm and then fixed for all experiments. To optimize the parameters of the PSO tracker, for instance, the tracker was run on a random subset of 13 CAVIAR videos using various parameter configurations. The parameters with the best average performance were then selected for the full tests. This was done individually for each tracker. Similarly, the smoothness parameters were also manually optimized.

For the exhaustive grid search tracker, the fitness is exhaustively computed at every pixel and a mean filter of $halfwidth = 7$ is applied to find the point of highest fitness. For PSO, $P = 30$, $I = 10$, $W_a = 1.0$, $W_b = 0.04$, and $W_c = 0.04$. For BFO, $P = 10$, $E = 1$, $R = 10$, $C = 1$, and $S = 5$. We use the CamShift available in OpenCV and the Particle Filter from OpenCVX [88] using 30 particles. For the detection-based trackers, the pedestrian detector is run on both the whole image and on every foreground blob for every frame without the advantage of skipping a blob if a tracker is covering it. Weights for smoothness are $W_d = 0.5$, $W_v = 0.5$, and $W_s = 0.01$.

3.5.3 Performance Metrics

Tracking accuracy is defined as the percentage of groundtruth ROIs covered by the tracker initialized on that pedestrian. A query ROI $Query$ is considered to be tracking a target if its intersection with the groundtruth ROI GT exceeds at least 50% of their union:

$$is\_tracked(Query, GT) = \frac{Query \cap GT}{Query \cup GT} > 0.50$$

50% is selected as a the benchmark overlap rate in the same fashion as used by CAVIAR's performance statistics [32].

This prevents an ROI which encompasses the whole frame from being considered "tracked". An accuracy of "40%" on CAVIAR means that on average 53,000 of the 133,000 groundtruth ROIs are tracked.

Processing speed is evaluated in frames per second (FPS).

3.5.4 Tracking Results

3.5.4.1 CAVIAR Dataset

Figure 3.9 shows the impact of the parts-based segmentation, improving track accuracy on average 6.5% over the single blob approach. Figure 3.12 shows that the swarm intelligence approaches perform on par with Particle Filter and BFO even achieves this performance in real-time using fewer resources (Figure 3.13). Figure 3.16 details the performance on all 26 videos and shows that segmentation improved tracking accuracy on almost all videos (single blob approaches may succeed in cases when bad detections result in improperly-segmented ROIs).

3.5.4.2 PETS2010 Dataset

The PETS2010 S2.L3 video shows that the number of pedestrians (over 40) significantly impacts the speed of the system, bring the same BFO tracker down to 3.3 FPS (Figure 3.14). However, the swarm intelligence approaches continue to perform on par (and even out-perform) the Particle Filter (Figure 3.15).

3.5.5 Discussion of Results

Figure 3.9 shows that segmenting body parts increases the discrimination power of appearance signatures compared to traditional whole-body appearance signatures. Figures 3.12 and 3.16 show that the PSO and BFO swarm intelligence approaches achieve comparable results to the more-often used particle filter [41] while Figures 3.13 and 3.14 show that such performance is achievable at faster speeds than with particle filter.

3.6 Conclusions

We adapted the Bacterial Foraging Optimization (BFO) algorithm for real-time tracking with a modified algorithm (m-BFO) which improves on the speed of Particle Swarm Optimization (PSO). PSO in turn is an improvement over traditional particle filtering methods [106]. We also proposed a parts-based appearance signature for defining the fitness function. The proposed appearance signature approach made use of the observation that appearance-based pedestrian signatures can often be broken into discrete patches due to a person's clothing. We provided in-depth results of the tracking performance of PSO and BFO and several other commonly-used trackers on the CAVIAR dataset and the S2.L3 scenario of the PETS2010 dataset. Since most previous work relies on blob-based similarity, this work can easily be integrated to improve tracking performance of both low-level and higher-level Data Association Trackers (DATs) [10, 58].

Acknowledgments

The authors would like to thank Dr. Ninad Thakoor for helpful discussions and hardware configuration.

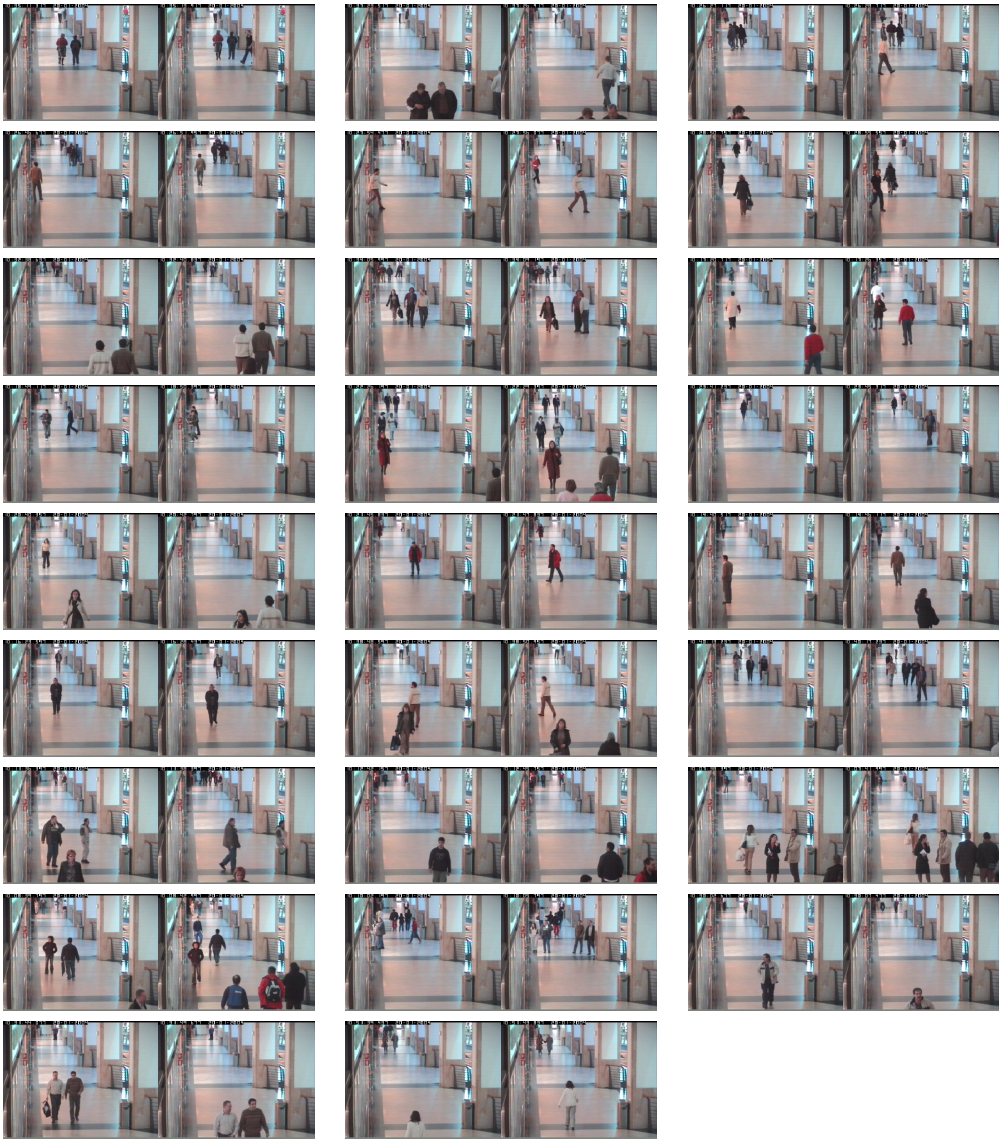


Figure 3.7: Sample frames from all 26 mall corridor videos of the CAVIAR dataset [32]. All videos have a resolution of 384×288 pixels and consist of a total of 35,913 frames and 131,288 pedestrian ROIs. The objective is to track all pedestrians entering and exiting the field of view under different shopping scenarios.

Figure 3.8: Sample frames of the "S2.L3" crowd scene from the PETS2010 dataset [30] (768×576 pixels, 240 frames, and 470 pedestrian ROIs). The objective is the track the two pedestrians labeled A and B as they join in walking with a large incoming crowd.

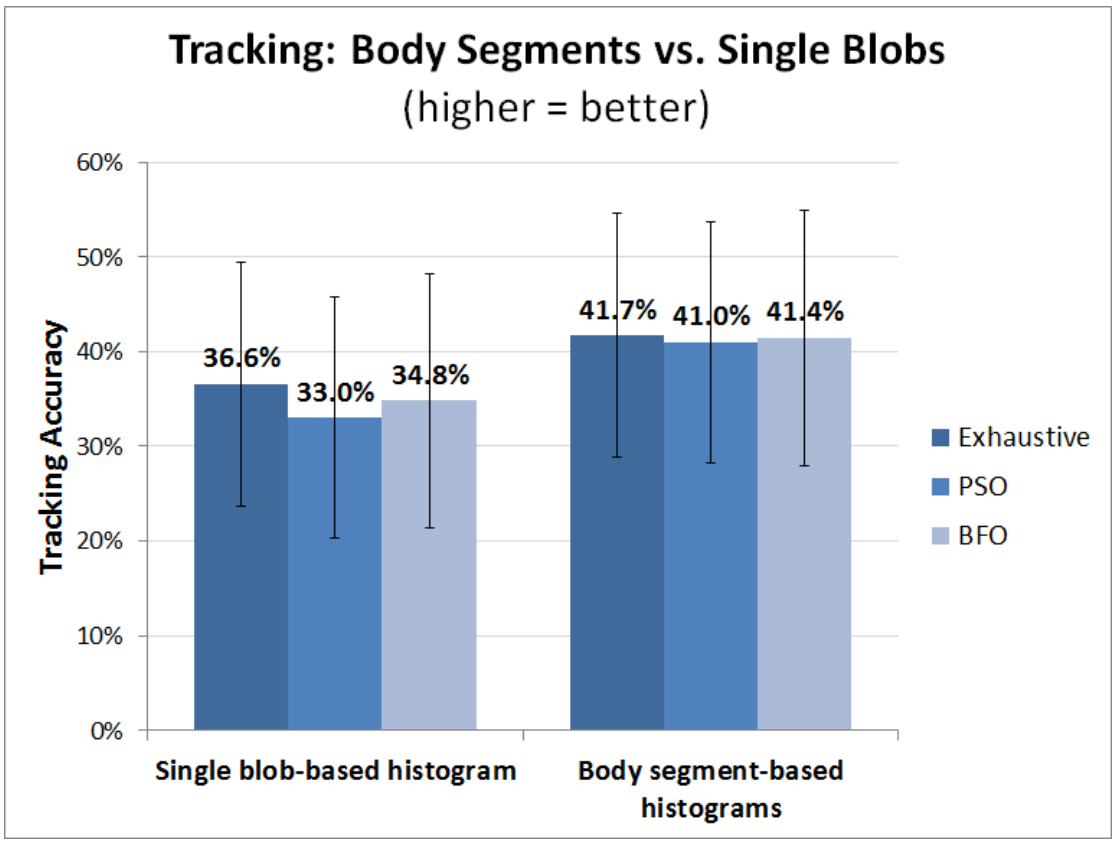


Figure 3.9: Segmenting the body into parts improves tracking performance as opposed to using traditional single blob-based histograms. Results are shown averaged with std. bars over all 26 corridor videos of the CAVIAR dataset, 30 runs for each video.

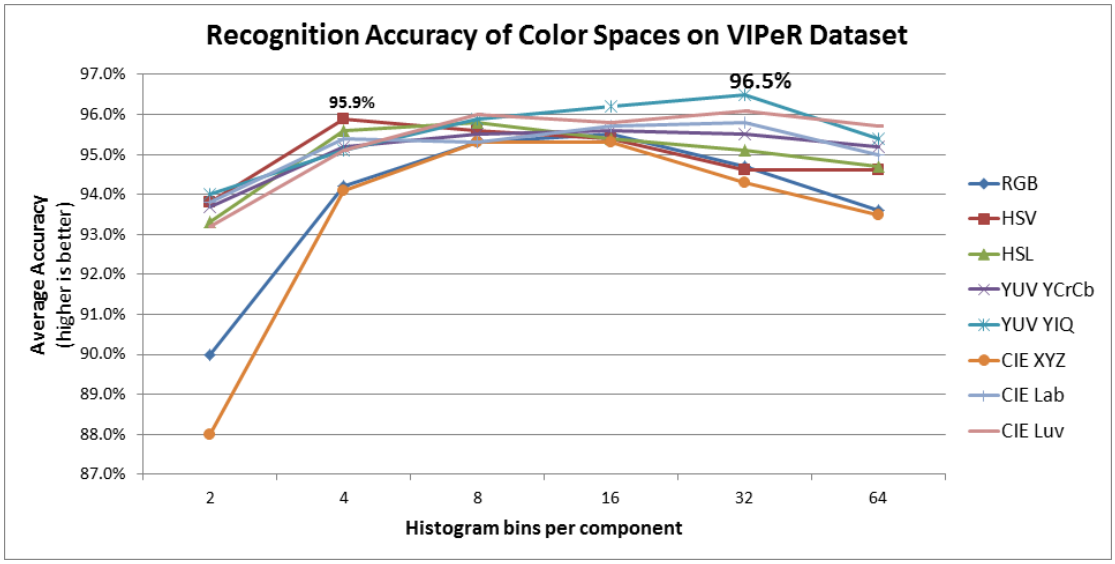


Figure 3.10: Average recognition accuracy of color histograms using boosted C4.5 trees on the Viewpoint Invariant Pedestrian Recognition (VIPeR) dataset [37].

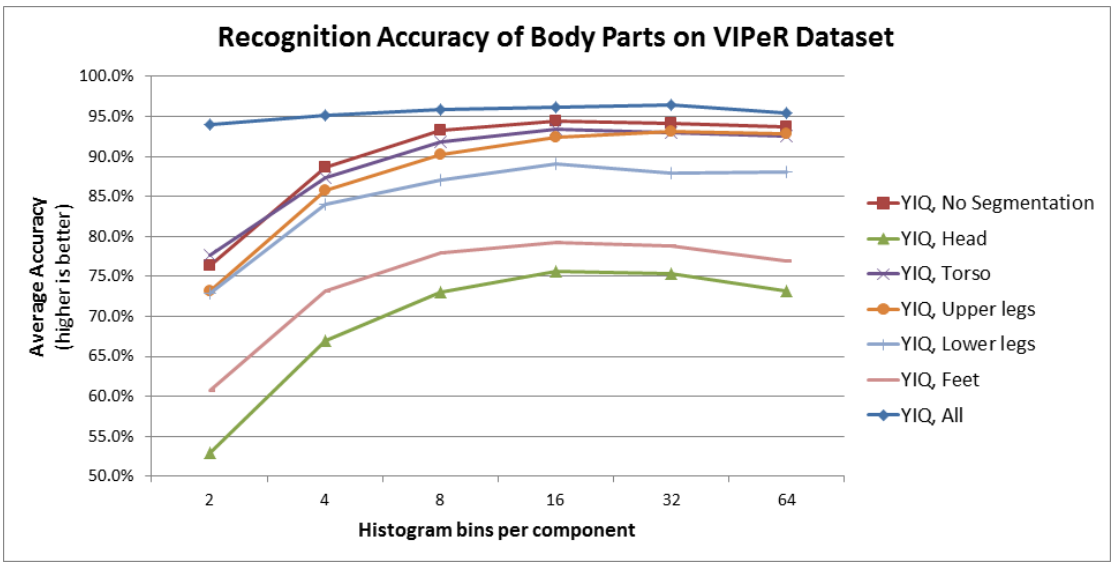


Figure 3.11: Average recognition accuracy of color histograms by body part on the Viewpoint Invariant Pedestrian Recognition (VIPeR) dataset [37].

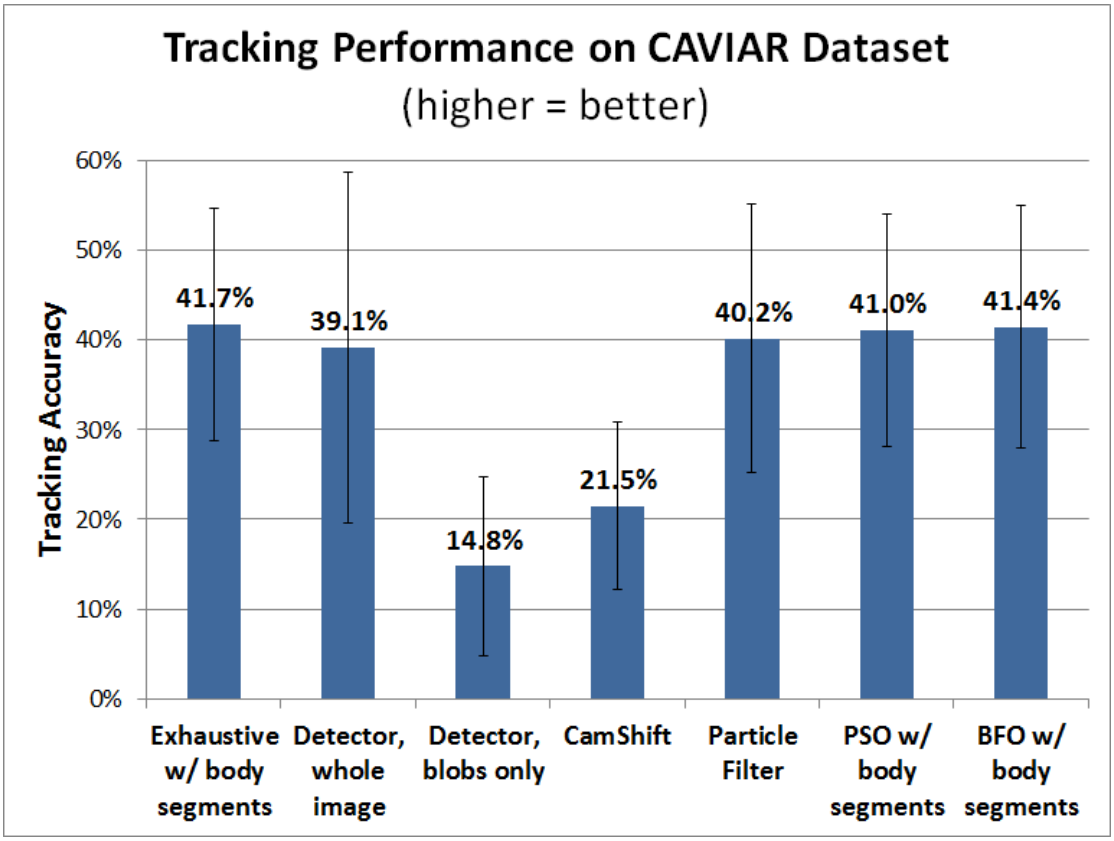


Figure 3.12: PSO, BFO, Particle Filter, and whole image detection-based tracking perform on par with exhaustive grid search. Of these trackers, BFO performs the fastest (see Figure 3.13). Results are shown averaged with std. bars over all 26 corridor videos of the CAVIAR dataset, 30 runs for each video.

CAVIAR Video	# Pedestrians	# ROIs	# Frames
Very Many People			
OneStopMoveEnter1cor	20	13,691	1,590
ShopAssistant2cor	20	11,942	3,700
ThreePastShop1cor	20	9,642	1,650
TwoEnterShop3cor	20	6,856	1,149
WalkByShop1cor	20	11,348	2,360
Many People			
TwoEnterShop2cor	16	7,930	1,605
OneStopMoveEnter2cor	14	8,418	2,237
TwoEnterShop1cor	12	7,190	1,645
TwoLeaveShop1cor	11	4,705	1,343
OneShopOneWait2cor	10	7,568	1,462
ThreePastShop2cor	10	9,452	1,521
Few People			
OneStopMoveNoEnter2cor	9	2,823	1,035
OneShopOneWait1cor	7	4,496	1,377
OneStopEnter1cor	7	2,390	1,500
OneStopEnter2cor	7	4,142	2,725
OneStopMoveNoEnter1cor	7	2,394	1,665
OneStopNoEnter2cor	7	2,950	1,500
ShopAssistant1cor	7	1,922	1,675
OneLeaveShop1cor	6	1,401	295
OneLeaveShopReenter2cor	6	2,147	560
Very Few People			
OneLeaveShop2cor	5	4,189	1,119
EnterExitCrossingPaths2cor	4	768	485
OneLeaveShopReenter1cor	3	408	390
OneStopNoEnter1cor	3	1,714	725
TwoLeaveShop2cor	3	802	600
Totals	254	131,288	35,913

Table 3.2: Statistics and categories for each video of the CAVIAR dataset.

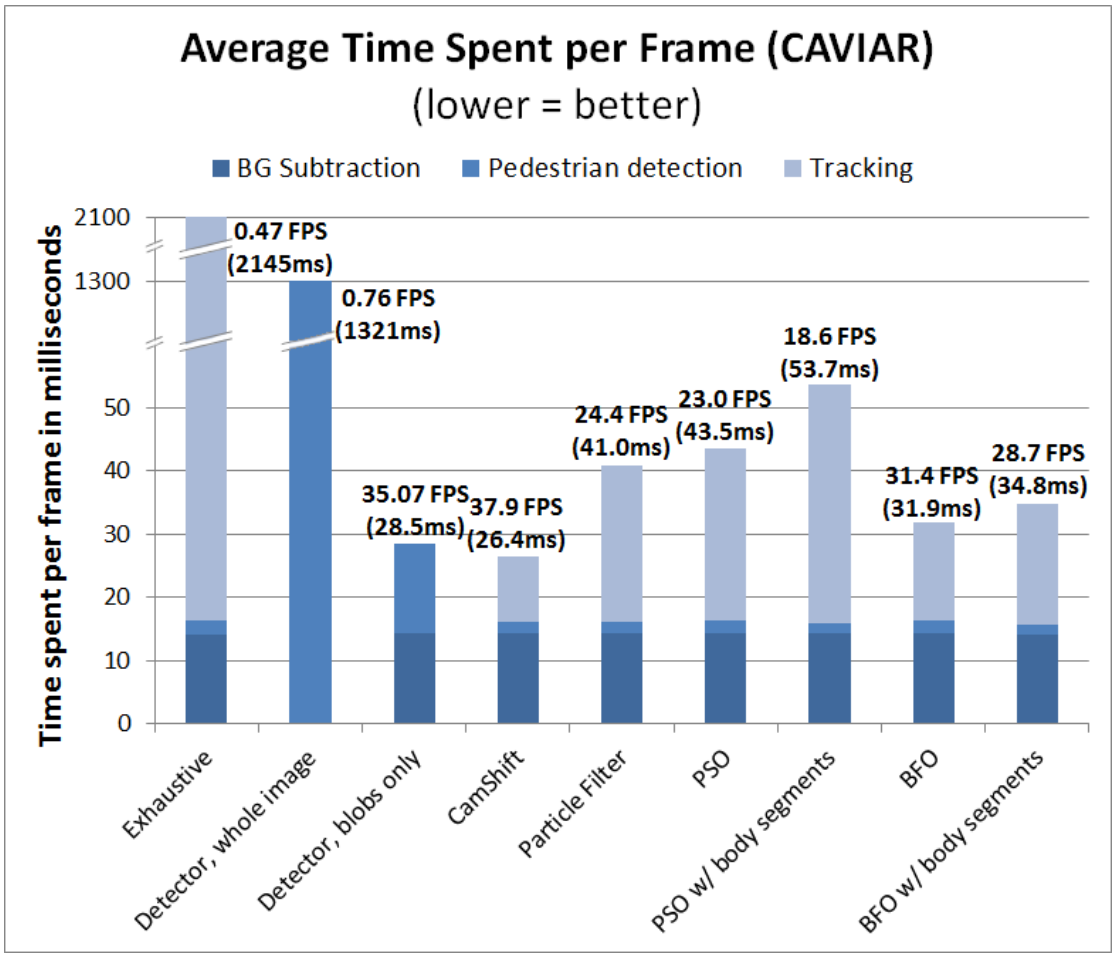


Figure 3.13: The entire system can be run in real-time on modest hardware (Intel Xeon E5345 2.33GHz). BFO performs the fastest amongst the trackers which perform on par with exhaustive search. Run times are averaged across all 26 corridor videos of the CAVIAR dataset and averaged over 30 runs per video.

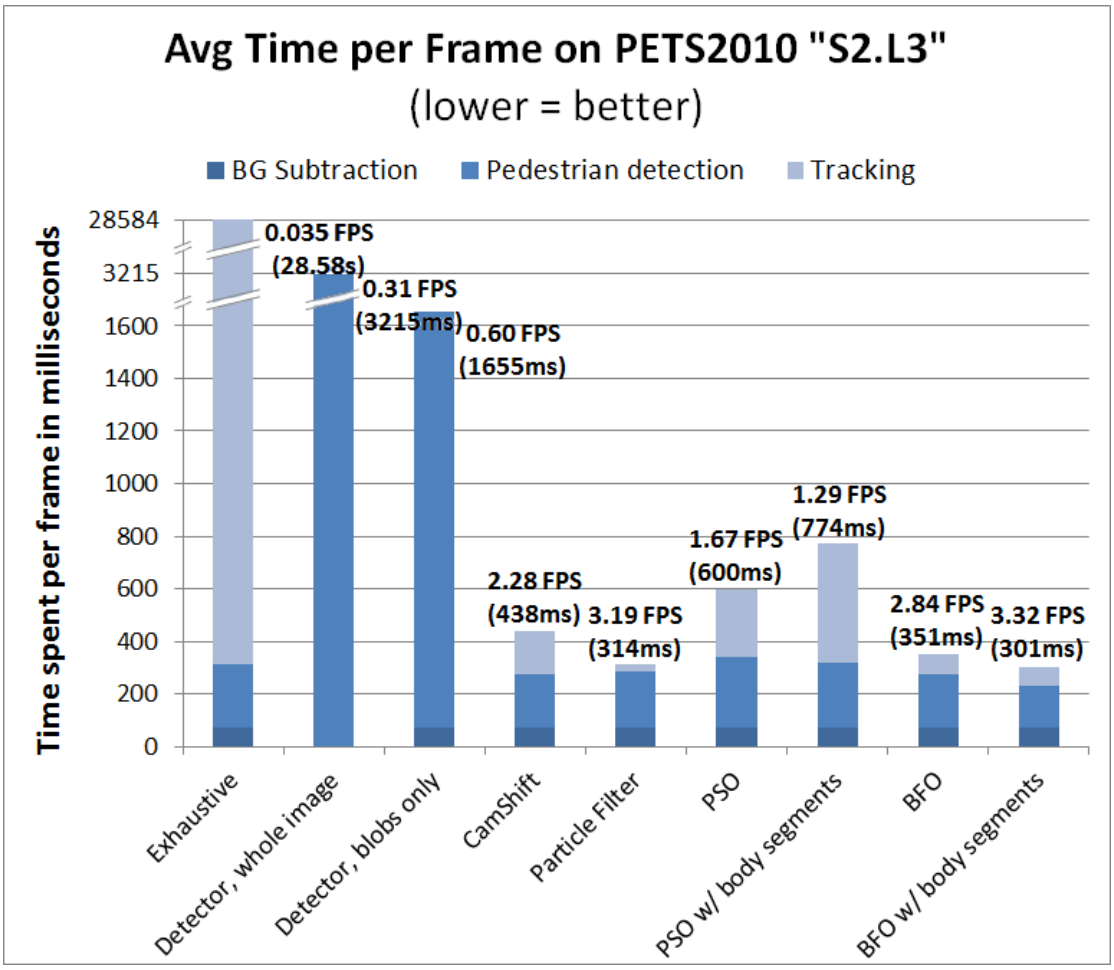


Figure 3.14: Even with multi-threaded tracking of each pedestrian, the number of pedestrians in the scene (over 40) saturates all available CPU resources (numbers are averaged over 30 runs on an Intel Xeon E5345 2.33GHz quad-core CPU).

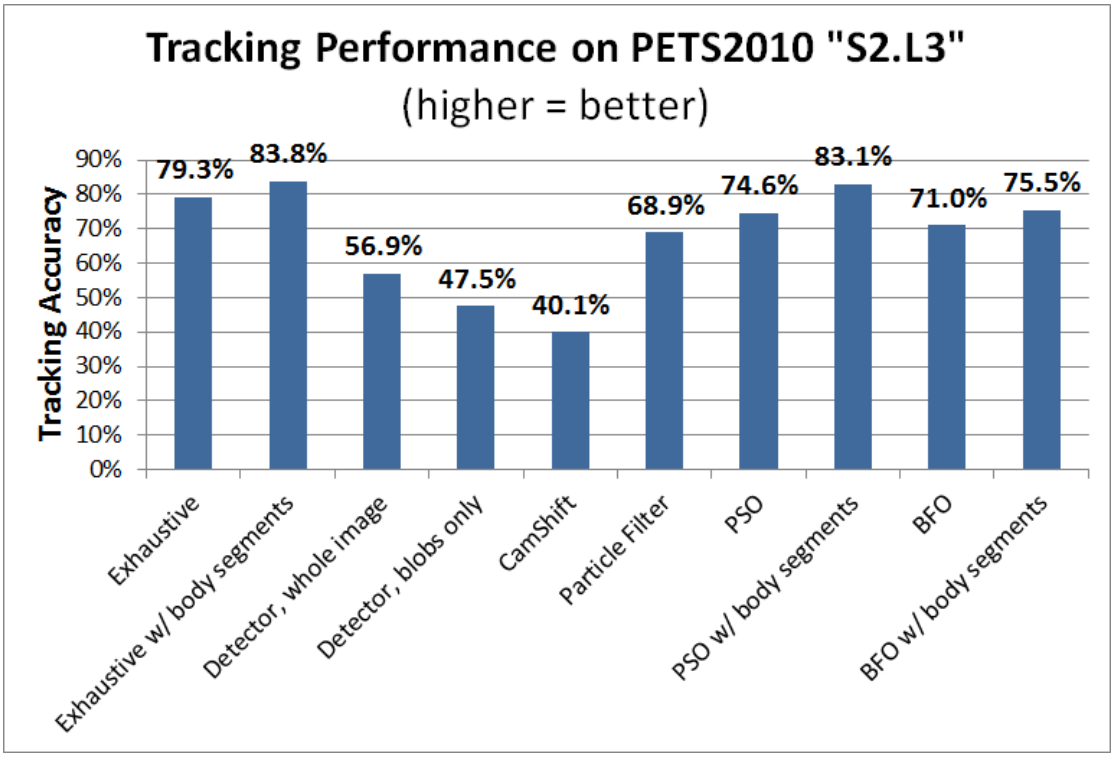


Figure 3.15: The swarm intelligence approaches perform slightly better than the same Particle Filter used on the CAVIAR videos, even without segmentation. Results are averaged over 30 runs on the video in Figure 3.8.

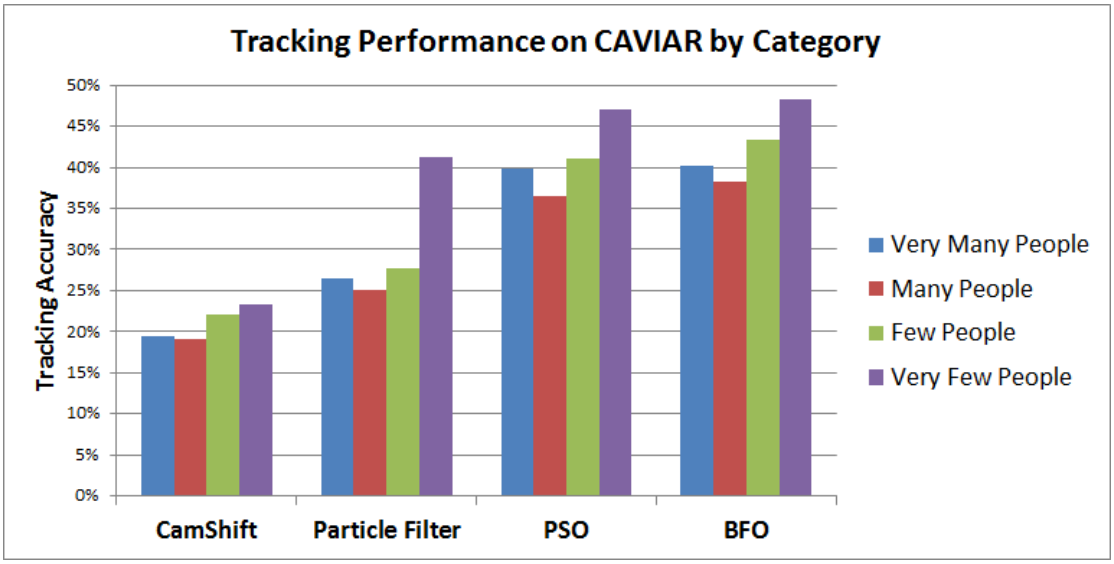


Figure 3.16: Comparison of tracking accuracy for each category of video of the CAVIAR dataset (see Table 3.2). Results are averaged over 30 runs for each video.

Chapter 4

Zombie Survival Optimization: A Search Optimization Framework Inspired By Zombie Foraging Behavior

Abstract

Search optimization algorithms have the challenging task of balancing between exploration of the search space (e.g., map locations, image pixels, parameter space) and exploitation of prior or learned information (e.g., statistics, regions of high fitness, model constraints). To address the challenge of automatically balancing resources between exploration and exploitation, we present a new algorithmic framework which we call Zombie Survival Optimization (ZSO), a novel swarm intelligence approach modeled after the hypothetical foraging behavior of zombies. Low-intelligence zombies (exploration agents) search in a space where the underlying fitness is modeled as a hypothetical airborne antidote which cures a zombie's aliments and turns it back into a human who is a high-intelligent agent and who attempts to survive by exploiting knowledge about the search space. Such an optimization algorithm is useful for tracking, such as searching an image for a pedestrian. Experiments on the CAVIAR dataset suggest improved efficiency over Particle Swarm Optimization (PSO) and Bacterial Foraging

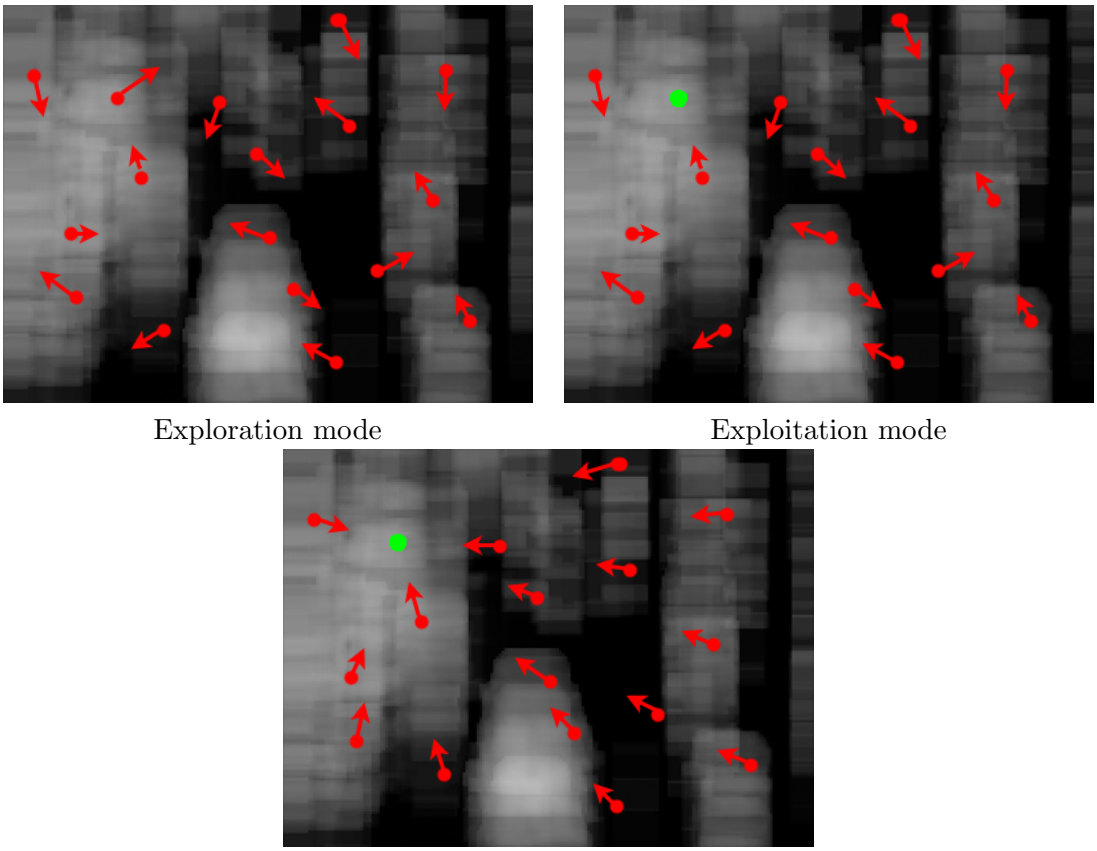


Figure 4.1: Agents automatically switch between 3 modes: random walking (search space exploration), survival mode (search space exploitation), and hunter mode (swarming).

Optimization (BFO). A C++ implementation is made available.

4.1 Introduction

Swarm intelligence is a family of decentralized stochastic algorithms inspired by the behavior of biological and artificial swarms. Taking cues from nature, the appeal of swarm intelligence algorithms comes from their ability to efficiently find near-optimal solutions in a simple distributed manner. Since standard particle filter implementations are outperformed by evolutionary computation methods such as Particle Swarm Optimization (PSO) [106], we set out to develop a framework which outperforms PSO and

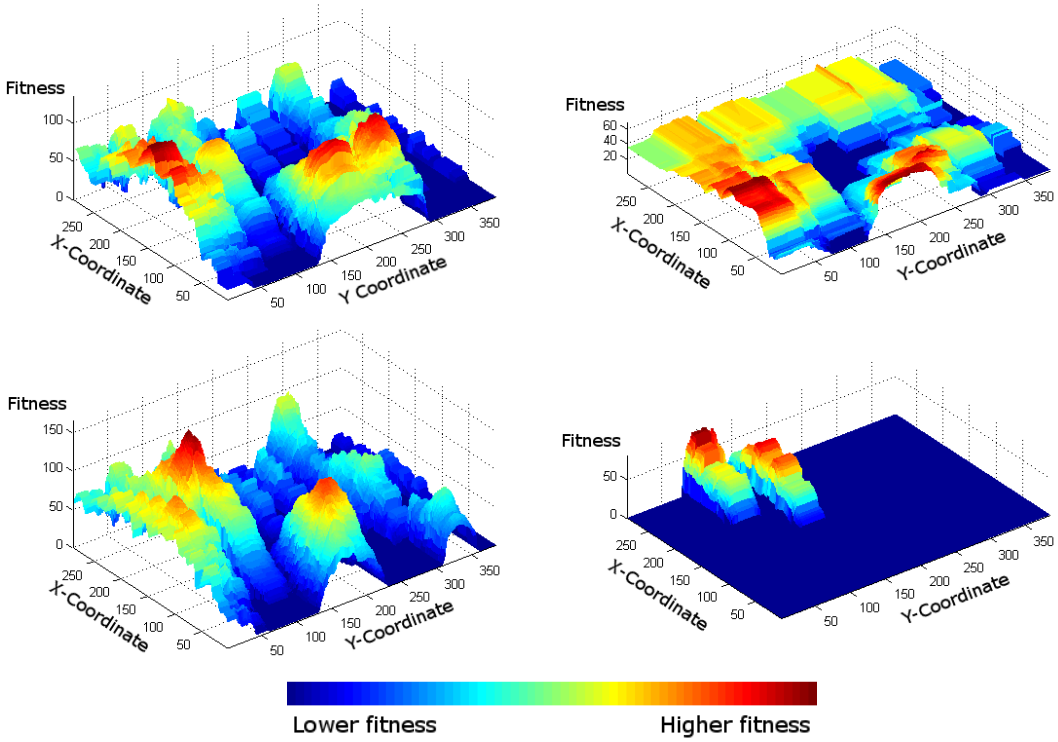


Figure 4.2: Four of the 1,843 fitness space images generated from the dataset using HSV signatures initialized on Viola-Jones pedestrian detections. Resolution is 384×288 with higher (red) locations representing high fitness and lower locations (blue) representing low fitness.

present a novel algorithm inspired by the foraging behavior of zombies.

The scenario is the following. A virus has broken out and a number of N zombies or agents occupy a search space. An airborne antidote has been dispersed into the search space in which the zombies randomly walk (exploration mode). The concentration of the antidote at any given location corresponds to the fitness function of that location (i.e., high fitness = high antidote concentration). Sufficient levels of antidote are able to "cure" any zombies who happen to breathe it in, turning the zombie agent into a human agent (exploitation mode). Other zombies sense these humans and attempt to catch them (hunter mode). Since high concentrations of antidote in turn can cure the chasing zombies, humans attempt to survive by exploiting the local fitness

Symbol	Definition
N	Number of zombies/agents
S	Speed of zombies, e.g., step size in pixels
V	Variance of zombie walk direction
A	Fitness threshold which "cures" a zombie
G	Max generations for stop condition
T	Target fitness for stop condition

Table 4.1: Definitions of symbols used in the Zombie Survival Optimization algorithm.

space, searching for the position of highest fitness which best barricades them from the impending zombies. Zombies which are able to successfully reach a human agent acquire sustenance and the resulting bite turns a human agent back into a zombie.

4.2 Related work

When we open our eyes, we do three things: 1) detection, 2) recognition, and 3) tracking. Tracking is a foundational problem in the field of computer vision which involves localizing the location of a target (e.g., object, pedestrian, point, feature) from one moment of time to the next, typically in a video sequence.

The most popular approach towards single-view tracking is performed using a particle filter [29, 100]. Swarm intelligence algorithms, on the other hand, have also been successful for tracking [63, 110]. include Particle Swarm Optimization (PSO), which is modeled after the swarming behavior of flocks of birds and schools of fish, [47], Bacterial Foraging Optimization (BFO), which is inspired by the feeding behavior of E. Coli bacteria, [76], and Ant Colony Optimization (ACO), an optimization algorithm designed after the stigmergy behavior of ants [38]. The ultimate goal of any of these optimization algorithms is to find the global best fitness as efficiently as possible. Since fitness/objective function calls are often the most resource-intensive component of an optimization algorithm, efficiency is often defined as using the least number of fitness

function calls as possible, i.e., fastest convergence to the global optimum.

Unique to ZSO is the automatic balance between exploration and exploitation, which is traditionally done via explicit parameters (e.g., PSO, BFO) or in a fixed manner (e.g., 90% exploration, 80% exploration). The ZSO algorithm automatically balances search agents between randomized exploration agents (zombies) and smarter exploitation agents (humans). Areas of higher fitness correspond to areas of higher antidote concentration, which thereby temporarily cures a zombie's aliments and encourages it to avoid other zombies and search for higher antidote concentrations.

4.3 Technical approach

Agents are initialized in the search space with a uniform distribution for random location and direction. There are three states or modes in which agents can be in:

1. **Exploration mode:** randomly exploring the search space
2. **Hunter mode:** actively hunting a human
3. **Human mode:** trying to exploit the local search space

Agents are initialized in the first mode which attempts to explore the search space. Table 4.1 defines all the symbols used in the algorithm.

4.3.1 Zombie exploration mode

While the exploration search function is generic, the default exploration function is defined as follows: at each time step, each zombie moves forward in a step of size S in the current direction with a variance of V. This variance adds noise to the system which aids in climbing out of local optima. Zombies continue in this manner until they

either 1) sense a human in which case they start chasing it, 2) stumble upon a position of high fitness and turn into a human, or 3) reach the boundary of the search space in which case they simply change direction.

4.3.2 Zombie hunter mode

Zombies are attracted to humans and chase after them. At each time step, if there are any humans, zombies will change their direction to face the nearest human and continue to stumble toward them. Stumbling consists of the same variation when walking as in exploration mode (i.e., zombies can't walk in perfect straight lines) with the exception that the general direction is continually reset to face the nearest human.

4.3.3 Human exploitation mode

Zombies can become humans if the fitness of their current position exceeds threshold A, which can be dynamically defined by the mean of the range R of fitness values so far:

$$A = \frac{max(R) + min(R)}{2} \tag{4.1}$$

Humans realize that they will be chased and attempt to find a local optimum near their current position. Knowing that this improves the odds of pursuant zombies turning into humans themselves, this offers them the best long-term strategy for survival. Like the exploration function, the exploitation search function is also generic but is defined as a local mean shift search [21].

The algorithm ends when either a target fitness value T is reached or a maximum number of generations G is reached, whichever comes first. Algorithm 3 details the full pseudo code for the ZSO algorithm. A C++ implementation can be downloaded at `http://www.cs.ucr.edu/~nthoang/zso/`.

4.4 Experimental results

4.4.1 Dataset

Experiments are performed on the CAVIAR [32] dataset (see Figure 4.4). 1,843 search spaces were generated from objects in the first 300 frames of the six recommended [91] corridor videos *EnterExitCrossingPaths1cor*, *OneStopMoveNoEnter1cor*, *ShopAssistant2cor*, *ThreePastShop1cor*, *TwoEnterShop3cor*, and *WalkByShop1cor* (see Figure 4.2). These fitness spaces were generated by automatically detecting pedestrians using a Viola-Jones head and shoulders detector [55, 97] and tracking the initialized signature on all the frames using the exhaustive search. These fitness spaces range in difficulty from simple to complex (see Figure 4.2).

4.4.2 Metrics and effect of parameters

Number of fitness evaluations is the number of function calls made to the fitness function (the lower, the better). Given a number of evaluations, the **best average fitness** is defined as the average of the best fitness of all 1,843 search spaces (the higher, the better).

Figures 4.5-4.8 demonstrate the effects of the parameters on algorithm performance.

4.4.3 Parameters

Experiments were performed on ZSO with 100 agents, 100 generations, 25 pixel step, 25 deg variance, 0.50 threshold. Experiments on PSO were performed with 50 agents and 200 generations. The experiments on BFO were perforumed using 10 agents, 500 reproductions, 10 swims, 5px step, 90% disperse. Parameters were individually optimized for each algorithm using a random subset and fixed for all remaining tests.

4.4.4 Results

Figure 4.9 shows the performance of ZSO, PSO, and BFO using the fixed parameters in Section 4.4.3. As the number of fitness evaluations increases, ZSO achieves higher performance over PSO and BFO, suggesting that ZSO may be better suited for finding the global optimum.

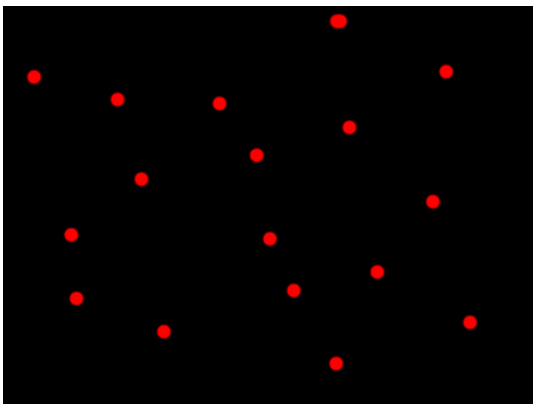
Despite its performance advantages, Zombie Survival Optimization exhibits some limitations compared to PSO, in particular:

- More parameters
- Discrete movement steps which may be sensitive to the search space
- A distance metric is required when determining which humans to swarm to

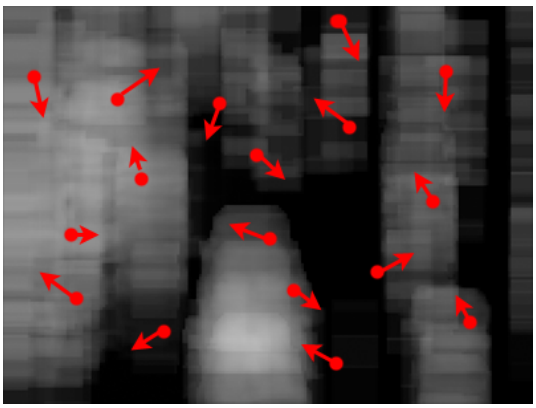
These limitations could be overcome by automatically setting some of the parameters (e.g., step size, fitness threshold) based on statistical calculations of the search space (e.g., increase step size if change in fitness is small, decrease step size if change in fitness is large). It is also possible to consider exploring the option of having zombie agents become attracted to any human agent instead of simply the closest one.

4.5 Conclusions

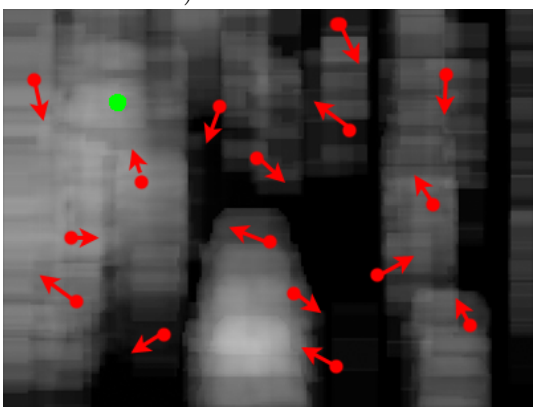
A new swarm intelligence algorithm modeled after the behavior of zombies vs. humans is proposed to address the challenge of balancing between exploration and exploitation for search optimization. Experiments on real-world multi-person tracking data show that it can be more efficient than BFO as well as PSO which in turn outperforms traditional particle filter [106].



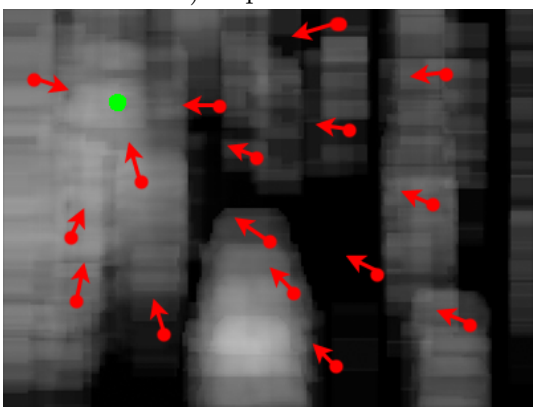
1) Initialization



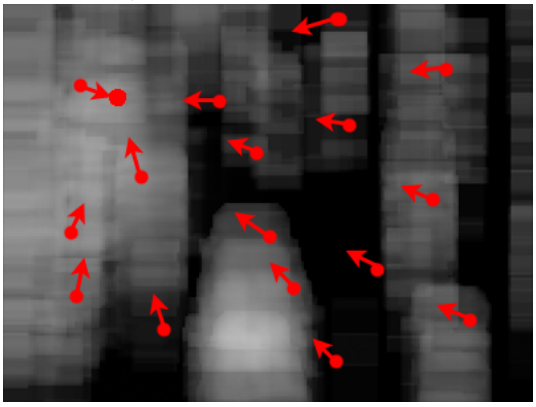
2) Exploration



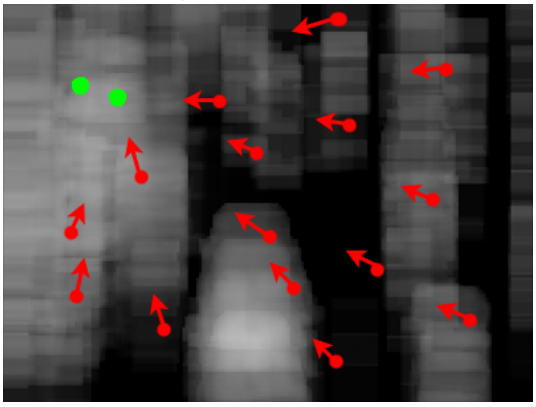
3) One agent is “cured”



4) Hunter mode



5a) Outcome: Defeat



5b) Outcome: Survival

Figure 4.3: 1-2) Initialize and explore. 3) One or more agents turn into humans (green) due to high antidote concentrations. 4) Zombies pursue humans. 5a-5b) Humans are either defeated or survive.

Algorithm 3 Zombie Survival Optimization (ZSO)

```
1: I ← image to search
2: procedure ZombieSurvivalOptimization
3:                                          ▷ Initialize N zombies in search space
4:     for all zombies z do
5:         z.location ← random point,
6:             e.g., ([0, I_width], [0, I_height])
7:         z.direction ← random direction,
8:             e.g., [0, 360) degrees
9:     end for
10:                                                ▷ Zombies hunt for humans
11:    for G generations do
12:        for all zombie z (asynchronous) do
13:            z.location ← z.location +
14:                (z.direction * V * S)
15:            f ← evaluate fitness at z.location
16:                                          ▷ Search exploitation mode (human)
17:            if f > threshold A then
18:                z.is_human ← true
19:                Gradient ascent search
20:                    of local neighborhood
21:                if any zombie z′ within S reach then
22:                                                    ▷ Bitten by zombie
23:                    z.is_human ← false
24:                end if
25:            else                                ▷ Exploration mode (zombie)
26:                if any humans exist then
27:                    Find closest human h
28:                    z.direction ← toward h
29:                end if
30:            end if
31:        end for                                          ▷ Zombie loop
32:    end for                                          ▷ Iteration loop
33:    return Location of best fitness
34: end procedure
```

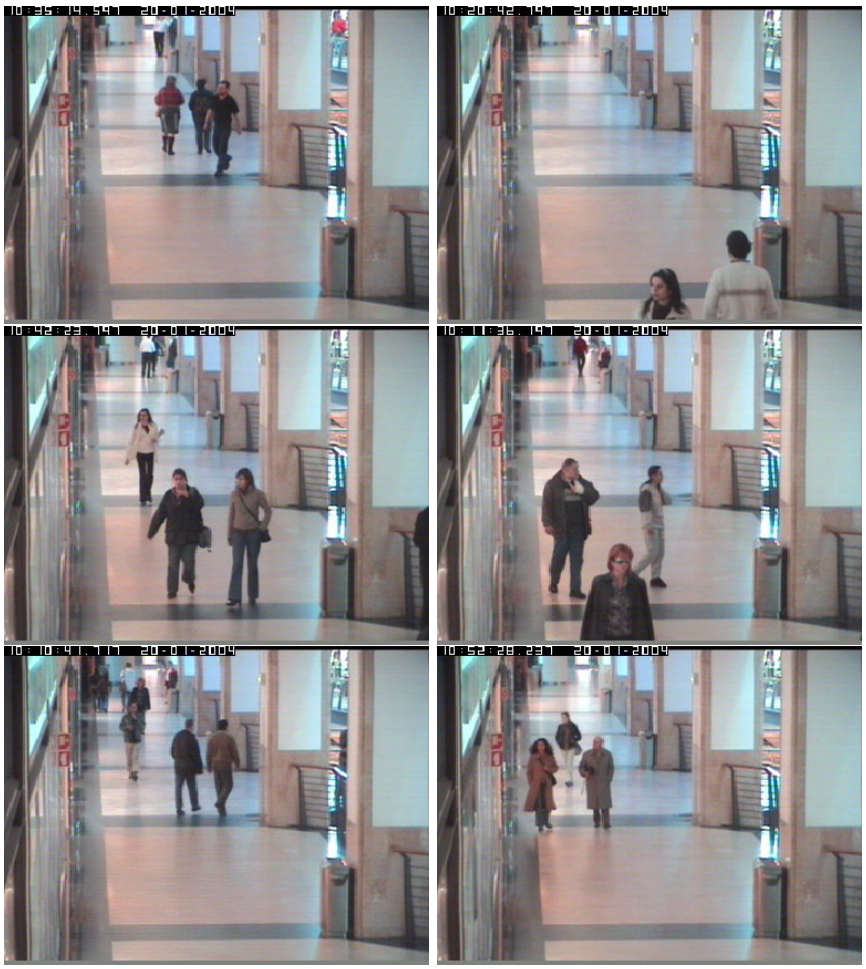


Figure 4.4: Samples frames from the CAVIAR dataset [32].

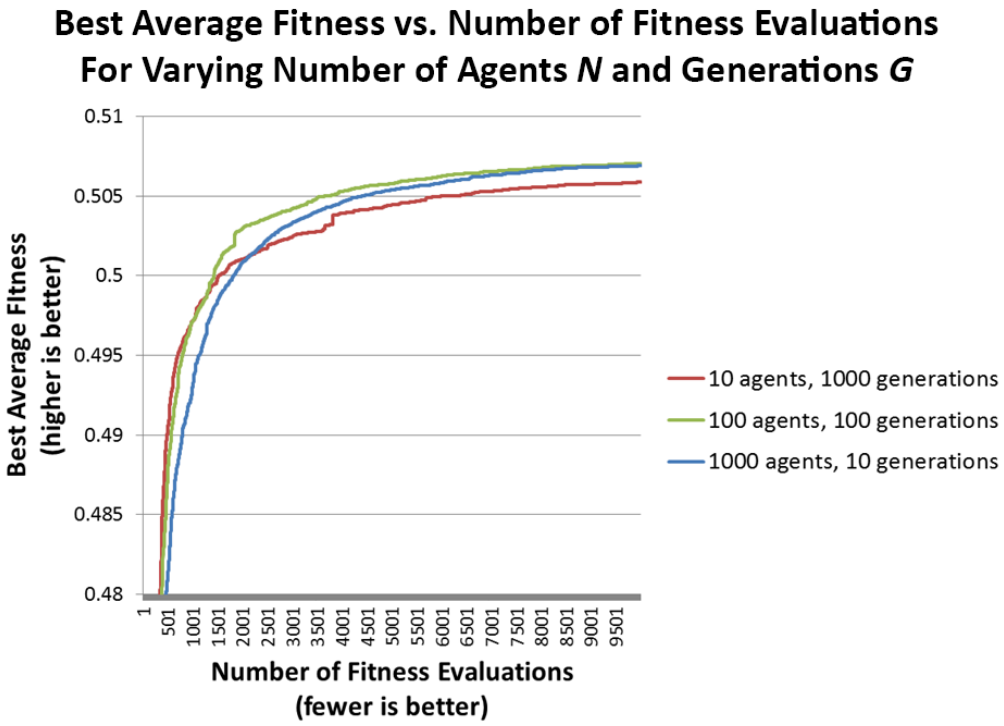


Figure 4.5: Effects of parameters N (number of agents) and G (number of generations). Adjusting these parameters can allow the number of total evaluations to remain constant (10,000 fitness evaluations in this case), while balancing the size of the swarm. Setting an even balance between the two achieves the best average performance.

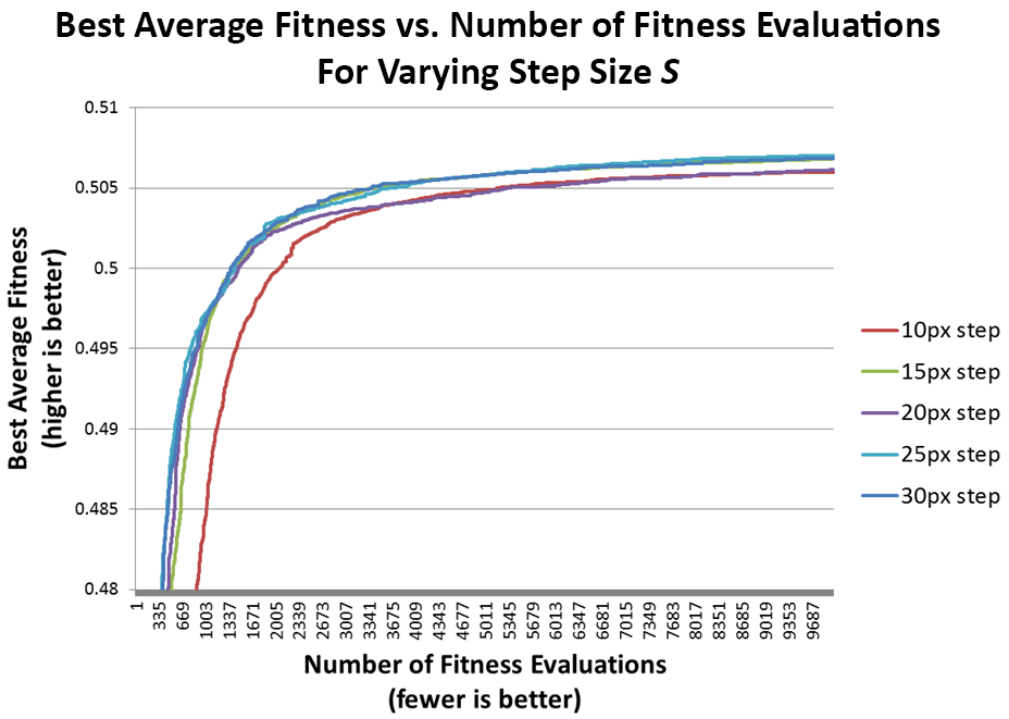


Figure 4.6: Effects of step size parameter S. This search space-dependent parameter may vary depending on how minute or spread apart fitness value changes may be in the underlying search space. In this case, the 30 pixel step size was suitable for the 384×288-pixel images.

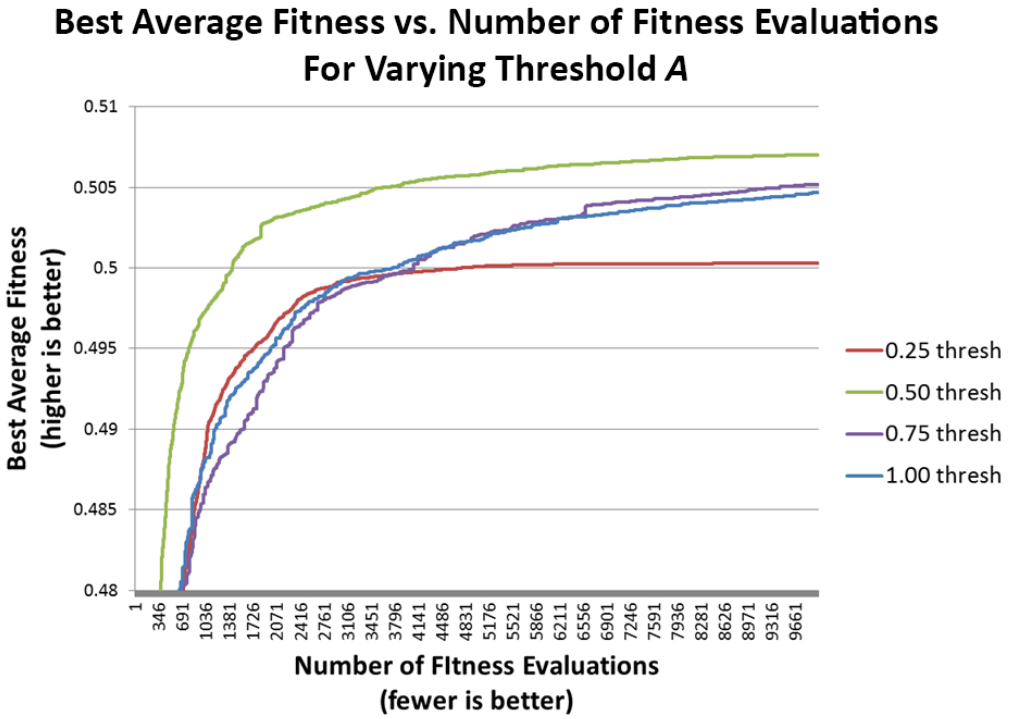


Figure 4.7: Effects of the zombie to human fitness threshold parameter A on best average fitness. Setting this parameter low results in earlier exploitation search while setting it higher favors exploration of the search space. A balanced threshold of 0.50 achieved the best performance, suggesting that exploration should be evenly balanced with exploitation.

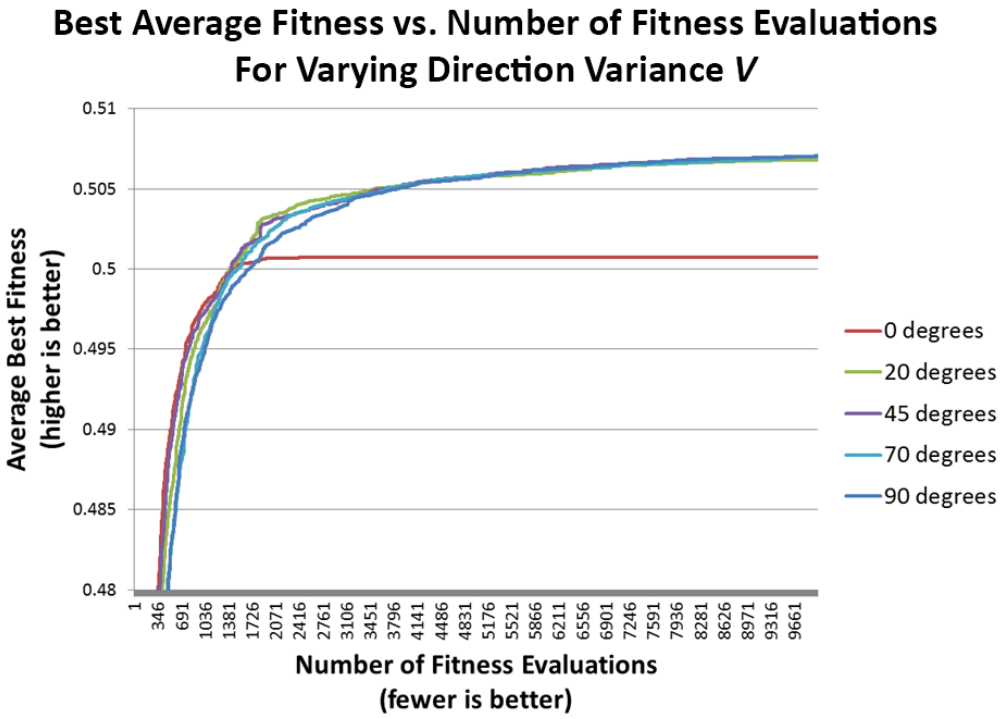


Figure 4.8: Effects of variance parameter V on best average fitness. The important thing to note is that adding randomness to the agents in their direction is more beneficial than moving in a straight line, e.g., the 0 degree variance.

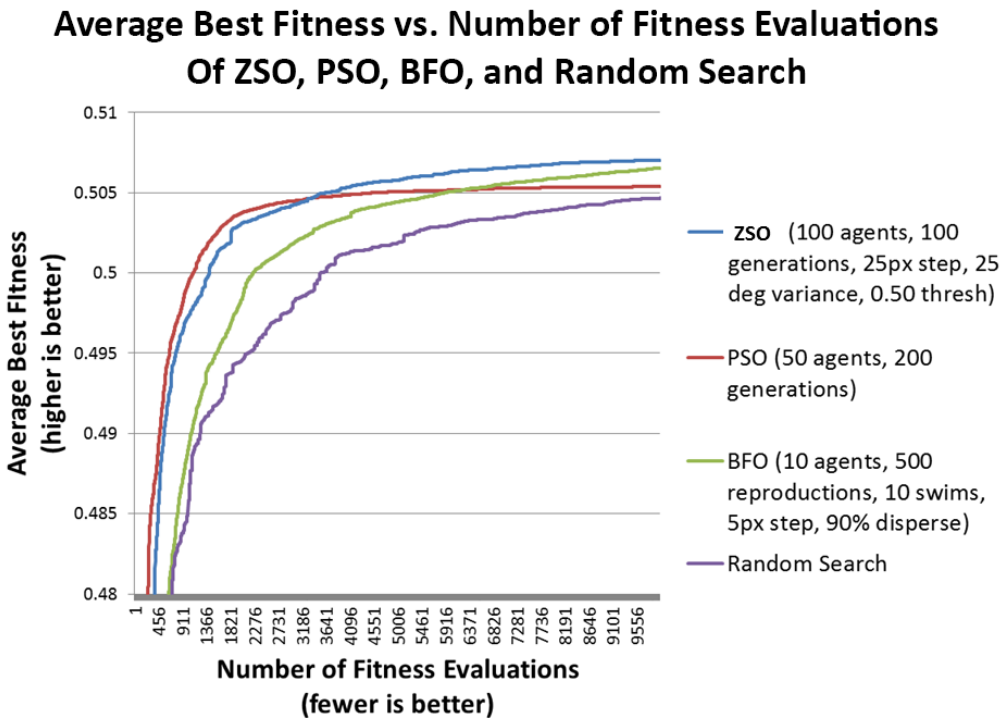


Figure 4.9: Experimental results comparing Zombie Survival Optimization with Particle Swarm Optimization, Bacterial Foraging Optimization, and Random Search. On the CAVIAR search spaces, ZSO is able to locate higher average fitness values over time.

Chapter 5

Multi-camera Appearance Signatures

Abstract

Tracking in an uncalibrated video network poses the challenging problem of determining whether or not a person or object seen in one camera is the same person or object seen in another camera. Pedestrians may appear in multiple cameras simultaneously or appear and reappear in other portions of the network at arbitrary intervals. In addition, the problem is made more challenging if the network's cameras are located in a variety of different environments (e.g., lit hallways vs. dark hallways vs. sunny courtyards). In order to approach the problem of multi-camera correspondence/recognition, we have devised a part-based appearance signature for identifying individuals.

5.1 Introduction

There are many challenges in developing a robust signature to best represent a pedestrian from one camera to another camera. Challenges include changes in pose of the pedestrian as they move in the video, point of view differences among cameras, varying distances between cameras and pedestrians, and lighting conditions. Logistical challenges include bandwidth constraints in sending the representations as well as time constraints in calculating the representation or signature. Without calibration data or

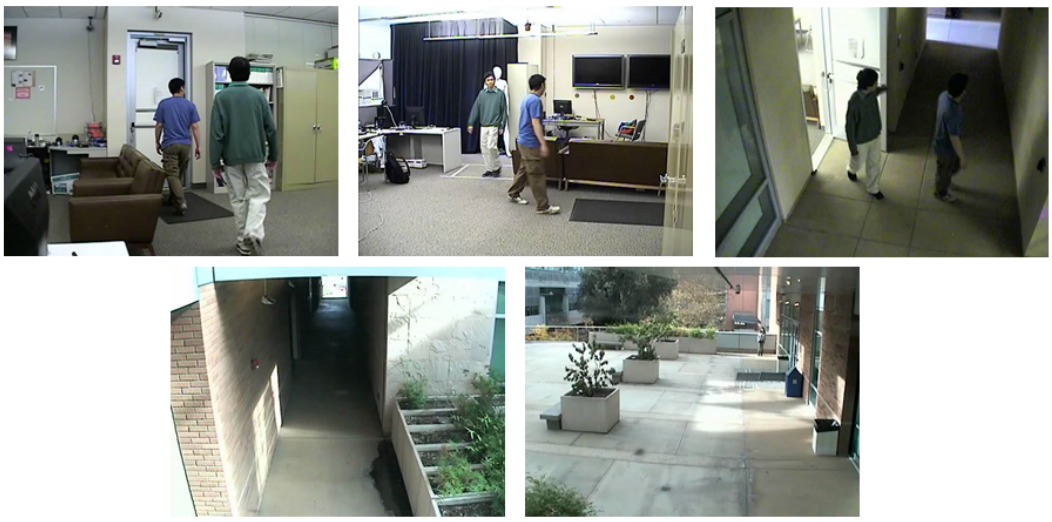


Figure 5.1: Sample dataset consisting of 5 cameras (2 indoor and 3 outdoor) were used to record 5 pedestrians. The dataset consists of 4000+ pedestrian ROIs for use in testing of the discriminative properties of the appearance signature representations.

information about the underlying layout of the video network, it is necessary to provide means for identification between targets in the network and an appearance signature is one such solution.

5.2 Technical Approach and Experiments

After obtaining an ROI of a pedestrian, the ROI is partitioned into 5 horizontal stripes based on body part proportions (see Figure 3.2). For each partition, all pixels in the foreground are analyzed to create the signature.

Color and texture features were selected as the two main components of the appearance signature. Since it is difficult to determine which color space is most suitable for use as an appearance signature, a wide array of color spaces were tested. The color spaces tested include: RGB, HSV, HSL, YUV YCrCb, YUV YIQ, CIE XYZ, CIE Lab, and CIE Luv. For the texture component of the signature, Local Binary Partitions were used with varying fixed and adaptive half-widths.

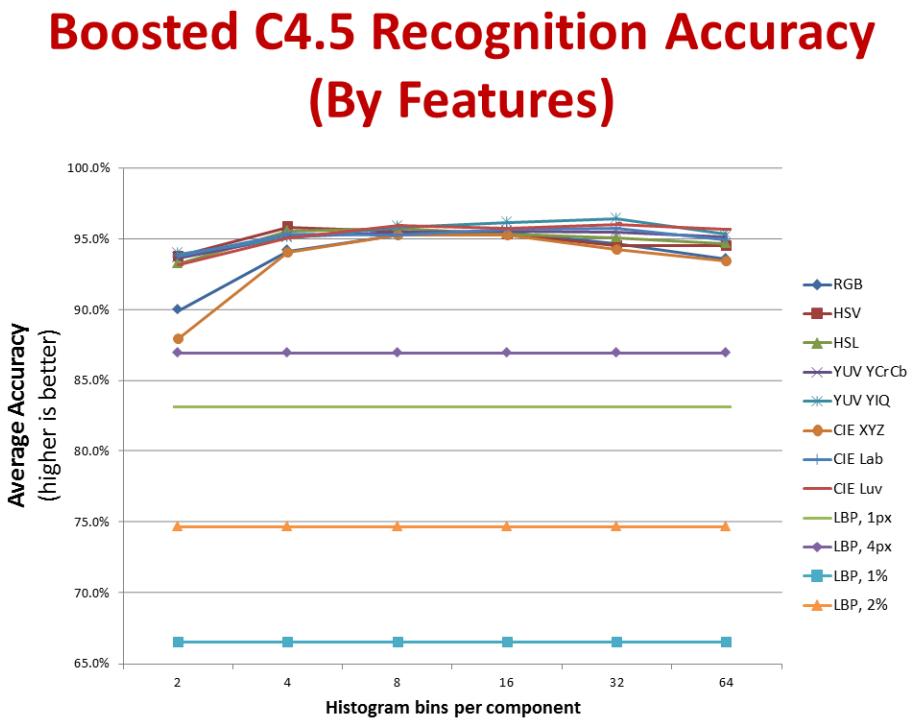


Figure 5.2: Recognition accuracy performance of all color and texture features. Color features far out-perform texture features when used alone.

A dataset consisting of 5 cameras (2 indoor and 3 outdoor) and 5 pedestrians was recorded (see Figure 5.1). A boosted C4.5 decision tree [80] was used to determine which color space provided by best overall discriminatory power.

5.3 Results

Figure 5.2 shows the performance in classifying all ROIs based on using individual features. Figure 5.3 shows a closer look at only the color features. Figure 5.4 shows the performance of signatures generated using each individual body part. Finally, Figure 5.5 shows the in-practice online performance of using the fused signatures when using nearest neighbor classification.

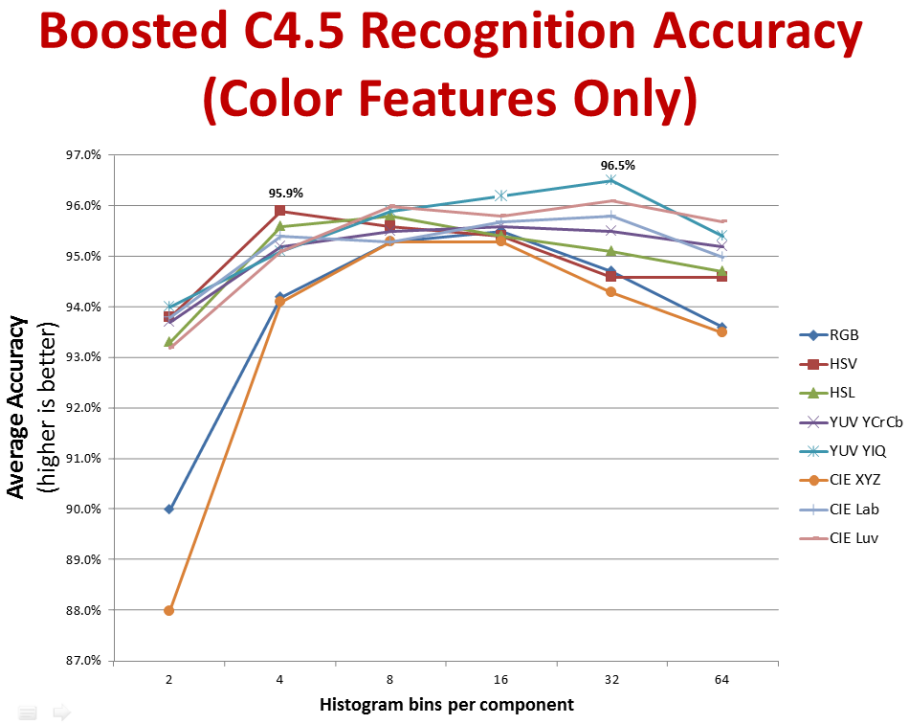


Figure 5.3: Recognition accuracy performance of all color features. HSV provides the best performance using 4 bins per HSV-component while an increase to 32 bins per component favors YUV YIQ.

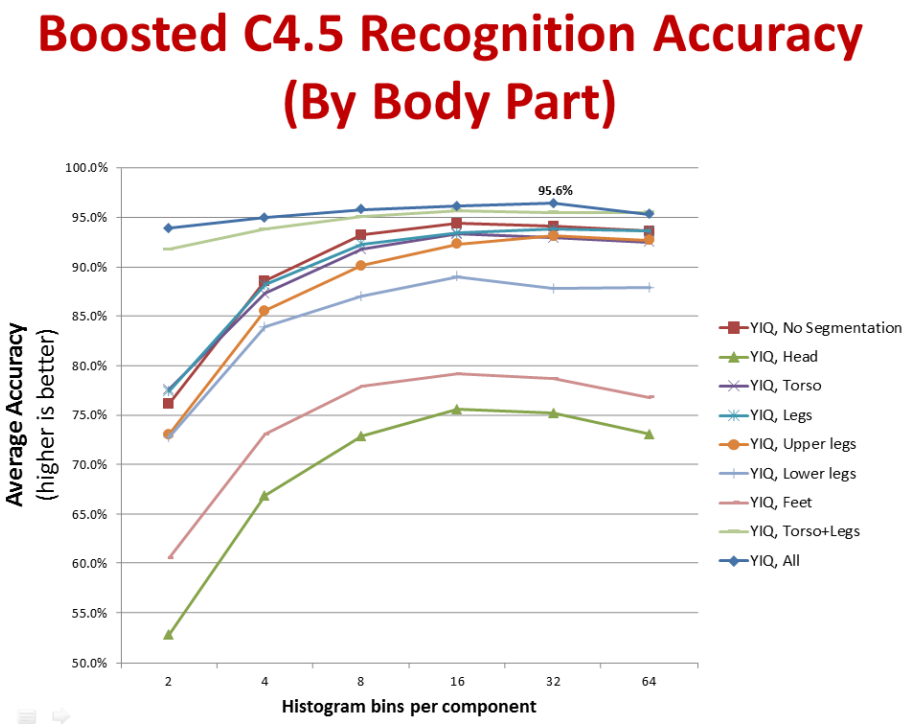


Figure 5.4: Recognition accuracy performance by body parts. Using all body parts as partitions provides improved accuracy over using all body parts as a single partition.

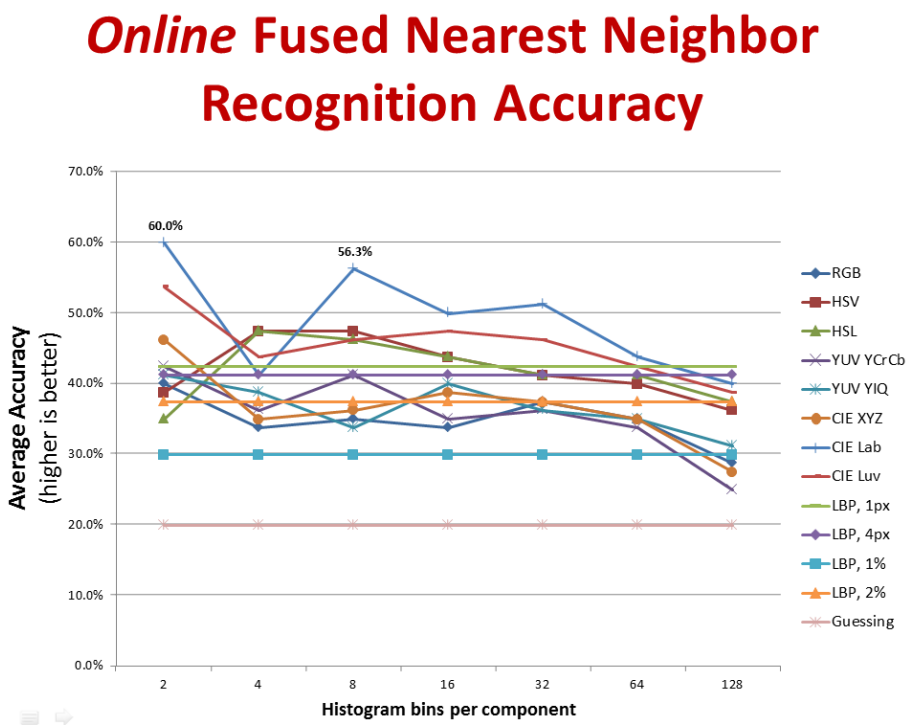


Figure 5.5: Online recognition accuracy using simple nearest neighbor comparison of histograms.

5.4 Conclusions

A multi-camera appearance signature was devised and deployed for use in a video network. It was found that color features outperform the tested texture features and that YUV YIQ signatures had the best classification performance when compared to all the other color spaces. In addition, using body partitions improved performance over using no partitions.

Chapter 6

Individual 3D Face Modeling

Abstract

3D face models are useful for a number of applications including human computer interaction (HCI), security, and entertainment. Automatically generating 3D face models from video networks is a complex task which faces many challenges such as low resolution, inconsistent lighting, and real-time constraints. In this paper, we use facial landmark points from multi-view video to morph a 3D model to fit a person's face. Automatic landmark detection is performed Constrainted Local Models (CLMs) [86]. 2D facial landmark points are then fused from multiple views and across multiple frames of video to reconstruct the 3D face model of pedestrians walking in the video network.

6.1 Introduction

A 3D morphable model approach is used to reconstruct the 3D shape of a person's head. The process can be broken down into the following steps:

1. **Image acquisition:** Acquiring the video images
2. **Face detection:** Detecting the location of the person's face in a frame
3. **Face alignment/pose estimation:** Locating the facial landmark feature points on the face and estimating the person's head pose

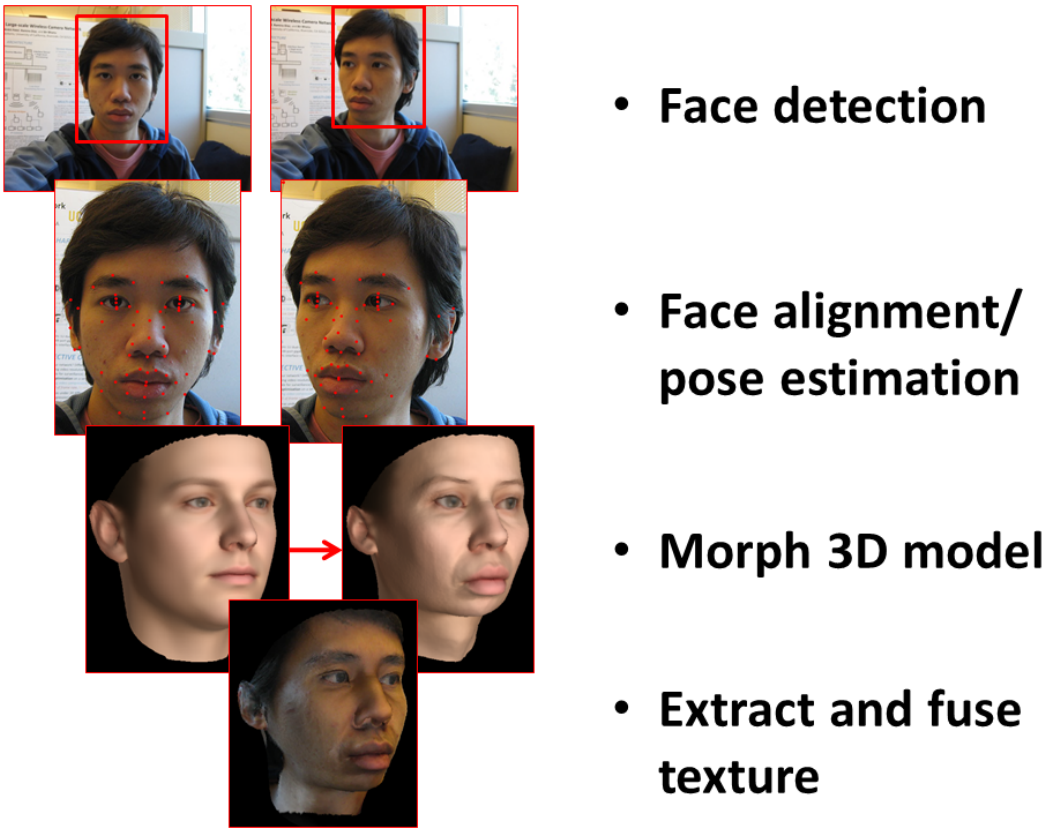


Figure 6.1: Overview of the proposed system. Once a face has been detected, face alignment is performed on it to recover landmark facial points. From these points, the pose is estimated and then used to morph a 3D model to fit the person's face.

4. **Morph 3D model:** Calculate the shape parameters to fit the 3D model to the 2D images

5. **Texture extraction and fusion:** Fusing the texture from multiple cameras into a single 3D texture

Figure 6.1 visualizes the steps of the approach.

6.2 Related Work

Besides morphable models [83] (deforming a generic model or 3D face "mask" to fit a 2D face image), other face modeling techniques include shape from silhouette [51] (calculating shape by estimating the pose and observing the contours of the face's sil-

houette), shape from shading [26] (modeling depth from the way the face is illuminated), and shape from motion [87] (inferring depth from spatial changes of landmarks or texture in response to change in pose with respect to the camera), as well as hybrid approaches which use any combination of the above methods [17, 84].

6.3 Image Acquisition and Face Detection

Images are captured from standard VGA cameras (i.e., 640×480 pixel resolution video). It is assumed that images from multiple cameras can be received at a central processing unit which performs the multi-camera face modeling. Face detection is performed using a Viola-Jones wavelet-based face detector [9, 97]. To improve performance, face detection can be performed exclusively on regions which are more likely to contain a person, such as the top portion of an object tracker or through motion detection.

6.4 Face Alignment and Pose Estimation

Face alignment involves detecting the location of various feature points or facial landmarks in a 2D image. Examples of such landmarks include the Farkas feature points [28], the MPEG-4 Facial Definition Parameters [79], or Active Appearance Models (AAMs) [99]. Facial landmarks are automatically located for use in the experiments using Constrained Local Models (CLMs) [86].

Figure 6.2: Automatic landmark detection done in real-time using Constrained Local Models (CLMs).

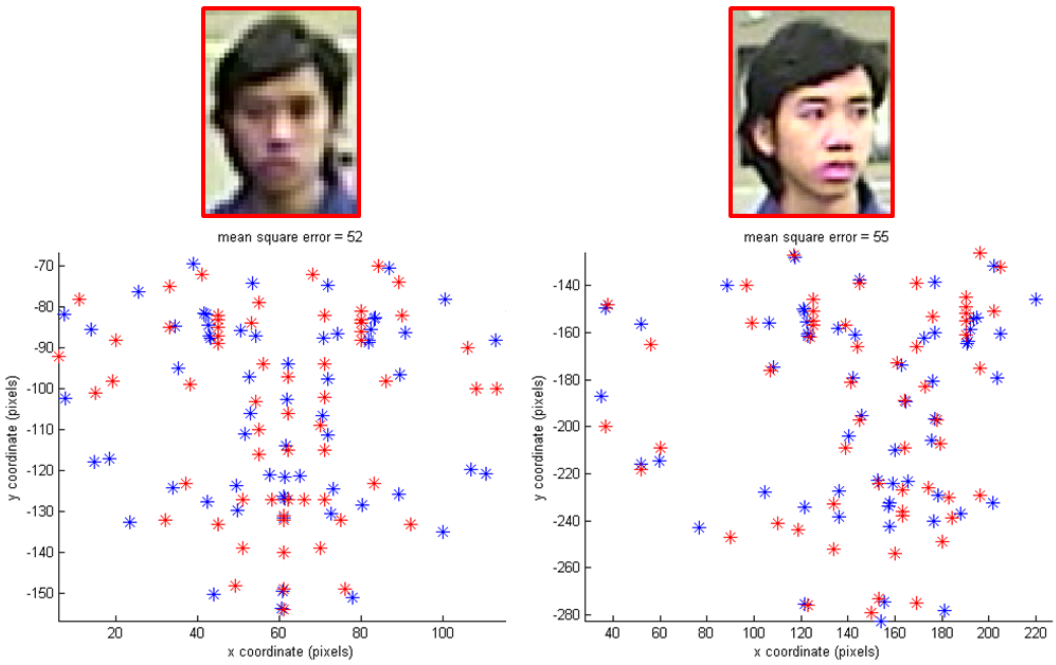


Figure 6.3: Given a set of landmark facial points, the pose of the head is estimated by minimizing the mean square error of the landmark points with the projection of the corresponding points on the 3D generic model. Red points represent the landmark facial points of the image while blue points represent the projected 3D model points.

Pose estimation is determining the orientation of the person's head in a 2D image. Pose can be represented as 7 variables: ϕ (roll), θ (pitch), ψ (yaw), $scale_x$, $scale_y$, $translation_x$, $translation_y$, where ϕ, θ, and ψ are the Euler rotation angles (about the X, Y, and Z axes, respectively), $scale_x$ and $scale_y$ are the percentage of scaling along the X and Y axes, and $translation_x$ and $translation_y$ are the offsets which will translate a face centered at [0,0] to the position seen in the image. Pose is estimated for each image as follows:

$$scale_x = \frac{\text{range of } x \text{ values of 2D landmarks}}{\text{range of } x \text{ values of respective projected 3D landmarks}}$$

$$scale_y = \frac{\text{range of } y \text{ values of 2D landmarks}}{\text{range of } y \text{ values of respective projected 3D landmarks}}$$

$$translation_x = mean(\text{ range of } x \text{ values of 2D landmarks })$$

Figure 6.4: The Basel Face Model (BFM) [77]. Left: mean face of 200 subjects. Right: same face morphed using the "age" set of parameters.

Figure 6.5: Sample camera environment consisting of three 640 × 480 pixel resolution videos from three different views of the same room.

$$translation_y = mean(\text{ range of } y \text{ values of 2D landmarks })$$

ϕ and θ are determined by least squares regression by minimizing the distance of the landmarks to the orthogonal projection of the mean face. If landmarks for both eyes are available, rotation about the Z axis is calculated as:

$$\begin{aligned} \psi &= \text{angle of line segment connecting the center of the eyes to } x \text{ axis} \\ &= \frac{180}{\pi} \times \tan^{-1} \frac{|\text{ right eye }_y - \text{ left eye }_y|}{|\text{ right eye }_x - \text{ left eye }_x|} \end{aligned}$$

Otherwise it is calculated in the same manner as ϕ and θ.

6.5 Calculating 3D Shape Parameters

6.5.1 The 3D generic model

The 3D morphable model approach to face modeling requires a 3D generic model of the human head. The recently proposed Basel Face Model (BFM) [77] is used to satisfy this requirement, though it is possible to create a model from scratch using a 3D scanner and a large number of volunteers. Limitations to using this model involves expression, as all face scans are in a neutral expression.

The Basel Face Model (Figure 6.4) is comprised of two primary components: the mean face of 200 subjects (100 males, 100 females, mostly European, ages 8-62) and 199 basis vectors/principal components computed by Principal Component Analysis (PCA). The model contains 53,490 vertices and is morphed as follows:

$$\text{Morphed Model} = \begin{bmatrix} Mean_{x0} \\ Mean_{y_0} \\ Mean_{z0} \\ \vdots \\ Mean_{x53490} \\ Mean_{y53490} \\ Mean_{z53490} \end{bmatrix} + \begin{bmatrix} Basis_{0,x_0} & \cdots & Basis_{198,x_0} \\ Basis_{0,y_0} & \cdots & Basis_{198,y_0} \\ Basis_{0,z_0} & \cdots & Basis_{198,z_0} \\ \vdots & \ddots & \vdots \\ Basis_{0,x_{53490}} & \cdots & Basis_{198,x_{53490}} \\ Basis_{0,y_{53490}} & \cdots & Basis_{198,y_{53490}} \\ Basis_{0,z_{53490}} & \cdots & Basis_{198,z_{53490}} \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ \vdots \\ s_{198} \end{bmatrix}$$

where S is a vector of 199 real-valued scalar coefficients which morph the model.

6.5.2 Fitting the 3D model

A projection method is required to model the relationship between the 3D face points to the 2D image. Orthogonal projection is used for its simplicity. This is a reasonable approach for uncalibrated cameras assuming that the change in depth between facial landmarks on the person's face is negligible in comparison to the distance

Figure 6.6: 13 face images extracted from the 3 videos. The faces range from 20px to 90px in height.

from the person to the camera and is typical for a camera network in which the cameras are relatively far from the subject physically. A 3D face point $[x; y; z]$ can be projected as a 2D point $[x'; y']$ on the image as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \end{bmatrix} \cdot \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_0q_1+q_2q_3) & 1-2(q_1^2+q_2^2) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix}$$

where s_x, s_y, t_x, and t_y are the respective scale and translation components of the estimated pose, and ϕ, θ, and ψ are the pose angles. That is, the projected 2D point is calculated by rotating the original 3D point to the appropriate pose, dropping its

Z coordinate (orthogonal projection), scaling it along the X and Y axes, and adding the translation component. The Euler pose angles are converted into a quaternion and then converted into a single rotation matrix. Quaternions are used to allow future work on optimizing the pose parameters themselves, in which case the smooth interpolation (e.g., spherical linear interpolation [90]) available to quaternions can be used.

In order to quantify the distance between the fitted model and the face alignment landmarks, mean square error is used on the Euclidean distance from the projected points of the fitted model to the 2D landmarks seen in the image. That is, for a single image with n landmarks, the mean square error is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(landmark_{i,x} - projected_{i_x})^2 + (landmark_{i,y} - projected_{i,y})^2$$

where landmark$_{i,x}$ and landmark$_{i,y}$ are the X and Y values of the i-th landmark and projected$_{i,x}$ and projected$_{i,y}$ are the respective 2D coordinates of the projected fitted model. For m multiple frames, this is extended to a weighted distance which takes into account the confidence of the landmarks in each image:

$$distance = \sum_{j=1}^{m} imageconfidence_j \times MSE_j$$

where $imageconfidence_j$ is a [0,1] real-valued scalar which quantitatively reflects the relative importance of each image and MSE is the mean square error of image j. For reconstruction, the confidence of image j is calculated as a function of image resolution:

$$imageconfidence = \frac{face_{width} \times face_{height}}{max_{width} \times max_{height}}$$

where max_{width} and max_{height} are the maximum width and height of the source image (in our example, 480 and 640 pixels, respectively). This allows higher-resolution images of a person's face to receive more weight in determining how to fit the 3D model. This

is a reasonable formulation given the assumption that the overall image quality of all images is equal (e.g., homogeneous mixture of cameras), though may be reformulated should sensors be weighted on their image quality (e.g., outdoor IR camera vs. high-resolution DSLR camera). Since the confidence value places more weight based on resolution, it is important to avoid utilizing digital zoom on the source video sensor which may artificially inflate the image's resolution without providing actual resolution gains.

Since the principal components of the model are independent, each coefficient (a real-valued scalar) is individually optimized to fit the landmarks by minimizing the distance. The curve of the distance function is estimated using weighted least squares regression; the derivative of this function is set to zero to find the shape parameter coefficient which minimizes the distance. Refinement of the 3D model from successive video frames is achieved through a closed feedback loop which feeds the current model as the starting point for model fitting in the next frame.

6.6 Extracting and Fusing Texture

Texture extraction is performed by projecting the 3D model onto the 2D image and sampling the RGB values of the image pixel. The primary challenge for a camera network lies in fusing the texture from multiple images (both from different camera perspectives as well as from sequential frames in the same video) to form a single texture. As a starting point, a weighted average technique is used which calculates a 3D point's RGB value as the average sampled pixels from the contrast-normalized images, using image confidence as the weight. This approach is planned for replacement with a more sophisticated local-alignment technique from the super resolution literature [103].

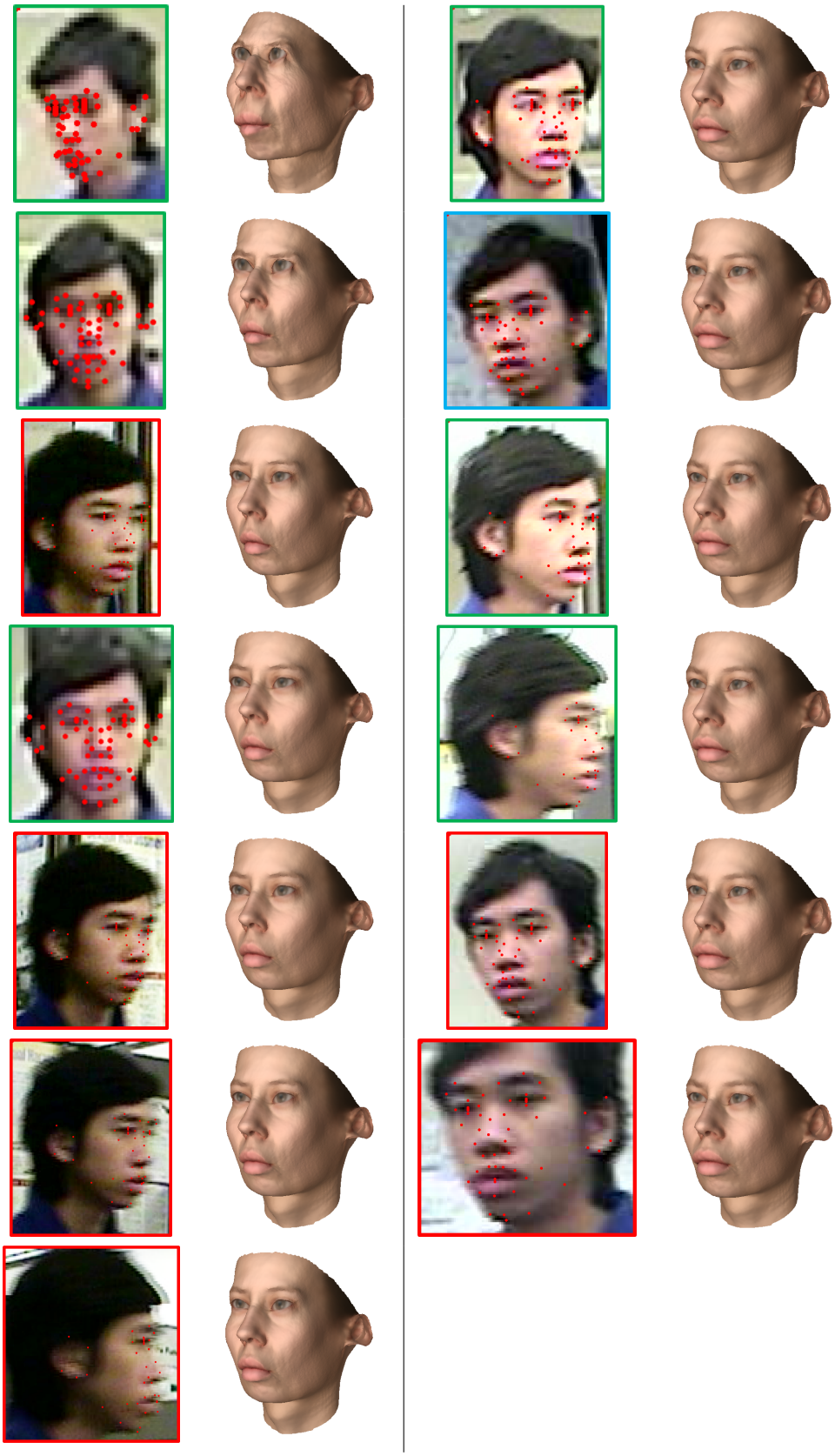


Figure 6.7: The morphed 3D model improves as more frames are added to it. Landmark points in these faces are manually located.

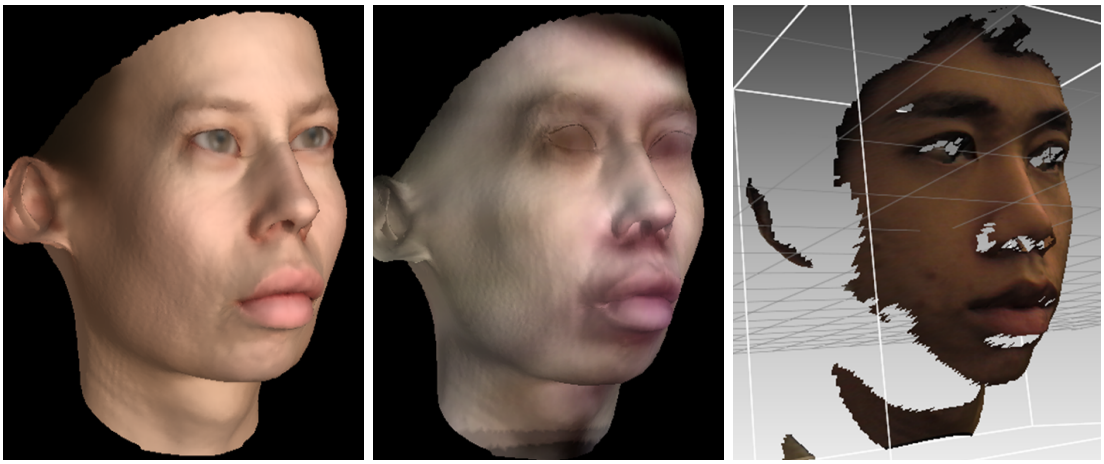


Figure 6.8: Comparison of morphed 3D model with groundtruth. Left: Morphed 3D model using the proposed approach. Center: Morphed model with fused texture. Right: Groundtruth 3D face scan using a Minolta laser scanner.

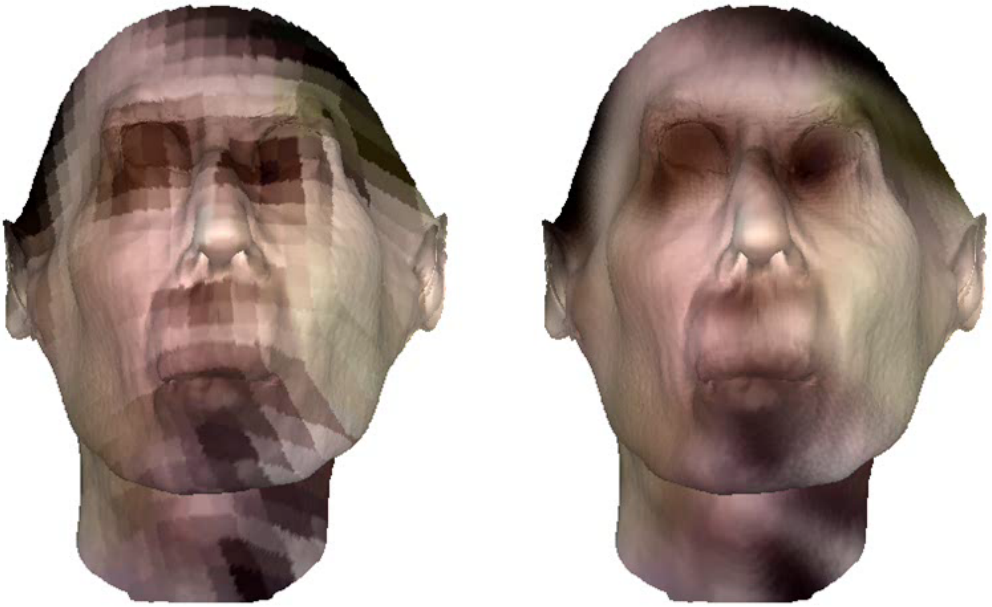


Figure 6.9: Bicubic interpolation is used to normalize all images to the same resolution before fusing. Left: Without interpolation. Right: With interpolation.

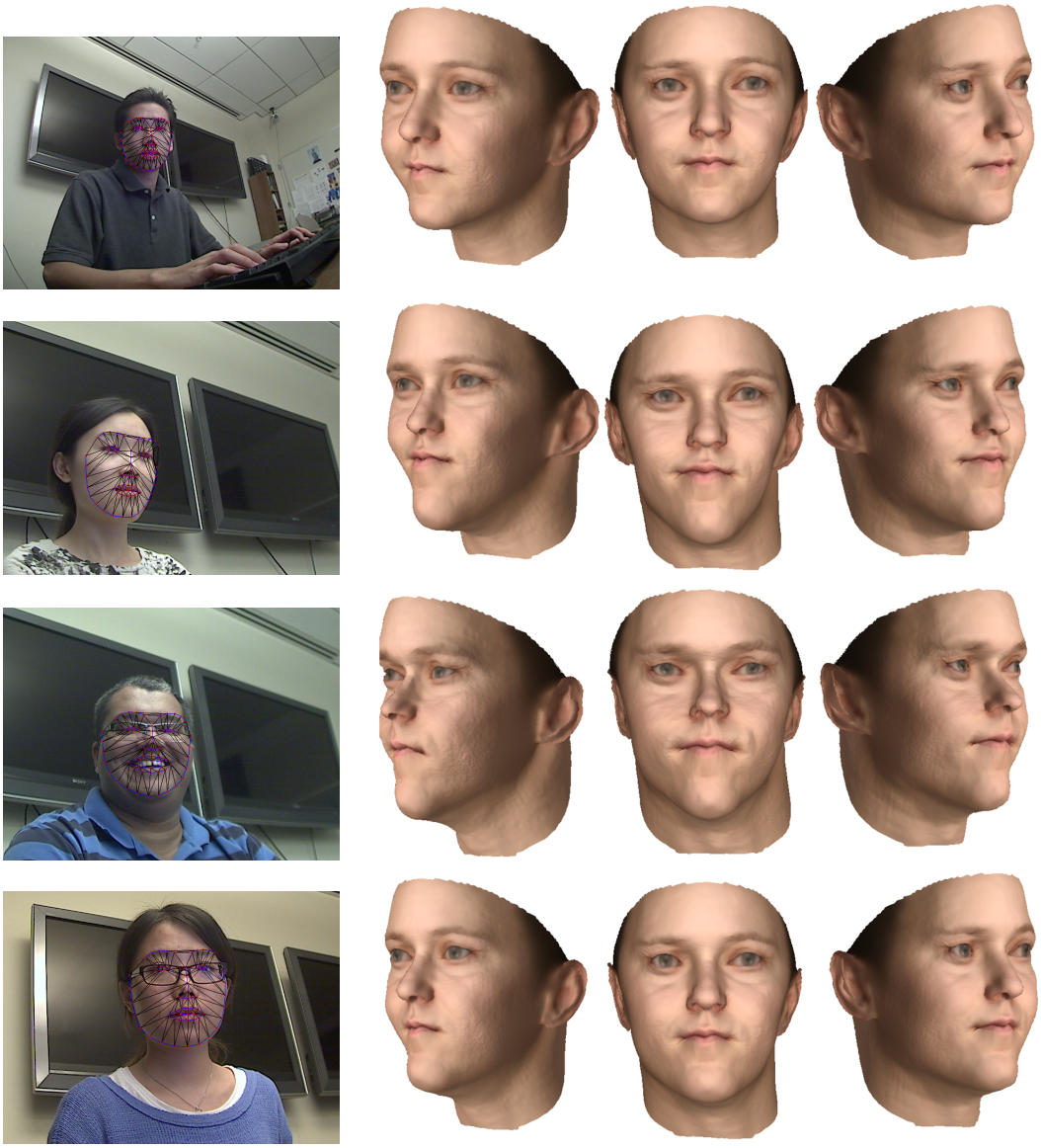


Figure 6.10: Automatically reconstructed 3D faces from video. Facial landmarks from the MPEG-4 Facial Definition Parameters [79] are detected using Constrained Local Models [86] and used to morph the Basel Face Model [77] to fit the person's face.

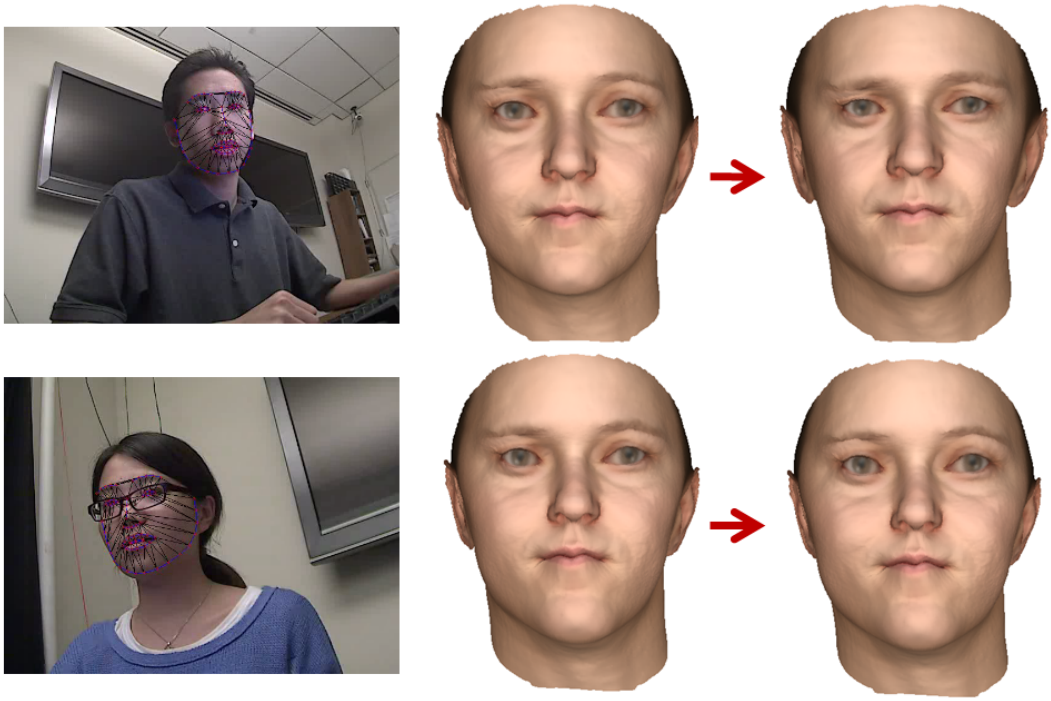


Figure 6.11: Using a gender detector to add shape priors to the reconstructed models.

6.7 Experimental Results

6.7.1 Dataset

Three videos are recorded at 640 × 480 resolution (Figure 6.5). 13 frames of these videos contain images of the same person, with face sizes ranging from 20px to 90px (Figure 6.6). A subset of 30 landmarks from the MPEG-4 Facial Definition Parameters [79] is used for 3D reconstruction from the 3D face alignment.

6.7.2 Results

Figure 6.9 shows the benefit of using bicubic interpolation to normalize the resolution of all images before fusing the texture. Normalization reduces the impact of pixelation from the low-resolution textures. Figures 6.7 and 6.8 shows the final morphed result with manually-located facial landmarks, both with and without texture, as well

as compared to an actual groundtruth 3D laser scan of the same individual. Figure 6.10 show the reconstructed 3D face results from using automatic facial landmark detection.

Using the FERET Dataset [78], a gender detector using on Fisherfaces [9] can be trained and used to add gender priors to the shape parameters, thereby improving the model's appearance (Figure 6.11);

6.8 Conclusions

A framework for automatic 3D face reconstruction is proposed using Active Appearance Model landmark facial points from multiple camera views to morph a generic 3D face model to the person's face. Results show the process can be completely automated and perform at near real-time speeds with satisfactory results, especially given the low VGA resolution of the input video.

Chapter 7

Conclusions

7.1 Building a Video Network

In order to facilitate research on multi-sensor systems, a 37-camera video network was designed and installed at the University of California, Riverside. The network consists of 37 outdoor pan/tilt/zoom (PTZ) cameras located in order to provide full coverage of the entire 2nd floor of the Winston Chung Hall engineering bulding, 16 indoor static cameras, and 3 camera-equipped mobile robots which are also connected to the network.

Multi-objective optimization was performed on the network's camera settings to gain insight into the benefits and drawbacks of optimizaing specific performance parameters (e.g., resolution, frame rate) toward specific tasks (e.g., tracking vs. recognition).

7.2 Tracking in a Video Network

Toward single-camera tracking in a video network, swarm intelligence algorithms were studied in depth to provide the foundational tracking algorithms used by the system for pedestrian tracking. Pedestrians are automatically detected, identified with a color and texture signature, and tracked using Bacterial Foraging Optimization (BFO).

In addition, a new swarm intelligence-based optimization algorithm, Zombie Survival Optimization (ZSO), was developed to provide improved speed and accuracy performance over previous swarm intelligence algorithms.

7.3 Multi-camera Recognition in a Video Network

To facilitate cooperation between cameras, multi-camera tracking was performed by creating normalized cross-camera pedestrian signatures based on color and texture. The signature uses a partition-based system based on statistical proportions of body parts in order to achieve generalization of pedestrian appearances across multiple views and lighting conditions which may be faced by the video network.

7.4 3D Model Reconstruction in a Video Network

Once a pedestrian has been located, face detection is performed on the pedestrian. If the pedestrian's face is visible from a frontal point of view, a Constrained Local Model (CLM) tracker is initialized on the person's face to locate facial landmark features. Pose estimation is perfored on these landmarks to determine the approximate orientation of the person's head.

Using the multi-camera signatures already generated, multiple sets of landmark points, either from consecutive video frames in a single sensor or from multiple sensors, are fused together to morph a 3D model. The 3D face model is reconstructed using a weighted system giving more weight to higher-resolution images of the person's face. After resolution normalization, the textures of the face are fused together to extract the final texture of the 3D model, thus finalizing the 3D face reconstruction. Additional face images and landmark points can incrementally be added to the reconstructed model in order to improve the texture of the model as well as refine the 3D shape parameters.

Bibliography

[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury. A survey on wireless multimedia sensor networks. *Computer Networks*, 51(4):921–960, 2007.

[2] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury. Wireless multimedia sensor networks: Applications and testbeds. *Proceedings of the IEEE*, 96(10):1588–1605, Oct. 2008.

[3] M. AlRashidi and M. El-Hawary. A survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation (TEVC)*, 13(4):913–918, August 2009.

[4] M. Altab Hossain, Y. Makihara, J. Wang, and Y. Yagi. Clothing-invariant gait identification using part-based clothing categorization and adaptive weight control. *Pattern Recognition (PR)*, 43:2281–2291, 2010.

[5] L. Bazzani, M. Cristani, A. Perina, M. Farenzena, and V. Murino. Multiple-shot person re-identification by hpe signature. In *20th International Conference on Pattern Recognition (ICPR)*, pages 1413–1416, August 2010.

[6] B. Bhanu, C. V. Ravishankar, H. Aghajan, A. K. Roy-Chowdhury, and D. Terzopoulos, editors. *Distributed Video Sensor Networks*. Springer, 1st edition, February 2011.

[7] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 232–237, 1998.

[8] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal Q2 1998*, 1998.

[9] G. R. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[10] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Online multi-person tracking-by-detection from a single, uncalibrated camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(9):1820–1833, September 2011.

[11] W. Burger and B. Bhanu. *Qualitative Motion Understanding*. Kluwer international series in engineering and computer science. Kluwer Academic Publishers, Boston, 1992.

[12] A. J. Carlisle. *Applying the Particle Swarm Optimizer to Non-stationary Environments*. PhD thesis, Auburn University, 2002.

[13] M. Castrillón, O. Déniz, M. Hernández, and C. Guerra. ENCARA2: Real-time detection of multiple faces at different resolutions in video streams. *Journal of Visual Communication and Image Representation (JVCIR)*, pages 130–140, 2007.

[14] T.-H. Chang and S. Gong. Tracking multiple people with a multi-camera system. In *IEEE Workshop on Multi-Object Tracking (MOT)*, pages 19–26, 2001.

[15] P. Chen, P. Ahammad, C. Boyer, S.-I. Huang, L. Lin, E. Lobaton, M. Meingast, S. Oh, S. Wang, P. Yan, A. Yang, C. Yeo, L.-C. Chang, J. Tygar, and S. Sastry. Citric: A low-bandwidth wireless camera network platform. In *Second ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2008)*, pages 1–10, Sept. 2008.

[16] W.-T. Chen, P.-Y. Chen, W.-S. Lee, and C.-F. Huang. Design and implementation of a real time video surveillance system with wireless sensor networks. In *IEEE Vehicular Technology Conference (VTC Spring 2008)*, pages 218–222, May 2008.

[17] C.-M. Cheng and S.-H. Lai. An integrated approach to 3d face model reconstruction from video. In *2001 IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 16–22, 2001.

[18] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. *First European Conference on Artificial Life*, 1991.

[19] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 142–149 vol.2, 2000.

[20] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893 vol. 1, June 2005.

[21] M. Deilamani and R. Asli. Moving object tracking based on mean shift algorithm and features fusion. In *2011 International Symposium on Artificial Intelligence and Signal Processing (AISP)*, pages 48–53, June 2011.

[22] W. T. Dempster and G. R. L. Gaughran. Properties of body segments based on size and weight. *American Journal of Anatomy*, 1967.

[23] W. Ding, Z. Gong, S. Xie, and H. Zou. Real-time vision-based object tracking from a moving platform in the air. *IEEE/RSJ Int. Conf. on Robots and Systems*, pages 681–685, 2006.

[24] S. Dockstader and A. Tekalp. Multiple camera tracking of interacting and occluded human motion. *Proceedings of the IEEE*, 89(10):1441–1455, Oct 2001.

[25] M. Enzweiler and D. Gavrila. Integrated pedestrian classification and orientation estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 982–989, 2010.

[26] M. Fanany, M. Ohno, and I. Kumazawa. A scheme for reconstructing face from shading using smooth projected polygon representation nn. In *2002 International Conference on Image Processing (ICIP)*, volume 2, pages II–305 – II–308 vol.2, 2002.

[27] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani. Person re-identification by symmetry-driven accumulation of local features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2360–2367, June 2010.

[28] L. G. Farkas. *Anthropometry of the Head and Face*. Raven Press, 1994.

[29] X. Fen and G. Ming. Pedestrian tracking using particle filter algorithm. In *IEEE International Conference on Electrical and Control Engineering (ICECE)*, pages 1478–1481, 2010.

[30] J. Ferryman and A. Ellis. PETS2010: Dataset and challenge. *Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 143–150, 2010. `http://pets2010.net/`.

[31] FFmpeg Team. libavcodec: audio/video codec library, 2010.

[32] R. B. Fisher. The PETS04 surveillance ground-truth data sets. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, pages 1–5, 2004. `http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`.

[33] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(2):267–282, 2008.

[34] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of rc4. In *RC4, Proceedings of the 4th Annual Workshop on Selected Areas of Cryptography*, pages 1–24, 2001.

[35] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[36] D. M. Gavrila. The visual analysis of human movement: a survey. *Computer Vision and Image Understanding (CVIU)*, 73(1):82–98, 1999.

[37] D. Gray, S. Brennan, and H. Tao. Evaluating appearance models for recognition, reacquisition, and tracking. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, 2007.

[38] Z. Hao, X. Zhang, P. Yu, and H. Li. Video object tracing based on particle filter with ant colony optimization. In *Second IEEE International Conference on Advanced Computer Control (ICACC)*, volume 3, pages 232–236, 2010.

[39] P. Hewitt and D. Dobberfuhl. The science and art of proportionality. *Science Scope*, 27(4):30–31, 2004.

[40] S. Ince and J. Konrad. Occlusion-aware optical flow estimation. *IEEE Transactions on Image Processing (TIP)*, 17(8):1443–1451, August 2008.

[41] M. Isard and A. Blake. CONDENSATION: Conditional density propagation for visual tracking. *International Journal of Computer Vision (IJCV)*, 29:5–28, 1998.

[42] F. Jean, R. Bergevin, and A. Albu. Body tracking in human walk from monocular video sequences. In *Second Canadian Conference on Computer and Robot Vision (CRV)*, pages 144–151, May 2005.

[43] N. Joshi, S. Avidan, W. Matusik, and D. Kriegman. Synthetic aperture tracking: Tracking through occlusions. In *Eleventh IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, October 2007.

[44] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[45] J. Kang, I. Cohen, G. Medioni, and C. Yuan. Detection and tracking of moving objects from a moving platform in presence of strong parallax. *ICCV 2005*, 1:10–17 Vol. 1, 2005.

[46] D. Karaboga. An idea based on honey bee swarm for numerical optimization. *Techn Rep TR06 Erciyes Univ Press Erciyes*, 129(2):2865, 2005.

[47] J. Kennedy and R. Eberhart. Particle swarm optimization. *International Conference on Neural Networks (ICNN)*, 4:1942–1948 vol.4, 1995.

[48] R. Kleihorst, A. Abbo, B. Schueler, and A. Danilin. Camera mote with a high-performance parallel processor for real-time frame-based video processing. In *IEEE Conference on Advanced Video and Signal Based Surveillance (PAVSS 2007)*, pages 69–74, Sept. 2007.

[49] C. M. Kohlhoff. Boost.Asio: a cross-platform c++ library for network and low-level I/O programming, 2010.

[50] M. Komeili, N. Armanfard, and E. Kabir. A fuzzy approach for multi-feature pedestrian tracking with particle filter. In *IST 2008*, pages 570–575, 2008.

[51] J. Landabaso, M. Pardas, and J. Casas. Reconstruction of 3d shapes considering inconsistent 2d silhouettes. In *2006 IEEE International Conference on Image Processing (ICIP)*, pages 2209–2212, oct. 2006.

[52] V. Le, Y. Hu, and T. Huang. A quantitative evaluation for 3d face reconstruction algorithms. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1269–1272, April 2009.

[53] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. In *ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*, pages 491–500, New York, NY, USA, 2002. ACM.

[54] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects. *Foundations and Trends in Computer Graphics and Vision (FTCGV)*, 1(1):1–89, 2005.

[55] M. Li, Z. Zhang, K. Huang, and T. Tan. Rapid and robust human detection and tracking based on omega-shape features. In *Sixteenth IEEE International Conference on Image Processing (ICIP)*, pages 2545–2548, November 2009.

[56] N. Li, B. Yan, and G. Chen. Measurement study on wireless camera networks. *Second ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2008)*, pages 1–10, Sept. 2008.

[57] Y. Li and B. Bhanu. Fusion of multiple trackers in video networks. In *Fifth ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–6, August 2011.

[58] Y. Li, C. Huang, and R. Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2953–2960, June 2009.

[59] Z. Li, Q. Tang, and N. Sang. Improved mean shift algorithm for occlusion pedestrian tracking. *IET Electronics Letters (EL)*, 44(10):622–623, 2008.

[60] J. Lim and D. Kriegman. Tracking humans using prior and learned representations of shape and appearance. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition (AFFG)*, pages 869–874, May 2004.

[61] S. Lim, J. Apostolopoulos, and A. Gamal. Optical flow estimation using temporally oversampled video. *IEEE Transactions on Image Processing (TIP)*, 14(8):1074–1087, August 2005.

[62] S. Lim and A. Gamal. Optical flow estimation using high frame rate sequences. In *IEEE International Conference on Image Processing (ICIP)*, volume 2, pages 925–928 vol.2, Oct 2001.

[63] H. Lu, W. Zhang, F. Yang, and X. Wang. Robust tracking based on PSO and on-line AdaBoost. In *Fifth IEEE Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, pages 690–693, 2009.

[64] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 674–679, 1981.

[65] I. Microsoft. Microsoft foundation classes (MFC), 1992-2008.

[66] Network Working Group. RFC 2435: RTP payload format for JPEG-compressed video, 1998.

[67] Network Working Group. RFC 2616: Hypertext transfer protocol – HTTP/1.1, 1999.

[68] H. T. Nguyen and B. Bhanu. Tracking multiple objects in non-stationary video. In *Eleventh ACM/IEEE Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 1561–1568, July 2009.

[69] H. T. Nguyen and B. Bhanu. Tracking pedestrians with bacterial foraging optimization swarms. In *IEEE Congress on Evolutionary Computation (CEC)*, July 2011.

[70] H. T. Nguyen, B. Bhanu, A. Patel, and R. Diaz. VideoWeb: Design of a wireless camera network for real-time monitoring of activities. In *Proceedings of the Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2009)*, August 2009.

[71] H. T. Nguyen, B. Bhanu, A. Patel, and R. Diaz. Design and optimization of the VideoWeb wireless camera network. *EURASIP Journal on Image and Video Processing (JIVP)*, (865803), 2010.

[72] T. B. Nguyen and S. T. Chung. An improved real-time blob detection for visual surveillance. In *CISP 2009*, pages 1–5, 2009.

[73] Y. Owechko and S. Medasani. Cognitive swarms for rapid detection of objects and associations in visual imagery. *IEEE Swarm Intelligence Symposium*, pages 420–423, 2005.

[74] C. Park and P. H. Chou. eCAM: ultra compact, high data-rate wireless sensor node with a miniature camera. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 359–360, New York, NY, USA, 2006. ACM.

[75] H. Park, J. Burke, and M. B. Srivastava. Design and implementation of a wireless sensor network for intelligent light control. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 370–379, New York, NY, USA, 2007. ACM.

[76] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine (CSM)*, Vol. 22, No. 3:52–67, 2002.

[77] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. A 3d face model for pose and illumination invariant face recognition. In *Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 296–301, September 2009.

[78] P. Phillips, H. Moon, S. Rizvi, and P. Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(10):1090–1104, October 2000.

[79] R. Pockaj. Mpeg-4 facial definition parameter specifications. *ISO/IEC MPEG-4 Part 1 (Systems)*, 1998.

[80] R. Quinlan. C4.5. http://www.rulequest.com/Personal/.

[81] M. Quinn, R. Mudumbai, T. Kuo, Z. Ni, C. D. Leo, and B. S. Manjunath. Visnet: A distributed vision testbed. In *Second ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2008)*, pages 364–371, Sep 2008.

[82] D. Ramanan, D. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(1):65–81, January 2007.

[83] S. Romdhani and T. Vetter. Efficient, robust and accurate fitting of a 3d morphable model. In *9th IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 59–66, oct. 2003.

[84] A. K. Roy-Chowdhury and R. Chellappa. Face reconstruction from video using uncertainty analysis and a generic model. *Computer Vision and Image Understanding (CVIU)*, 91(1-2):188–213, 2003.

[85] P. Saisan, S. Medasani, and Y. Owechko. Multi-view classifier swarms for pedestrian detection and tracking. In *CVPR Workshops 2005*, page 18, 2005.

[86] J. Saragih, S. Lucey, and J. Cohn. Face alignment through subspace constrained mean-shifts. In *12th IEEE International Conference on Computer Vision (ICCV)*, pages 1034–1041, October 2009.

[87] D. S. Sen Wang, Lei Zhang. Face reconstruction across different poses and arbitrary illumination conditions (avbpa). In *Biometric Authentication Workshop*, pages 91–101, 2005.

[88] N. Seo. OpenCVX: Yet another OpenCV eXtension. `http://code.google.com/p/opencvx/`.

[89] L. G. Shapiro and G. Stockman. *Computer Vision*. Prentice Hall, January 2001.

[90] K. Shoemake. Animating rotation with quaternion curves. *ACM Siggraph Computer Graphics*, 19:245–254, 1985.

[91] B. Song, T.-Y. Jeng, E. Staudt, and A. K. Roy-Chowdhury. A stochastic graph evolution framework for robust multi-target tracking. In *Eleventh European Conference on Computer Vision (ECCV)*, pages 605–619, Berlin, Heidelberg, 2010. Springer-Verlag.

[92] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision (IJCV)*, 7(1):11–32, 1991.

[93] T. Teixeira, D. Lymberopoulos, E. Culurciello, Y. Aloimonos, and A. Savvides. A lightweight camera sensor network operating on symbolic information. In *First Workshop on Distributed Smart Cameras 2006*, November 2006.

[94] D. Teodorovic, P. Lucic, G. Markovic, and M. D. Orco. Bee colony optimization: Principles and applications. In *Eighth Seminar on Neural Network Applications in Electrical Engineering (NEUREL)*, pages 151–156, September 2006.

[95] The MathWorks. MATLAB, 1994-2012.

[96] F. van den Bergh and A. Engelbrecht. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation (TEVC)*, 8(3):225–239, June 2004.

[97] P. Viola and M. Jones. Robust real-time object detection. *Second International Workshop on Statistical and Computational Theories of Vision (SCTV)*, 2001.

[98] D. Vlasic, M. Brand, H. Pfister, and J. Popovi'c. Face transfer with multilinear models. *ACM Transactions on Graphics (TOG)*, 24(3):426–433, 2005.

[99] J. Xiao, S. Baker, I. Matthews, and T. Kanade. Real-time combined 2d+3d active appearance models. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–535 – II–542 Vol.2, June-2 July 2004.

[100] J. Xing, H. Ai, and S. Lao. Multiple human tracking based on multi-view upper-body detection and discriminative learning. In *20th IEEE International Conference on Pattern Recognition (ICPR)*, pages 1698–1701, aug. 2010.

[101] F. Xu and M. Gao. Human detection and tracking based on HOG and particle filter. In *CISP 2010*, volume 3, pages 1503–1507, 2010.

[102] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13, 2006.

[103] J. Yu and B. Bhanu. Super-resolution of deformed facial images in video. In *15th IEEE International Conference on Image Processing (ICIP)*, pages 1160–1163, October 2008.

[104] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008.

[105] X. Zhang, W. Hu, W. Li, W. Qu, and S. Maybank. Multi-object tracking via species based particle swarm optimization. In *12th IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1105–1112, October 2009.

[106] 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)*, pages 1–8, June 2008.

[107] Z. Zhang, H. Gunes, and M. Piccardi. Tracking people in crowds by a part matching approach. In *IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 88–95, September 2008.

[108] T. Zhao, M. Aggarwal, R. Kumar, and H. Sawhney. Real-time wide area multi-camera stereo tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 976–983 vol. 1, June 2005.

[109] T. Zhao, R. Nevatia, and B. Wu. Segmentation and tracking of multiple humans in crowded environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(7):1198–1211, July 2008.

[110] Y. Zheng and Y. Meng. The PSO-based adaptive window for people tracking. In *First IEEE Symposium on Computational Intelligence for Security and Defence Applications (CISDA)*, pages 23–29, 2007.

[111] Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. In *17th IEEE International Conference on Pattern Recognition (ICPR)*, volume 2, pages 28–31, 2004.

[112] Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters (PRL)*, 27:773–780, 2006.