Roger Simmons
Editor

Computer Science, Technology and Applications

Surveillance Systems

Design, Applications and Technology

NOVA

COMPUTER SCIENCE, TECHNOLOGY AND APPLICATIONS

SURVEILLANCE SYSTEMS

DESIGN, APPLICATIONS AND TECHNOLOGY

No part of this digital document may be reproduced, stored in a retrieval system or transmitted in any form or by any means. The publisher has taken reasonable care in the preparation of this digital document, but makes no expressed or implied warranty of any kind and assumes no responsibility for any errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of information contained herein. This digital document is sold with the clear understanding that the publisher is not engaged in rendering legal, medical or any other professional services.

COMPUTER SCIENCE, TECHNOLOGY AND APPLICATIONS

Additional books in this series can be found on Nova's website under the Series tab.

Additional e-books in this series can be found on Nova's website under the eBooks tab.

COMPUTER SCIENCE, TECHNOLOGY AND APPLICATIONS

SURVEILLANCE SYSTEMS

DESIGN, APPLICATIONS AND TECHNOLOGY

ROGER SIMMONS
EDITOR

nova publishers
New York

Copyright © 2017 by Nova Science Publishers, Inc.

All rights reserved. No part of this book may be reproduced, stored in a retrieval system or transmitted in any form or by any means: electronic, electrostatic, magnetic, tape, mechanical photocopying, recording or otherwise without the written permission of the Publisher.

We have partnered with Copyright Clearance Center to make it easy for you to obtain permissions to reuse content from this publication. Simply navigate to this publication's page on Nova's website and locate the "Get Permission" button below the title description. This button is linked directly to the title's permission page on copyright.com. Alternatively, you can visit copyright.com and search by title, ISBN, or ISSN.

For further questions about using the service on copyright.com, please contact:
Copyright Clearance Center
Phone: +1-(978) 750-8400 Fax: +1-(978) 750-4470 E-mail: info@copyright.com.

NOTICE TO THE READER

The Publisher has taken reasonable care in the preparation of this book, but makes no expressed or implied warranty of any kind and assumes no responsibility for any errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of information contained in this book. The Publisher shall not be liable for any special, consequential, or exemplary damages resulting, in whole or in part, from the readers' use of, or reliance upon, this material. Any parts of this book based on government reports are so indicated and copyright is claimed for those parts to the extent applicable to compilations of such works.

Independent verification should be sought for any data, advice or recommendations contained in this book. In addition, no responsibility is assumed by the publisher for any injury and/or damage to persons or property arising from any methods, products, instructions, ideas or otherwise contained in this publication.

This publication is designed to provide accurate and authoritative information with regard to the subject matter covered herein. It is sold with the clear understanding that the Publisher is not engaged in rendering legal or any other professional services. If legal or any other expert assistance is required, the services of a competent person should be sought. FROM A DECLARATION OF PARTICIPANTS JOINTLY ADOPTED BY A COMMITTEE OF THE AMERICAN BAR ASSOCIATION AND A COMMITTEE OF PUBLISHERS.

Additional color graphics may be available in the e-book version of this book.

Library of Congress Cataloging-in-Publication Data

ISBN: 978-1-53610-726-5 (eBook)

Published by Nova Science Publishers, Inc. † New York

CONTENTS

Preface		**vii**
Chapter 1	Omnidirectional Surveillance System for Household Safety *Kai Yiat Jim, Wai Kit Wong and Yee Kit Chan*	**1**
Chapter 2	Tracking Moving Objects in Video Surveillance Systems with Kalman and Particle Filters – A Practical Approach *Grzegorz Szwoch*	**55**
Chapter 3	Performance Evaluation of Single Object Visual Tracking: Methodology, Dataset and Experiments *Juan C. SanMiguel, José M. Martínez and Mónica Lozano*	**107**
Index		**143**

PREFACE

In this book, Chapter One reviews basis elementary of residence security, classical residence security and health care surveillance system versus computer vision technique system, as well as directional versus omnidirectional imaging. Chapter Two provides practical guidelines for specialists who design, tune and evaluate video surveillance systems based on the automated tracking of moving objects. Chapter Three presents a methodology for tracker evaluation that quantifies performance against variations of the tracker input.

Chapter 1 – Recent statistical results on home security reveal that around 3.7 million home break-ins are committed each year in the United States. On average, there is a home intrusion every 8.4 seconds. Homes without security systems are up to 300% more likely to be broken into and usually, police can only clear 13% of all reported burglaries due to the lack of witnesses or physical evidence. Obviously, this shows the necessity of installing a trespasser detection surveillance system in a residence to mitigate burglaries. Besides that, according to the 2010 National Health Interview Survey, the overall rate of nonfatal fall injury cases for which a health-care professional was contacted, was 43 per 1,000 people. This means that the other 957 fall injury cases were probably fatal due to inattentiveness or late notification. Therefore, to save more lives, there is also a need to install a health care surveillance system in a residence for human faint detection. In the last few decades, visual surveillance in residence areas has become an active research area for the computer vision field. Visual surveillance has a broad range of applications in day-to-day life, such as trespasser detection for security and faint detection for health care purposes in a residence. Therefore, numerous research personnel from video and image technology areas had paid more

deliberations to study and develop highly sophisticated surveillance system. Apart from this, some researchers had also worked in the development of the wide area monitoring surveillance systems, and they had even tried to formulate an algorithm to analyze the captured data. However, there are still some issues that are required to be addressed in this context, which include: (1) wide area coverage using minimal hardware, and (2) improved algorithm for object detection, tracking and even identifying. To address these issues, an effective omnidirectional surveillance system is proposed and developed, with the following features: 1. 360 degrees view angle using a single imaging tool. 2. Effective automatic trespasser detection that will raise alerts/alarms whenever any security threat (intruder break-in, burglar) arises. 3. Effective automatic faint detection that will raise alerts/alarms whenever any fainted human is detected. In this chapter, topics such as residence security, classical residence security, health care surveillance system, computer vision technique system, and directional versus omnidirectional imaging, will be discussed. The algorithm and utilization of some universal unwarping methods such as Discrete Geometric Transform method, Pano-mapping table method and Log-polar mapping method, will also be explained. In addition, automatic trespasser detection method using Extreme Point Curvature Analysis Algorithm and automatic faint detection method using Integrated Body Contours Algorithm, are implemented in the developed omnidirectional imaging system. Experimental results (from the experiments carried out to test the proposed algorithms for both security and health care surveillance in household wellbeing), are also shown. In the last session of this chapter, the work is summarized and some future enhancement is envied.

Chapter 2 – The development and tuning of an automated object tracking system for implementation in a video surveillance system is a complex task, requiring understanding how these algorithms work, and also the experience with choosing proper algorithm parameters in order to obtain accurate results. This Chapter presents a practical approach to the problem of a single camera object tracking, based on the object detection and tracking with Kalman filters and particle filters. The aim is to provide practical guidelines for specialists who design, tune and evaluate video surveillance systems based on the automated tracking of moving objects. The main components of the tracking system, the most important parameters and their influence on the obtained results, are discussed. The described tracking algorithm starts with the detection phase which identifies areas in each video image that represent moving objects, employing background subtraction and morphological processing. Next, movement of each detected object is tracked on a frame-by-

frame basis, providing a ‘track’ of each object. First, the Kalman filter approach is presented. Implementation of a dynamic model for the filter prediction, methods of obtaining the measurement for updating the filter, and the influence of the noise variance parameters on the results, are discussed. Tracking with Kalman filters fails in many practical situations when the tracked objects come into a conflict due to the object occlusion and fragmentation in the camera images. Another method presented here is based on particle filters which are updated using color histograms of the tracked objects. This method is more robust to tracking conflicts than the Kalman filter, but it is less accurate in describing the object size, and it is also much more demanding in terms of computation. Therefore, a combined approach for resolving the tracking conflicts, is proposed. This algorithm uses Kalman filters for the basic, non-conflict tracking, and switches to the particle filter for resolving cases of occlusion and fragmentation. A methodology of evaluation of tracking algorithms is also presented, and an example of testing the three presented tracking algorithms on a sample test video is shown.

Chapter 3 – Performance evaluation of visual tracking approaches (trackers) based on ground-truth data allows to determine their strengths and weaknesses. In this paper, the author present a methodology for tracker evaluation that quantifies performance against variations of the tracker input (data and configuration). It addresses three aspects: dataset, performance criteria and evaluation measure. A dataset with ground-truth is designed including common tracking problems such as illumination changes, complex movements and occlusions. Four performance criteria are defined: parameter stability, initialization robustness, global accuracy and computational complexity. A new measure is proposed to estimate spatio-temporal tracker accuracy to account for the human errors in the generation of ground-truth data. Then, such measure is compared with the related state-of-the-art showing its superiority to evaluate trackers. Finally, the proposed methodology is validated on state-of-the-art trackers demonstrating their utility to identify tracker characteristics.

In: Surveillance Systems
Editor: Roger Simmons
ISBN: 978-1-53610-703-6
© 2017 Nova Science Publishers, Inc.

Chapter 1

OMNIDIRECTIONAL SURVEILLANCE SYSTEM FOR HOUSEHOLD SAFETY

Kai Yiat Jim*, Wai Kit Wong† and Yee Kit Chan‡
Faculty of Engineering and Technology, Multimedia University,
Ayer Keroh Lama, Melaka, Malaysia

ABSTRACT

Recent statistical results on home security reveal that around 3.7 million home break-ins are committed each year in the United States. On average, there is a home intrusion every 8.4 seconds. Homes without security systems are up to 300% more likely to be broken into and usually, police can only clear 13% of all reported burglaries due to the lack of witnesses or physical evidence. Obviously, this shows the necessity of installing a trespasser detection surveillance system in a residence to mitigate burglaries. Besides that, according to the 2010 National Health Interview Survey, the overall rate of nonfatal fall injury cases for which a health-care professional was contacted, was 43 per 1,000 people. This means that the other 957 fall injury cases were probably fatal due to inattentiveness or late notification. Therefore, to save more lives, there is also a need to install a health care surveillance system in a residence for human faint detection.

* Email: helium_jim@yahoo.com.
† wkwong@mmu.edu.my.
‡ ykchan@mmu.edu.my.

In the last few decades, visual surveillance in residence areas has become an active research area for the computer vision field. Visual surveillance has a broad range of applications in day-to-day life, such as trespasser detection for security and faint detection for health care purposes in a residence. Therefore, numerous research personnel from video and image technology areas had paid more deliberations to study and develop highly sophisticated surveillance system. Apart from this, some researchers had also worked in the development of the wide area monitoring surveillance systems, and they had even tried to formulate an algorithm to analyze the captured data. However, there are still some issues that are required to be addressed in this context, which include: (1) wide area coverage using minimal hardware, and (2) improved algorithm for object detection, tracking and even identifying.

To address these issues, an effective omnidirectional surveillance system is proposed and developed, with the following features:

1. 360 degrees view angle using a single imaging tool.
2. Effective automatic trespasser detection that will raise alerts/alarms whenever any security threat (intruder break-in, burglar) arises.
3. Effective automatic faint detection that will raise alerts/alarms whenever any fainted human is detected.

In this chapter, topics such as residence security, classical residence security, health care surveillance system, computer vision technique system, and directional versus omnidirectional imaging, will be discussed. The algorithm and utilization of some universal unwarping methods such as Discrete Geometric Transform method, Pano-mapping table method and Log-polar mapping method, will also be explained. In addition, automatic trespasser detection method using Extreme Point Curvature Analysis Algorithm and automatic faint detection method using Integrated Body Contours Algorithm, are implemented in the developed omnidirectional imaging system. Experimental results (from the experiments carried out to test the proposed algorithms for both security and health care surveillance in household wellbeing), are also shown. In the last session of this chapter, the work is summarized and some future enhancement is envied.

1. INTRODUCTION

Household surveillance has been an important part of our daily lives, whether it is to prevent trespassing or to ensure the safety of our loved ones. Trespassing problem has been around for a few decades and it keeps on

increasing, causing potential danger to our safety. On the other hand, cases where the elderly faint and experience fatal injuries, have also been increasing throughout the years. Hence, this work is proposed to solve both the trespassing problem and the elderly fainting issue.

Burglar alarm system and video surveillance have been widely used around the world as a solution for trespassing detection. On the other hand, wearable sensors and video monitoring have been implemented as a healthcare surveillance for the elderly. However, all of them are not as flexible as the image processing based surveillance. Image processing based surveillance is able to detect automatically, capture images and cover a wide area of surveillance.

Common video surveillance employs directional view with a limitation of 180 degrees view angle and more cameras will be needed to cover a wider view angle. However, more cameras will also increase the total cost of the surveillance system. Therefore, a method has been devised to obtain omnidirectional image with 360 degrees view angle using only minimal hardware. Generally, mechanical approach and optical approach have been used by practitioners to obtain omnidirectional image. However, optical approach has always been favoured since mechanical approach leads to many problems of discontinuity and inconsistency.

Optical approach often has image deformation problem which causes difficulty to interpret the image taken. Thus, it is necessary to implement an efficient unwarping method on the omni-image taken. Basically, unwarping is the process in digital image processing that “opens’ up an omni-image into a panoramic image. Subsequently, information of the panoramic image can be easily interpreted for any direct implementations. There are 3 unwarping methods actively adopted in the application of visual surveillance system all around the world, which are the pano-mapping table method, discrete geometry techniques (DGT) method and log-polar mapping method. The best method is selected based on the advantages and disadvantages of each methods.

Finally, automatic trespasser and faint detection algorithm will be implemented into the hardware setup to form a complete surveillance system. The proposed methods include the extreme point curvature analysis algorithm and the integrated body contours algorithm. Extreme point curvature analysis algorithm checks the curves on the top, bottom, left and right of an object blob to detect trespasser, whereas integrated body contours algorithm combines head detection, leg detection, ellipse fitting’s ratio and orientation as the main features to detect faint. These automatic detection algorithms are needed due

to the fact that the current video surveillance requires monitoring by humans and the efficiency will drop as they grow tired.

In this chapter, the topics will be divided into surveillance system, omnidirectional imaging system, automatic trespasser & faint detection algorithm, experimental results & discussion and lastly the conclusion & future research direction.

2. SURVEILLANCE SYSTEM

2.1. Trespasser Surveillance System

An act of trespassing can be defined as entering another person's property without permission of the owner or his/her agent and without lawful authority and causing any damage, no matter how slight. Any interference with the owner's use of the property is a sufficient showing of damage and is a civil wrong (tort) sufficient to form the basis for lawsuit against the trespasser by the owner or a tenant using the property [1].

In this work, we are focusing solely on trespassing in common household (for any reasons involving burglary, theft and privacy invasion), where the household members are not present during the trespassing incident. According to a report by Bureau of Justice Statistics [2], an average of 3.7 million burglaries had occurred annually from 2003 to 2007 in the United States of America. This shows that there was a household burglary case committed in every 8.4 s. Statistics also show that without a proper security system, households are 300% more likely to be victims of burglary and out of all the reported cases, only a partial or 13%, can be solved by the police due to the lack of witnesses and physical evidence [3].

Hence, having an efficient security or surveillance system is crucial in combating household burglary as well as preventing any loss or damage to the property. Trespasser surveillance system functions as a shield that detects intrusion, identifies crime and activates an alarm as a response to emergency. Commonly used or conventional trespasser surveillance systems available in the market, can be divided mainly into burglar alarm system and video surveillance.

Table 1. Type of burglar alarm system

Type of burglar alarm system	Descriptions
1. Passive infrared (PIR) motion detector system	- It functions by learning the ambient temperature of the surroundings within its field of view, and then detecting a change in the temperature caused by the newly detected object. This change is detected through radiating infrared (IR) light. By using comparison in the differentiation principle, PIR-based motion detector can determine whether intrusion has occurred. - It has simple installation process, low cost and less sensitive to illumination changes. However, it can be easily set off by moving objects. It cannot detects people who are standing still and it does not tolerate large areas or large temperature changes.
2. Ultrasonic motion detector system	- An ultrasonic signal is radiated from the transmitter into the area of surveillance and subsequently reflected by any solid objects (such as walls, floors and ceilings) before it is detected by the receiver. When the surfaces are stationary, the received frequency should be equal to the transmitted frequency. Therefore, when a change in frequency occurs due to a moving object or a person (based on the Doppler's principle), this indicates the possibility of trespassing. - Ultrasonic motion detector system is rarely installed because it is prone to false alarms. These false alarms are caused by animals, insects and even gusts of wind. Moreover, it has poor efficiency for trespasser detection in areas with large objects.
3. Glass breaks detector system	- This system is commonly placed near a glass door or a glass window, to detect any possible glass shattering or breaking condition, which indicates trespassing. Any noise or vibration from the glass will be detected by the microphone, and then being processed by the detector circuitry. This detector would simply compares the received signal to the signal of typical glass shattering, using signal transforms such as Discrete Cosine transform or fast Fourier transform. If both the amplitude threshold showed similarity, then trespassing is assumed to have happened. - For the downside, glass break detector can only be used at limited areas where glass is present. False alarms might happen as it is sensitive to environmental effects such as rain or impacts to the glass by any object.
4. Photoelectric beam system	- This system is usually located at openings such as hallways or doorways, and it acts similarly to a trip wire. The transmitter emits a consistent beam of infrared light to the receiver using Light Emitting Diode (LED), and the receiver which consists of mostly photoelectric cells, will detect the presence of the beam. If the detected beam of light drops to 90% detection, an alarm signal will be generated to indicate intrusion.

Table 1. (Continued)

	- It is easy to use, highly immune to ambient light and its functionality is not affected by electrical and magnetic fields. However, it is quite costly to install the system (mounting, wiring and adjusting) and false alarms are inevitable due to the detection of small objects such as animals.
5. Vibration sensors system	- When movement or vibration is detected, the unstable mechanical configuration in the electrical circuit shifts and breaks the current flow. This will activates the alarm. The technology of vibration sensors system varied and they are sensitive to different levels of vibrations. Hence, suitable sensors must be selected for different types of structures and configurations. - Vibration sensors are reliable as they generate low false alarm rate in trespasser detection and they have a moderate price range. However, vibration sensors system is not widely used in the market as it is a rather new technology with unproven record of use, and the installation requires it to be fence mounted.
6. Passive magnetic field detection system	- The system applies an electromagnetic field generator powered by two wires running in parallel. The wires are connected to a signal processor which senses the anomaly in the magnetic field. - Although this detection system has a very low false alarm rate, it can have high interference if it is installed near high voltage lines or radars.
7. Micro-phonic detection system	- The signal processor analyzes the signals generated by the minute flexing of triboelectric coaxial sensor cable, to detect any sounds similar to climbing, cutting or moving the fence structure. The system's sensitivity can be adjusted to suit required environmental conditions. Once the detected signals exceed the pre-set conditions, then the alarm is generated. - Micro-phonic detection system is easy to install, simple in configuration and cheap in cost. However, it has a high false alarm rate because it can be easily set off by large animal's contact, extreme weather and overgrown vegetation.
8. Taut wire perimeter security system	- A taut wire perimeter security system is made up of a stream of tensioned tripwires located usually on a wall or a fence to detect any physical attempt to trespass. Detectors or switches (such as electronic strain gauge, static force transducer or simple mechanical contact), detect movements at each end of the tensioned wires and activate the alarm if the inputs exceed the intended threshold. - This system has reliable sensors, low false alarm rate and high detection rate. However, it is expensive, complicated to install and the technology is quite ancient.

2.1.1. Burglar Alarm System

Burglar alarm system is an electronic alarm made to alert the users to any intrusion in the selected property. It functions by having detection sensors connected to a control unit through a narrowband RF signal or low voltage wiring. Then, the control unit communicates with a response device such as a sound alarm. Burglar alarm system can be either hardwired or wirelessly installed depending on the requirements of the user. Hardwired system is more efficient while wireless system is easier to be installed. Nevertheless, burglar alarm system can be divided into many types, such as passive infrared (PIR) motion detector system, ultrasonic motion detector system, glass breaks detector system, photo electric beams system, vibration sensors system, passive magnetic field detection system, micro-phonic system, taut wire perimeter security system and much more. Each of these burglar alarm systems will be summarised in the following table.

As a conclusion, conventional burglar alarm systems are widely used by the community because of the simple implementation, ease of installation and low overall cost. However, we could not deny that these alarm systems have high false alarm rate due to factors such as weather, animal and electronic interference. Besides, these systems can be easily disabled by any trained trespassers.

2.1.2. Video Surveillance

Video surveillance dates back to as early as 1965, with the start of analog video surveillance or simple closed circuit television (CCTV) monitoring, where it is implemented by police in the United States to ensure public places security. CCTV is the application of video cameras to send image signal to a specific location, on a limited set of monitors.

In the mid of 1990s, the introduction of digital technology simply replaces the available analog technology in video surveillance. This is possible due to the dropping price (as a result of the computer revolution) of digital recording. Besides, digital surveillance is faster, more efficient and clearer compared to analog surveillance. Instead of changing analog video tapes daily, hard drives with high compression capability and lower storage cost, enable digital surveillance users to record for as much as a month’s worth of surveillance contents.

Most of the digital video surveillances used in common household for trespassing prevention, can be divided into either wireless security digital camera or wired surveillance digital camera. Wireless security digital cameras are easy to install, small in size, wireless and flexible. On the other hand,

wired surveillance digital cameras are wired, lack in flexibility and appropriate for permanent setup. Both serve the same functions as to transmit image signals to a center hub which are then being displayed on a monitor screen for viewing. However, manpower is still required to observe the monitor screen and to determine the presence of trespassers. Continuous monitoring will be less effective as the person gets distracted due to fatigue [4]. This will eventually causes errors such as false alarm and unnoticed trespassing. As a conclusion, although video surveillance is widely used for security purposes, it does not serves as the best solution to counter trespassing in places such as households.

2.2. Health Care Surveillance System

Health care can be defined as the prevention, treatment, and management of illness and the preservation of mental and physical well-being through the services offered by the medical and allied health professions [5]. Health care is also any field or enterprise concerned with supplying services, equipment, information, etc., for the maintenance or restoration of health [6]. Hence, health care surveillance system acts as a tool to inform a health personnel or authority whenever a faint occurs. This is to prevent any further injuries and to preserve the health of the victim.

In this work, we are only focusing on faint cases among the elderly and patients, in an indoor environment. According to the 2010 National Health Interview Survey [7], 43 out of 1000 people experienced nonfatal fall injury where a health-care professional was contacted for immediate attention. This means that the other 957 patients may have experienced death causing fall injury, due to late notification or inattentiveness. Fall and faint has also grown to be the major cause of injury and death among the elderly. 1 out of 3 elderly people ages 65 and above falls each year and 20% to 30% of them suffered moderate to severe injuries that may increase the risk of early death [8].

Thus, having equipped with an efficient health care surveillance system could prevent severe injuries and even death. Related personnel will be notified whenever a fall or faint situation occurs and immediate actions can be taken. There are numerous products, tools and applications available in the market for the field of healthcare surveillance system. They can be divided into two categories which are the wearable sensor and also video surveillance.

2.2.1. Wearable Sensor

Wearable sensor devices exist in many different forms such as watches, tablets, mobiles phones, and much more. They assist to monitor, gather information and send alert to the caretakers in case of emergency. Following are a few examples of wearable sensors available in the market [9, 10]:

Table 2. Type of wearable sensor

Name of wearable sensor	Descriptions
1. Metria wearable sensor	This small wearable device is a discreet and comfortable way for medical providers to monitor the health of seniors and patients. It sticks directly to the body with skin-friendly adhesive and collects data such as heart rate, blood pressure and amount of sleep. Then these data are interpreted by advanced computer algorithms. Patients and caregivers can also examine the data through a mobile app.
2. The Jawbone UP system	A wristband monitor that tracks movement and sleep details while enabling logging of exercise, food and hydration. These are done through a mobile app. Caregivers can always monitor the fitness, nutrition and sleep patterns information at anytime.
3. EverThere emergency response system	A light portable device (compatible with AT&T's cellular network) that has a call button which informs any nearby care center whenever the user experiences emergency such as a fall. It also includes hands-free voice communication in case of any emergency disrupting the user's mobility and it can even make a call just by giving a command. The care center can determine the user's location from the internal GPS and take appropriate actions.
4. Lechal insoles	These smart insoles allow seniors to have a greater degree of independence by providing direction assistance in the form of gentle vibrations and phone notifications. The compatible app also allows for location sharing which enables caretaker to keep tabs on the user no matter where they are.
5. Withings Pulse O_x	This wearable wristband monitors the user's health information such as heart rate, blood oxygen levels, activity status and sleep quality. This tracker isn't optimized for sharing data with caregivers. Instead, it allows the wearer more control over their own health and fitness. By monitoring the statistics or vitals, they can recognize any potential hazardous deviations and take action to alert the related authorities.

These wearable sensors are efficient and may even be lifesaving tools whenever there is an emergency. However, we could not overlook the possibility that the elderly or patients will forget to use the product (wearing them or bringing with them). Elderly people tend to be more forgetful and having to use the product at all times would be meticulous to them.

2.2.2. Video Surveillance

With the improvement of technology and reduction of cost, video monitoring/surveillance has become a very popular selection for keeping an eye on the elderly or patients, who are alone at home. Home video surveillance cameras available in the market today are sleek, small and easy to install. With the integration of wireless internet connection, some video surveillance are able to be streamed live at anytime and anywhere, while providing cloud based storage capability which is very useful for long hour recordings. Although camera capabilities will vary depending on the products, all the best devices provide wide-view angle, high definition video quality, night vision and built-in motion or sound detection that can notify the caretaker whenever something is happening. The two-way audio even allows the user and caretaker to communicate easily [11].

However, video surveillance still requires manpower to monitor the activities of the users. It is not possible for the caretaker to be putting continuous attention on the video surveillance for the entire day. Errors such as overlooking are prone to happen as the caretaker gets tired from daily activities while monitoring. In fact, fall or faint for a short period of time among elders or patients, may be a huge threat depending on the cause of the injury. Hence, it is important that faint cases are reported as soon as possible so that immediate treatments can be taken.

2.3. Image Processing

Image processing based surveillance system is the integration of a video output from a digital camera and a set of detection algorithm. Although image processing based surveillance is not widely used in common household for both trespasser and health care surveillance system, it is favoured by many practitioners/researchers. This is because image processing based surveillance provides more flexibility in terms of functionality, cost and customization. Following are a few benefits of image processing based surveillance for both trespasser and healthcare surveillance system:

Table 3. Benefits of image processing on different type of surveillance system

Type of system	Benefits of image processing
Trespasser surveillance system	It can automatically detects trespasser and captures images as evidence. This is an added advantage since conventional burglar alarm system could not capture images and conventional video surveillance requires monitoring by humans to detect trespasser. It has a low cost since it requires only a simple digital camera and software to operate, whereas some of the high-end burglar alarm systems can be very expensive.
Healthcare surveillance system	It can automatically detects fainting and alerts the caretaker, whereas the common video monitoring system requires human observation. It covers a wide area of surveillance which is more convenient and efficient, as compared to wearable sensors that could always be forgotten to be put on.

Based on the arguments stated above, image processing based surveillance system has been chosen to be used in this work.

2.4. Directional vs Omnidirectional

Most of the surveillance products or digital cameras available in the market apply directional viewing angle that is restricted to a maximum range of 180 degrees. Directional viewing angle can only monitor a limited amount of space, and multiple devices are required to increase the viewing angle. Moreover, the total cost will be more expensive with the use of multiple devices. On the other hand, omnidirectional prompts to the concept of existence in all directions, with 360 degrees of area coverage on a single plane/axis. With such a large visual field coverage, it will be beneficial in areas such as panoramic imaging or robotics.

An early approach to achieve omnidirectional viewing is by combining snapshots captured separately into a single and continuous image such as the RANSAC iterative algorithm [12]. RANSAC or “Random Sample Consensus” was first introduced by Fischler and Boller in 1981 to solve a problem involving a pair of stereo imaging tools. This algorithm is able to estimate the parameters with high degree of accuracy but it is time consuming and sometimes endless.

Moreover, omnidirectional viewing angle is being widely used in robotics and computer vision field with applications such as visual odomotery [13] and SLAM [14]. Visual odometry is the process of defining the position and orientation of a robot by analysing the captured image from the attached imaging tools, whereas SLAM or simulatenous localization and mapping, is a technique applied by autonomous vehicles and mobile robots to form or update a map within an unknown or known environment while keeping track of their current location. With the help of omnidirectional visualization, these applications are able to achieve better results in terms of optical flow, feature selection and matching.

With all the points stated above, omnidirectional view proves to be more useful than directional view especially in surveillance. Thus, omnidirectional viewing angle is selected to be used in this work. Section 2.4.1 below will further discuss on the methods applied to acquire omnidirectional view.

2.4.1. Mechanical Approach vs Optical Approach

The methods applied to obtain omnidirectional images can be divided into mechanical approach and optical approach. Mechanical approach gathers multiple images and combines them to generate an omnidirectional image, while optical approach captures an omnidirectional image at an instance.

Mechanical approach can be classified into two categories [15], which are the single viewpoint and multiple viewpoints. A rotating camera [16-19], is an example of single viewpoint mechanical approach. The camera rotates around the centre of projection using a rotating motor while taking multiple images. These images are then joined together to obtain a panoramic or omnidirectional view of the scene. However, since the time of each images taken are not of the same instance due to the rotating property of the camera, it is impossible to generate a real-time omnidirectional image. On the contrary, omnidirectional image is relatively easier to construct by using multiple viewpoints mechanical approach. Multiple cameras are used to capture images at the same time at multiple viewpoints, and then, these images are combined to form an omnidirectional image. Such technology is adopted by Quick Time VR system [20] where it has many market applications. However, the images generated by this system are not always continuous and consistent. It also cannot capture the dynamic scene at video rate.

Optical approach does not need the use of motor, and it can capture omnidirectional image at once, with the help of an imaging tool. Two famous alternatives used in this matter include the special purpose lens or fish eye lens [21], and hyperbolic optical mirror [22]. Fish eye lens with a very short focal

length, allows the camera to view in a much wider range that resembles a hemisphere scene. Although fish eye lens has been used in numerous applications that require wide angle [23, 24], Nalwa [25] found that it is difficult to design a fish eye lens that can ensures all incoming principal rays intersect at a single point to provide a fixed viewpoint. This means that the obtained image does not provide distortion free perspective image of the viewed scene. Hence, a complex and large design is required to build an optimal fish eye lens that can capture a good omnidirectional view image. However, this optimal fish eye lens may cost a fortune. Meanwhile, hyperbolic optical mirror offers a cheaper solution, less complexity in design and provides the same reflective quality as the fish eye lens.

Since mechanical approach leads to many problems on discontinuity and inconsistency, optical approach is selected to be used in this work. Optical approach particularly the hyperbolic optical mirror is preferred, as it outperforms fish eye lens as stated above. The proposed omnidirectional surveillance system model will be discussed in section 3.1 below.

3. OMNIDIRECTIONAL IMAGING SYSTEM

3.1. Proposed System Model

Figure 1 shows the proposed omnidirectional surveillance system model. Initially, the image is being captured by using the combined camera set as shown in Figure 2(d). The camera set consists of a wireless webcam and a custom hyperbolic mirror, attached together using a custom bracket. Then the image taken is fed to the personal computer through a wireless router. Wireless router acts only as the medium for transfer, and it can be connected to multiple wireless webcams if monitoring on multiple locations is needed. Lastly, the image taken is being analysed in the personal computer with the use of image processing or computer vision. Once the required conditions are met, the alarm is activated.

The camera used in this system model is the DCS-1130L Dlink IP camera with 640 x 480 resolution as shown in Figure 2(a). Next, the custom hyperbolic optical mirror used is a small size wide view type, with an outer diameter of 40mm and an angle of view of 45 degrees above horizontal plane, as shown in Figure 2(c). A laptop computer with Intel Core I5 2.4Ghz, 4GB of RAM and image processing software MATLAB, is chosen to be used. Last but not least, the laptop speakers are treated as the alarm in this work.

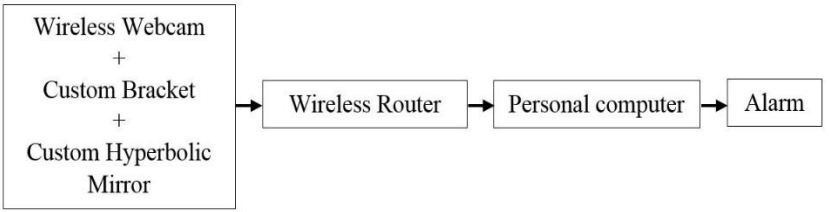


Figure 1. Omnidirectional surveillance system model.

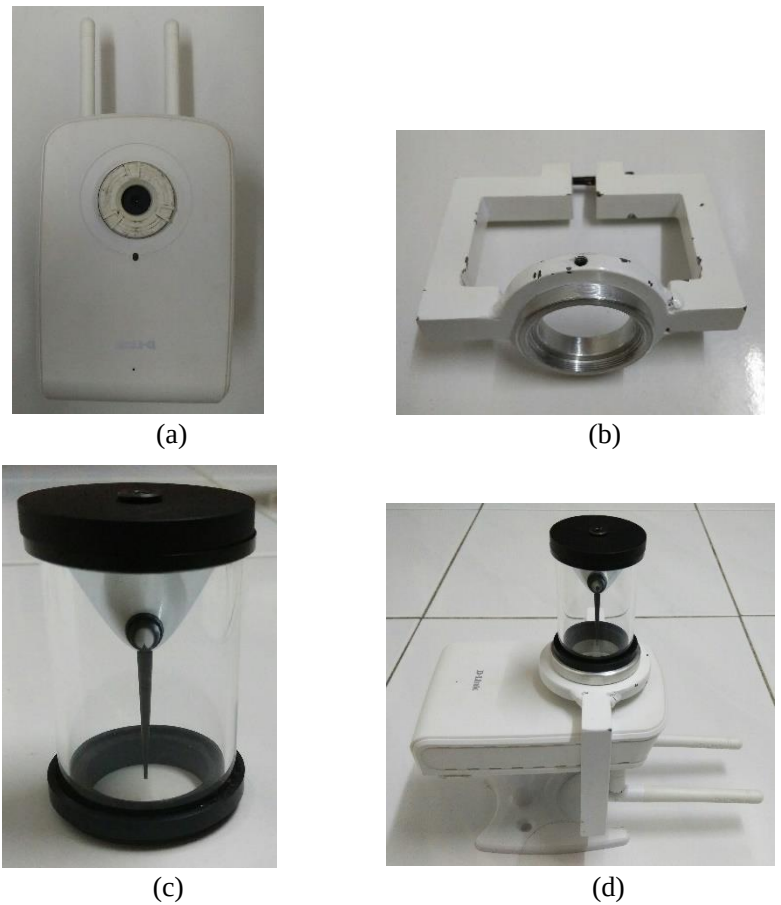


Figure 2. (a) Wireless webcam, (b) Custom bracket, (c) Custom hyperbolic mirror, (d) Combined camera set.

3.2. Unwarping Methods

The hyperbolic mirror image taken, has image deformation that may leads to analysis difficulty. Therefore, it is necessary to have a suitable method for unwarping the hyperbolic mirror image into an easy to read form. Generally, unwarping is a method in digital image processing, where the spherical hyperbolic mirror image is 'opened' up into a panoramic image that can be directly used and understood. There are 3 universal unwarping methods that are currently applied actively around the world for transforming omnidirectional mirror image into panoramic image. These methods include the discrete geometry technique method (DGT) [26], pano-mapping table method [27] and the log-polar mapping method [28]. The following review on the unwarping methods is done in a work [29] by W. S. Pua.

3.2.1. Discrete Geometry Technique Method

Discrete geometry techniques (DGT) method, by the name itself, means that this technique is used by applying one-by-one, the geometry of the image, discretely, in order to successfully un-warp the omnidirectional mirror image into a panoramic image. This method is practically used in transforming the omnidirectional images on a cylindrical surface into panoramic images, using PDE based resampling models [26].

In DGT method, it is required to perform the calculation of each and every pixels in the omni-image first, and then check for its corresponding radius from the center of the omnidirectional image. These are to determine whether the pixels should be considered in the next process or not. The calculations start from a fixed position and direction, which are from the right side and goes counter-clockwise for 360°. A circle with a radius of 1, will be visualized in the center of an omnidirectional image, which also means that the circle will be in size of 3x3 pixels. All of the pixels in this boundary of 3x3 pixels will be considered, and their corresponding radius will be calculated. For all pixels which fall within the radius of 1 (radius of concern), will be considered in the conversion. Since the pixels are generally an area of data information, it is possible that the circle will lie in between the pixels. Therefore, a tolerance of ±½ radius is set to counter this problem. In other words, a circle of radius 1, will consider the pixels lying between radius of 0.5 to 1.4, and a circle of radius 2, will consider the pixels lying between radius of 1.5 to 2.4, and so on. An example is shown in Figure 3(a).

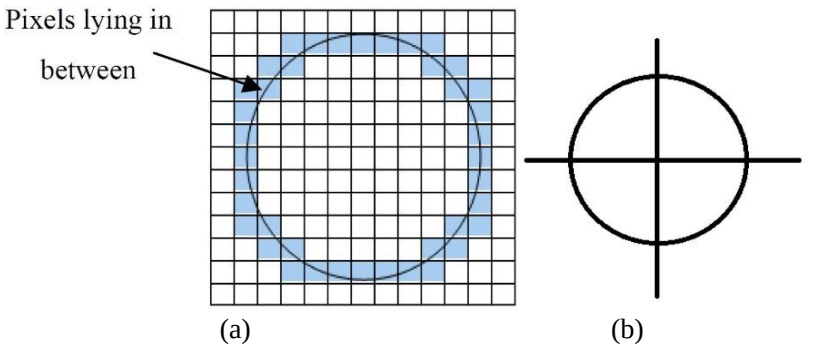


Figure 3. (a) Circle lying in between pixels, (b) Circle being split into 4 sections.

As soon as a pixel in the boundary is deemed to be considered or in range of the radius, it will be mapped into a new matrix of panoramic-image. However, since the pixels mapped into the panoramic image must be in order so that the image will not be distorted, the image will be split into 4 sections of 90° each, as shown in Figure 3(b). Each section will perform the calculation based on the moving direction of the circle. For example, for a circle drawn starting from the right side in a counter-clockwise direction, the pixels in the section at the upper right part, will be taken and calculated. The pixels are calculated one by one from the most right pixel to most left pixel starting from the bottom part of the section. Then the same calculation manner is repeated for each layer of pixels till the upper part of the section. On the other hand, for the lower left part of the section, the calculation will go from the most left pixel to the most right pixel, starting from the top of the section and ends at the bottom.

However, since the considered pixels will be non-uniform for different circle of different radius as shown in Figure 4(a), a re-sampling process is needed to standardize the pixels in every row of the panoramic image. Therefore, after every pixel in the whole omni-image is mapped onto the panoramic image plane, spacing will be inserted in between pixels of each row (as shown in Figure 4(b)) in order to standardize the resolution of the panoramic image. This will generate a standard resolution of panoramic image. However, since spacing are generally empty pixels with no data information, a row with very little pixels will be hard to understand. Therefore, the pixels will be duplicated over the spacing instead of inserting empty pixels into it, and an understandable uniform resolution panoramic image can be generated.

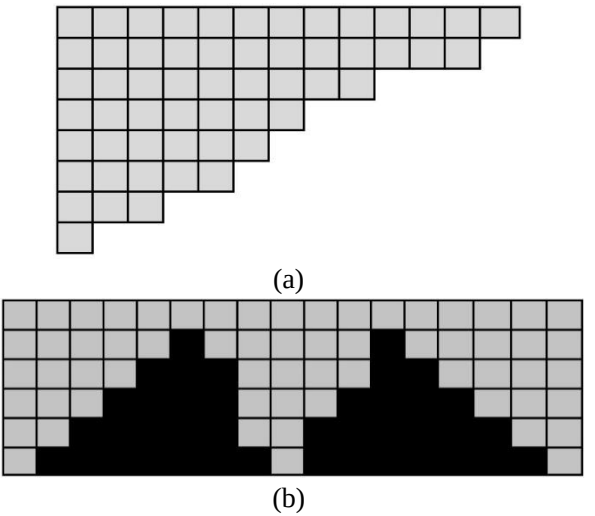


Figure 4. (a) Non-uniform resolution of panoramic image, (b) Spacing is inserted in between pixels, denoted by black dots.

3.2.2. Pano-Mapping Table Method

This method uses a table which is called the pano-mapping table to process the image conversion. Pano-mapping table will be created “once and for all” and it consists of many co-ordinates corresponding to the co-ordinates taken from the omnidirectional mirror image. These co-ordinates will then be mapped into a new panoramic image respectively. This method is practically used in omnidirectional visual tracking [30], and in any unwarping process of omni-images taken by almost any kind of omni-cameras prior to requiring any knowledge about the camera parameters in advance, as proposed by Jeng, Tsai andWu [27, 31].

In Pano-mapping table method, it is required to select 5 landmark points from the omnidirectional image first. These points will be taken from the same line which is drawn from the center of the omni-image to the circumference of the image. This line is also known as the radius of the image. Five points in between both ends of this line will be picked, and the value corresponding to their radius from the center will be obtained. Then, these values are being used to find the 5 coefficients of a_0 through a_4 in the 'radial stretching function', $f_r(p)$ described by the following 4th-degree polynomial function of:

$$r = f_r(p) = a_0 + a_1 p_1 + a_2 p_2 + a_3 p_3 + a_4 p_4 \quad (1)$$

where r corresponds to the radius, p is the particular radius for each of the 5 points taken, and $a_0 \sim a_4$ are five coefficients to be estimated using the values obtained from the landmark points.

Once the 5 coefficients are obtained, the pano-mapping table (T_{MN}) can be generated. The size of the table will be determined manually, by setting it to a table of size M x N. Hence, in order to fill up a table with M x N entries, the landmark point (p), which correspond to the radius of the omnidirectional mirror image, will be divided into M separated parts, and the angle (θ) will be divided into N parts as follows:

$$p_{ij} = i \times \frac{radius}{M} \tag{2}$$

$$\theta_{ij} = j \times \frac{360^0}{N} \tag{3}$$

The calculation will be processed by taking the first point where i = 1 and j = 1, which gives $p_{11} = radius/M$, and $\theta_{11} = 360^0/N$. The value of p_{ij} will then be substituted into the "radial stretching function" in order to obtain the particular radius at that particular landmark point. This radius obtained, will be substituted into the equation below and then rounded up, in order to get the corresponding co-ordinates in the omnidirectional image.

$$v = r\cos\theta \tag{4}$$

$$u = r\sin\theta \tag{5}$$

where v and u correspond to the x and y co-ordinates of the omnidirectional mirror image. The coordinate (u, v) obtained is inserted into the pano-mapping table ($T_{MN} = T_{ij}$). The u and v will then be processed for N times by increasing j for N times to obtain different angle (θ). This is done to determine all the co-ordinates corresponding to the value of the landmark point. These co-ordinates obtained, are inserted into the table of i = 1 with their corresponding j = 1 to j = N. Next, the i will be increased by 1, and the process is repeated for j = 1 to j = N to determine all co-ordinates related to i = 2. This i will be repeated for M times, and a table of M x N entries with all the co-ordinates can be generated. Lastly, the co-ordinates in each of the entries are taken one by one, in order to map each and every pixel of the omnidirectional

mirror image into a new panoramic-image. The conversion is completed upon the end of the table mapping.

3.2.3. Log-Polar Mapping Method

Log-polar mapping is a type of spatially-variant image representation whereby pixel separations increase linearly with distance. It enables the concentration of computational resource on a region of interest as well as maintaining the information from a wider view. This method is implemented by applying log-polar geometry representations. The captured omnidirectional mirror image will first be sampled from a Cartesian form into a log-polar form by using spatially-variant grid. The spatially-variant grid representing log-polar mapping will be formed by i number of concentric circles with N number of samples, and the omnidirectional mirror image will then be un-warped into a panoramic image in another Cartesian form.

This method is practically used in robust image registration [32], or in robotic vision, particularly in visual attention, target tracking, ego motion estimation, and 3D perception [33]. It is also practiced in vision-based navigation, environmental representations and imaging geometries [34], proposed by José Santos-Victor, and Alexandre Bernardino. In log-polar mapping method, the center pixel for log-polar sampling is calculated by:

$$p(x_i, y_i) = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} \tag{6}$$

$$\theta(x_i, y_i) = \left(\frac{N}{2\pi}\right) tan^{-1} \frac{y_i - y_c}{x_i - x_c} \tag{7}$$

And the center pixel for log polar mapping is calculated by

$$x_0(p, \theta) = pcos\theta + x_c \tag{8}$$

$$y_0(p, \theta) = pcos\theta + y_c \tag{9}$$

where x_c, y_c is the center point of our original Cartesian form's co-ordinate, and N is the number of samples in each and every concentric circles taken. The original (x_i, y_i) in Cartesian form is sampled into log-polar co-ordinate of (p, θ), as shown in Figure 5. The center point is calculated by using equations stated above to get the respective p and θ, which cover a region of the original Cartesian pixels of radius:

$$r_n = br_{n-1} \tag{10}$$

and

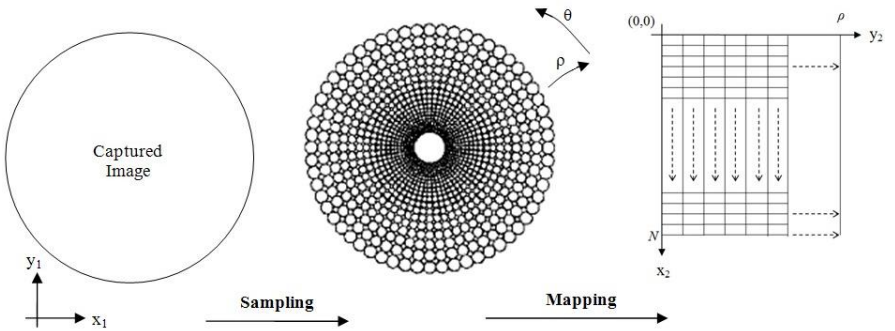


Figure 5. Process of log-polar mapping.

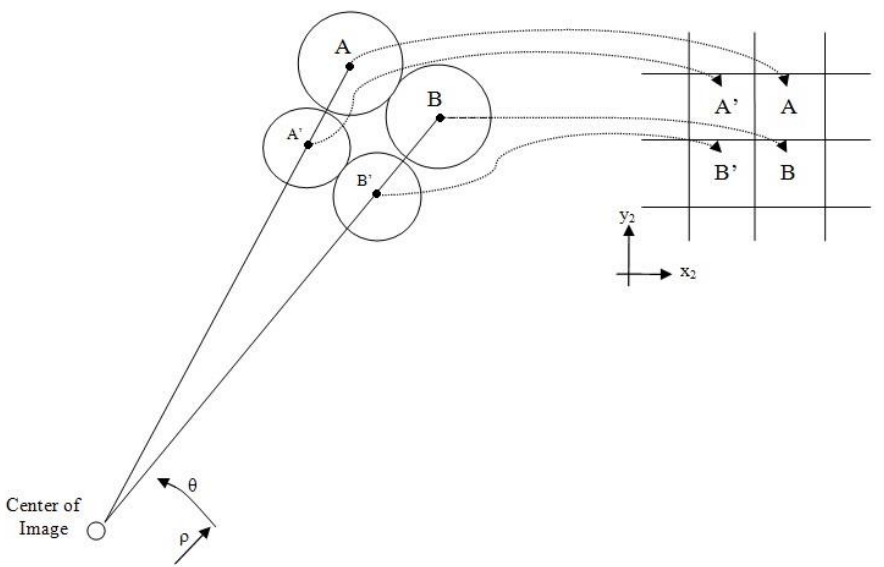


Figure 6. Circular Sampling Structure and the unwarping process.

$$b = \frac{N+\pi}{N-\pi} \tag{11}$$

where r is the sampling circle radius and b is the ratio between 2 apparent sampling circles. Figure 6 shows the circular sampling structure and the unwarping process done by using the log-polar mapping method [32]. The

mean value of pixels within each and every circular sampling is calculated and it will be assigned to the center point of the circular sampling. The process will then continue by mapping the mean value of log-polar pixel (p, θ) into another Cartesian form using $x_0(p,\theta) = pcos\theta + x_c$ and $y_0(p,\theta) = pcos\theta + y_c$ as stated above. Finally, the un-warping process will be completed at the end of the mapping.

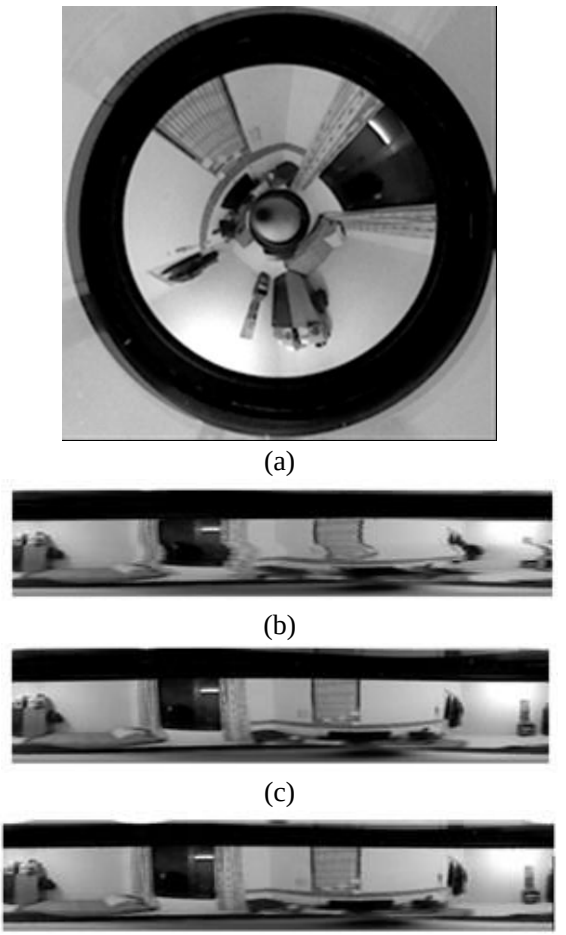
3.2.4. Performance Evaluation

This subsection reports the performance evaluation for different unwarping methods. Few important factors selected for the performance evaluation include: resolution of the image generated, quality of image, algorithm used in performing the un-warping process, complexity, processing time, and data compression. A captured omnidirectional mirror image as shown in Figure 7 will be used to test the unwarping methods.

A. *Resolution of the image generated*: The resolution of each generated panoramic image using log-polar mapping method, discrete geometry techniques and pano-mapping table method is being discussed in this subsection. The log-polar mapping method provides smaller resolution of dimension equals to 1/4 fold of the omnidirectional mirror image. As for the DGT method and pano-mapping table method, the resolution of the panoramic-image produced can be as large as the length of the perimeter of the omnidirectional mirror image, with the width being equal to the radius of the omnidirectional mirror image. However, since the images had been re-scaled for viewing purposes, the difference is not obvious in this paper.
B. *Quality of image*: Since the images are re-scaled, the difference in quality is not apparent as well. However, pano-mapping table method is found to produce the highest quality of image, followed by the log-polar mapping method and lastly, the DGT method.
C. *Algorithm used in performing the un-warping process*: In log-polar mapping algorithm, the omnidirectional mirror image is in the form of a number of sectors, and each sector consists of a group of pixels that will be extracted sector by sector accordingly into a rectangular form of panoramic image. Meanwhile, for the DGT method, pixel by pixel is to be extracted and arranged into a rectangular form image before they are re-produced, or duplicated, in order to standardize the number of pixels available in each row of the panoramic image. For the pano-mapping table method, a table is created at initialization to

indicate the co-ordinates of the pixels to be extracted from the omnidirectional mirror image. Once the table is completed, it will be used over and over again to map each of the pixel at that particular co-ordinate, one by one, from the omnidirectional mirror image into a panoramic image.

D. *Complexity*: Table 4 shows the Big-O complexity of log-polar mapping method, DGT method, and pano-mapping table method.



(a)

(b)

(c)

(d)

Figure 7. (a) Sample of omnidirectional mirror image, (b) Panoramic image generated by using DGT method, (c) Panoramic image generated by using pano-mapping table method, (d) Panoramic image generated by using log-polar method.

Table 4. Big-O Complexity

	DGT	**Log-polar mapping**	**Pano-mapping table**
Addition	$O(XY^2)$	$O(X^2Y^2)$	$O(Y^2)$
Subtraction			
Multiplication	$O(Y)$	$O(X^2)$	$O(Y^2)$
Division			
Logarithmic	-	$O\left(\frac{\log(X)}{\log(Y)}\right)$	-

where X = length of the panoramic image = perimeter of the omnidirectional mirror image taken into consideration, and Y = height of the panoramic image = radius of the omnidirectional mirror image taken into consideration.

E. *Processing time*: The processing time for the 3 unwarping methods to transform an omnidirectional mirror image into a panoramic image is calculated using the matlab function “cputime”. The program is being processed 5 times on 5 different images, and the average processing time is computed. It is found that the pano-mapping table method has the lowest processing time, with 1.220seconds, followed by log-polar mapping method with 2.003seconds, and DGT method with 3.426seconds.

F. *Data compression:* The generated panoramic image produced by the log-polar mapping method, DGT method and pano-mapping table method has the resolution of 473 x 114, 1472 x 235, and 1146 x 243 respectively. The original omnidirectional image has the resolution 473 x 473. From the output resolution, it is clear that log-polar mapping has the highest compression rate, with up to 4 folds image compression as compared to DGT method (0.65 folds –image expansion) and pano-mapping table method (0.80 folds–image expansion).

In terms of *resolution of the image generated*, although the image generated by the DGT method and pano-mapping table method have higher resolution as compared to the image generated by the log-polar mapping method, these 2 methods seem to elongate the actual size of the image. In other words, these methods tend to make the objects in the image looked 'broader' than their original sizes. Due to this elongation, it will be harder to examine the picture and the objects, as the sense of the size had been eliminated. As for the log-polar mapping method, the extension is not much,

and it is not as obvious as the DGT method and pano-mapping table method. In terms of *quality of image*, the pano-mapping table method has the highest image quality among the 3 methods, followed by the log-polar mapping method with a slightly lower image quality (but still within an acceptable range), and lastly the DGT method with the lowest image quality. In terms of *algorithm used in performing the un-warping process*, the pano-mapping table method has the most simple algorithm, followed the log-polar mapping method with a slightly more complex algorithm, and lastly, the DGT method with the most complicated algorithm. In terms of *complexity*, it is found that the pano-mapping table method has the least complexity, followed by the DGT method, and lastly the log-polar mapping method. In terms of *processing time*, on average, the pano-mapping table method has the lowest processing time (to transform an omnidirectional mirror image into a panoramic image), followed by the log-polar mapping method and DGT method. In terms of *data compression*, the log-polar mapping method has the highest data compression rate as compared to the pano-mapping table method and DGT method. High compression rate is important to preserve CPU's memory, and the memory available is usually very limited.

After comparing and weighing each category of performance for the 3 unwarping methods mentioned above, we have decided to implement the log-polar mapping method as the unwarping method to be used in this work. This is due to its all rounded performance that is suitable for our work.

4. AUTOMATIC TRESPASSER AND FAINT DETECTION ALGORITHM

The proposed trespasser and faint detection algorithm implement image processing, omnidirectional viewing through optical approach and log-polar mapping unwarping method. These are done to achieve automatic detection in an indoor environment with good light condition for both trespassing and fainting. The automatic trespasser detection algorithm will be named as Extreme Point Curvature Analysis Algorithm while the automatic faint detection algorithm will be named as Integrated Body Contours Algorithm. Following will be the detailed explanation on the algorithm's execution.

Step 1: *Image acquisition*: Image of the monitored area will be taken using the setup proposed in Section 3.1. The image taken can be seen in Figure 8.

Figure 8. Image acquisition.

Step 2: *Image unwarping*: The image taken in step 1 will be un-warped into panoramic view using the log-polar unwarping method mentioned in Section 3.2.3 for ease of analysis and image processing. The un-warped image can be seen in Figure 9 below.

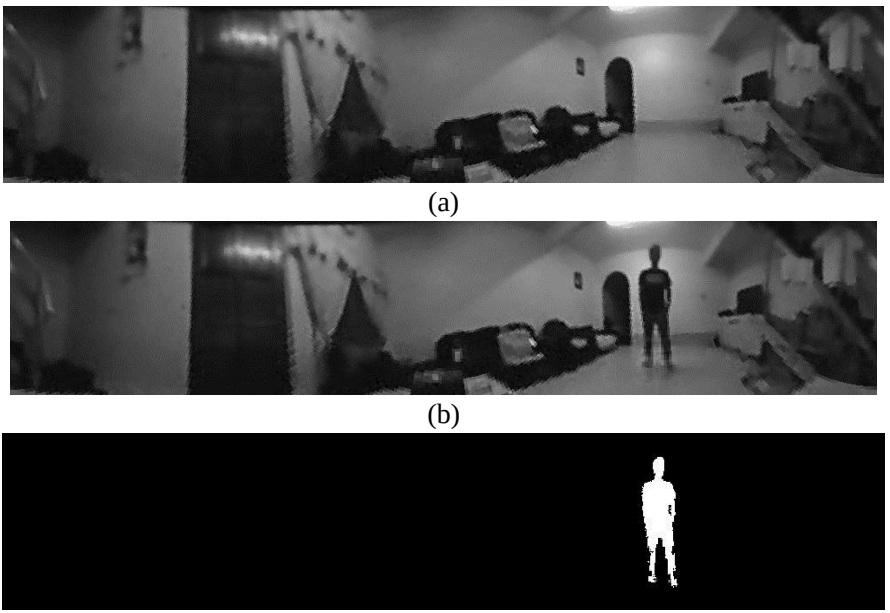
Figure 9. Image unwarping.

Step 3: *Object extraction*: Background subtraction method is used to obtain moving objects by checking the difference between current image (I_C) and background image (I_B). Initially, image of the monitored area without any moving objects (human, animals and etc.) is taken and stored as background image (I_B). Then, during usage of the system, any image taken also known as current image (I_C), will be compared with the stored background image (I_B), pixel by pixel using the following rules:

Convert both current image (I_C) and background image (I_B) into grayscale image. Check:

IF $(X_{I_C}, Y_{I_C}) - (X_{I_B}, Y_{I_B}) > T_{BS}$ (background subtraction threshold);

THEN (X_{I_R}, Y_{I_R}) is set to be white pixel (value 1 for binary);



(a)

(b)

(c)

Figure 10. (a) Background image, (b) Current image, (c) Resultant image.

where, I_R = resultant image.

The extracted moving objects will be a white blob. Finally, the resultant image (I_R) is stored for further processing. The object extraction process is shown in Figure 10.

Step 4: *Morphological steps*: From the resultant image (I_R) taken in step 3, any noise with the number of pixels less than 300 pixels will be removed and filled with black pixels. Then morphological closing will also be performed on the remaining objects.

Step 5: *Features extraction*: The boundaries of the object are obtained and stored as $(X_a, Y_a)_{obj}$, where a = the location of a pixel in the boundary and obj = the number of objects in the image. For each $(X_a, Y_a)_{obj}$, features extraction in Section 4.1 is performed to obtain the head

detection, leg detection, ellipse fitting's ratio and orientation. Then either Mode 1 or Mode 2 is selected. Mode 1 would be the trespasser detection algorithm (Extreme Point Curvature Analysis Algorithm) in Section 4.2 and Mode 2 would be the faint detection algorithm (Integrated Body Contours Algorithm) in Section 4.3.

4.1. Features Extraction

4.1.1. Head Detection

The head detection includes Center Head Position, Right Head Position and Left Head Position.

4.1.1.1. Center Head Position

Step 1: *Top peak point*: From every pixel of the object's boundary, check:

IF $(Y_a - Y_{a-1} < 0)$ & $(Y_{a+1} - Y_a > 0)$;

THEN set $(X_a, Y_a) = F_n$;

where n = number of peak points in the object.

NEXT set F_n with minimum y-coordinate as F_{tpp} or Top Peak Point.

Step 2: *Side turning point*: Following the boundary, from F_{tpp}, search clockwise and check:

IF $(X_a - X_{a-1} > 0)$ & $(X_{a+1} - X_a < 0)$;

THEN set $(X_a, Y_a) = F_{right}$;

NEXT, from F_{right}, search clockwise along the boundary and check:

IF $(X_a - X_{a-1} < 0)$ & $(X_{a+1} - X_a > 0)$;

THEN set $(X_a, Y_a) = F_{rightN}$.

Similarly, for left side, following the boundary, from F_{tpp}, search anticlockwise and check:

IF $(X_{a+1} - X_a > 0)$ & $(X_a - X_{a-1} < 0)$;

THEN set $(X_a, Y_a) = F_{left}$;

NEXT, from F_{left}, search anticlockwise along the boundary and check:

IF $(X_{a+1} - X_a < 0)$ & $(X_a - X_{a-1} > 0)$;

THEN set $(X_a, Y_a) = F_{leftN}$.

*Only the first turning point encountered is recorded and used.

Step 3: *Check head symmetry and position*: Check the following conditions:

IF $[(HR_{height}/HL_{height}) < 2]$ & $[(HL_{height}/HR_{height}) < 2]$;

AND $C_{distance} < (2*N_{distance})$;

where HR_{height} = vertical distance between F_{tpp} and F_{rightN},
HL_{height} = vertical distance between F_{tpp} and F_{leftN},
$N_{distance}$ = horizontal distance between F_{leftN} and F_{rightN},
$C_{distance}$ = [horizontal distance between F_{tpp} and F_{leftN}] – [$N_{distance}/2$],
THEN proceed to Step 4,
ELSE H_{center} = 0 or 'center head position' is not detected and skip to Section 4.1.1.2.

Step 4: *Check head curve*: Check the following conditions along the boundaries:

For every pixel from F_{tpp} to F_{right},
IF $(X_a - X_{a-1} > 0)$ & $(Y_a - Y_{a-1} >= 0)$;
THEN C_{total} = Count + 1;
where Count = 0 by default,
For every pixel from F_{right} to F_{rightN},
IF $(X_a - X_{a-1} < 0)$ & $(Y_a - Y_{a-1} >= 0)$;
THEN C_{total} = Count + 1;
For every pixel from F_{tpp} to F_{left},
IF $(X_a - X_{a-1} > 0)$ & $(Y_a - Y_{a-1} <= 0)$;
THEN C_{total} = Count + 1;
For every pixel from F_{left} to F_{leftN},
IF $(X_a - X_{a-1} < 0)$ & $(Y_a - Y_{a-1} <= 0)$;
THEN C_{total} = Count + 1;
Finally check:
IF $C_{total} > C_{thresholdC}$,
THEN H_{center} = 1 or 'center head position' is detected. Section 4.1.1.2 and Section 4.1.1.3 is ignored.
ELSE H_{center} = 0 and proceed to Section 4.1.1.2.
Center head position can be seen in Figure 11.

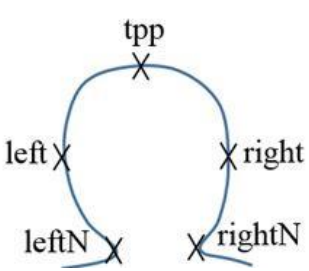


Figure 11. Center head position.

4.1.1.2. Right HEAD Position

Step 1: *Right peak point*: From every pixel of the object's boundary, check:

IF $(X_a - X_{a-1} > 0)$ & $(X_{a+1} - X_a < 0)$;

THEN set $(X_a, Y_a) = F_n$;

where n = number of right peak points in the object.

NEXT set F_n with maximum x-coordinate as F_{rpp} or Right Peak Point.

Step 2: *Side turning point*: Following the boundary, from F_{rpp}, search clockwise and check:

IF $(Y_a - Y_{a-1} > 0)$ & $(Y_{a+1} - Y_a < 0)$;

THEN set $(X_a, Y_a) = F_{bottom}$;

NEXT, from F_{bottom}, search clockwise along the boundary and check:

IF $(Y_a - Y_{a-1} < 0)$ & $(Y_{a+1} - Y_a > 0)$;

THEN set $(X_a, Y_a) = F_{bottomN}$.

Similarly, for top side, following the boundary, from F_{rpp}, search anticlockwise and check:

IF $(Y_{a+1} - Y_a > 0)$ & $(Y_a - Y_{a-1} < 0)$;

THEN set $(X_a, Y_a) = F_{top}$;

NEXT, from F_{top}, search anticlockwise along the boundary and check:

IF $(Y_{a+1} - Y_a < 0)$ & $(Y_a - Y_{a-1} > 0)$;

THEN set $(X_a, Y_a) = F_{topN}$.

*Only the first turning point encountered is recorded and used.

Step 3: *Check head symmetry and position*: Check the following conditions:

IF $[(HB_{height}/HT_{height}) < 2]$ & $[(HT_{height}/HB_{height}) < 2]$;

AND $C_{distance} < (2*N_{distance})$;

where HB_{height} = horizontal distance between F_{rpp} and $F_{bottomN}$,

HT_{height} = horizontal distance between F_{rpp} and F_{topN},

$N_{distance}$ = vertical distance between $F_{bottomN}$ and F_{topN},

$C_{distance}$ = [vertical distance between F_{rpp} and F_{topN}] – $[N_{distance}/2]$,

THEN proceed to Step 4,

ELSE $H_{right} = 0$ or 'right head position' is not detected and skip to Section 4.1.1.3.

Step 4: *Check head curve*: Check the following conditions along the boundaries:

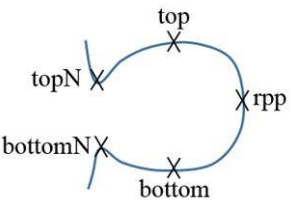


Figure 12. Right head position.

For every pixel from F_{rpp} to F_{bottom},
IF $(X_a - X_{a-1} <= 0)$ & $(Y_a - Y_{a-1} > 0)$;
THEN C_{total} = Count + 1;
where Count = 0 by default,
For every pixel from F_{bottom} to $F_{bottomN}$,
IF $(X_a - X_{a-1} <= 0)$ & $(Y_a - Y_{a-1} < 0)$;
THEN C_{total} = Count + 1;
For every pixel from F_{rpp} to F_{top},
IF $(X_a - X_{a-1} >= 0)$ & $(Y_a - Y_{a-1} > 0)$;
THEN C_{total} = Count + 1;
For every pixel from F_{top} to F_{topN},
IF $(X_a - X_{a-1} >= 0)$ & $(Y_a - Y_{a-1} < 0)$;
THEN C_{total} = Count + 1;
Finally check:
IF $C_{total} > C_{thresholdR}$,
THEN H_{right} = 1 or 'right head position' is detected. Section 4.1.1.3 is ignored.
ELSE H_{right} = 0 and proceed to Section 4.1.1.3.
Right head position can be seen in Figure 12.

4.1.1.3. Left Head Position

Step 1: *Left peak point*: From every pixel of the object's boundary, check:
IF $(X_a - X_{a-1} < 0)$ & $(X_{a+1} - X_a > 0)$;
THEN set $(X_a, Y_a) = F_n$;
where n = number of peak points in the object.
NEXT set F_n with minimum x-coordinate as F_{lpp} or Left Peak Point.

Step 2: *Side turning point*: Following the boundary, from F_{lpp}, search clockwise and check:

IF $(Y_a - Y_{a-1} < 0)$ & $(Y_{a+1} - Y_a > 0)$;
THEN set $(X_a, Y_a) = F_{top}$;
NEXT, from F_{top}, search clockwise along the boundary and check:
IF $(Y_a - Y_{a-1} > 0)$ & $(Y_{a+1} - Y_a < 0)$;
THEN set $(X_a, Y_a) = F_{topN}$.
Similarly, for left side, following the boundary, from F_{lpp}, search anticlockwise and check:
IF $(Y_{a+1} - Y_a < 0)$ & $(Y_a - Y_{a-1} > 0)$;
THEN set $(X_a, Y_a) = F_{bottom}$;
NEXT, from F_{bottom}, search anticlockwise along the boundary and check:
IF $(Y_{a+1} - Y_a > 0)$ & $(Y_a - Y_{a-1} < 0)$;
THEN set $(X_a, Y_a) = F_{bottomN}$.
*Only the first turning point encountered is recorded and used.

Step 3: *Check head symmetry and position*: Check the following conditions:
IF $[(HB_{height}/HT_{height}) < 2]$ & $[(HT_{height}/HB_{height}) < 2]$;
AND $C_{distance} < (2*N_{distance})$;
where HB_{height} = horizontal distance between F_{lpp} and $F_{bottomN}$,
HT_{height} = horizontal distance between F_{lpp} and F_{topN},
$N_{distance}$ = vertical distance between $F_{bottomN}$ and F_{topN},
$C_{distance}$ = [vertical distance between F_{lpp} and $F_{bottomN}$] – $[N_{distance}/2]$,
THEN proceed to Step 4,
ELSE $H_{left} = 0$ or 'left head position' is not detected.

Step 4: *Check head curve*: Check the following conditions along the boundaries:
For every pixel from F_{lpp} to F_{top},
IF $(X_a - X_{a-1} >= 0)$ & $(Y_a - Y_{a-1} < 0)$;
THEN C_{total} = Count + 1;
where Count = 0 by default,
For every pixel from F_{top} to F_{topN},
IF $(X_a - X_{a-1} >= 0)$ & $(Y_a - Y_{a-1} > 0)$;
THEN C_{total} = Count + 1;
For every pixel from F_{lpp} to F_{bottom},
IF $(X_a - X_{a-1} <= 0)$ & $(Y_a - Y_{a-1} < 0))$;

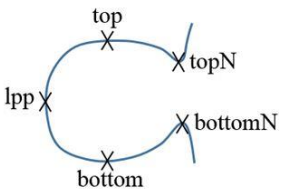


Figure 13. Left head position.

THEN C_{total} = Count + 1;
For every pixel from F_{bottom} to $F_{bottomN}$,
IF $(X_a - X_{a-1} <= 0)$ & $(Y_a - Y_{a-1} > 0)$;
THEN C_{total} = Count + 1;
Finally check:
IF $C_{total} > C_{thresholdL}$,
THEN H_{left} = 1 or 'left head position' is detected.
ELSE H_{left} = 0.
Left head position can be seen in Figure 13.

4.1.2. Leg Detection

Following are the steps to detect the Leg curve:

Step 1: *Obtain start point*: Find the lowest point of the object with maximum y-coordinate and minimum x-coordinate. Record the point as Leg Start Point, $F_{legSP} = (X_{legSP}, Y_{legSP})$. Next, find the mean x-coordinate and mean y-coordinate of the object and set that point as Object Middle Point, $F_{objMP} = (X_{objMP}, Y_{objMP})$.

Step 2: *Obtain turning point*:

i. ***IF*** $(X_{legSP} - X_{objMP} < 0)$, search anticlockwise along the boundary from F_{legSP} and check:
IF $(Y_a - Y_{a-1} < 0)$ & $(Y_{a+1} - Y_a > 0)$;
THEN set $(X_a, Y_a) = F_{legTP}$ and proceed to Step 3(i);

ii. ***IF*** $(X_{legSP} - X_{objMP} > 0)$, search clockwise along the boundary from F_{legSP} and check:
IF $(Y_a - Y_{a-1} < 0)$ & $(Y_{a+1} - Y_a > 0)$;
THEN set $(X_a, Y_a) = F_{legTP}$ and proceed to Step 3(ii);

Step 3: *Obtain end point:*

i. From F_{legTP}, search anticlockwise along the boundary and check:

IF $(Y_a - Y_{a-1} > 0)$;
THEN set $(X_a, Y_a) = F_{legEP}$ and proceed to Step 4(i);

ii. From F_{legTP}, search clockwise along the boundary and check:
IF $(Y_{a+1} - Y_a < 0)$,
THEN set $(X_a, Y_a) = F_{legEP}$ and proceed to Step 4(ii);

Step 4: *Check leg symmetry and curve:*

Let Lheight1= vertical distance between F_{legTP} and F_{legSP}.
Lheight2= vertical distance between F_{legTP} and F_{legEP}.

i. ***IF*** $[(L_{height1}/L_{height2}) < 2]$ & $[(L_{height2}/L_{height1}) < 2]$ is true, check the following conditions. Else, $L_{presence} = 0$;
For every pixel from F_{legSP} to F_{legTP},
IF $(X_a - X_{a-1} <= 0)$ & $(Y_a - Y_{a-1} > 0)$;
THEN C_{total} = Count + 1;
where Count = 0 by default,
For every pixel from F_{legTP} to F_{legEP},
IF $(X_a - X_{a-1} <= 0)$ & $(Y_a - Y_{a-1} < 0)$;
THEN C_{total} = Count + 1;
Finally check:
IF $C_{total} > C_{leg}$;
THEN $L_{presence} = 1$ or 'leg presence' is detected;
ELSE $L_{presence} = 0$.

ii ***IF*** $[(L_{height1}/L_{height2}) < 2]$ & $[(L_{height2}/L_{height1}) < 2]$ is true, check the following conditions. Else, $L_{presence} = 0$;
For every pixel from F_{legSP} to F_{legTP},
IF $(X_a - X_{a-1} <= 0)$ & $(Y_a - Y_{a-1} < 0)$;
THEN C_{total} = Count + 1;
where Count = 0 by default,
For every pixel from F_{legTP} to F_{legEP},
IF $(X_a - X_{a-1} <= 0)$ & $(Y_a - Y_{a-1} > 0)$;
THEN C_{total} = Count + 1;
Finally check:
IF $C_{total} > C_{leg}$;
THEN $L_{presence} = 1$ or 'leg presence' is detected;
ELSE $L_{presence} = 0$.

Leg curve can be seen in Figure 14.

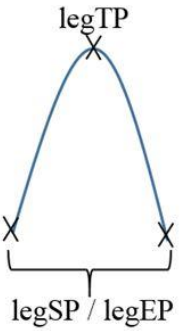


Figure 14. Leg curve.

4.1.3. Ellipse Fitting

Ellipse fitting method [35 - 38] is being used to obtain the ellipse's fitting ratio and orientation.

The moments for binary object blobs can be obtained by using:

$$m_{pq} = \sum_{x,y} x^p y^q f(x,y) \tag{12}$$

where, p, q = 0, 1, 2, 3 …

The center of ellipse: $\bar{x} = m_{10}/m_{00}$ and $\bar{y} = m_{01}/m_{00}$ can be derived from the first-order and zero-order spatial moments. Then the central moment can be calculated with:

$$\mu_{pq} = \sum_{x,y} (x-\bar{x})^p (y-\bar{y})^q f(x,y) \tag{13}$$

where, p,q = 0, 1, 2, 3 …

Using the central moment, the ellipse's orientation or the angle between the major axis of the person and the horizontal axis x can be computed as follows:

$$\theta = \frac{1}{2}\arctan\left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}\right) \tag{14}$$

Then to recover the major semi-axis a and the minor semi-axis b, the greatest moments of inertia, I_{max} and the least moments of inertia I_{min} must be computed. They can be calculated by evaluating the eigenvalues of the covariance matrix:

$$J = \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix} \tag{15}$$

where the eigenvalues I_{max} and I_{min} are given as:

$$I_{max} = \frac{\mu_{20} + \mu_{02} + \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}}{2} \tag{16}$$

$$I_{min} = \frac{\mu_{20} + \mu_{02} - \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}}{2} \tag{17}$$

Finally, the major semi-axis a and the minor semi-axis b of the best fitting ellipse can be calculated as follows:

$$a = (4/\pi)^{\frac{1}{4}} \left[\frac{(I_{max})^3}{I_{min}}\right]^{1/8} \tag{18}$$

$$b = (4/\pi)^{\frac{1}{4}} \left[\frac{(I_{min})^3}{I_{max}}\right]^{1/8} \tag{19}$$

Hence, the ellipse fitting's ratio would be $R_{object} = (a/b)$ and the ellipse fitting's orientation would be $O_{object} = (\theta)$ as mentioned above.

4.2. Extreme Point Curvature Analysis Algorithm

By analyzing the human body shapes, a consistent characteristic can be found to exist in most people which is the extreme point curvature. Extreme point curvature is the curve that is present on a human body at the most top, bottom, left and right position. When a human is in an upright position whether in standing or walking situation, top curve would be the head shape and bottom curve would be the leg shape. Meanwhile, the left curve and right curve would be the head shape when a human is bending down. Hence, by detecting the presence of a head shape (top, left or right) or a leg shape on an object in the image using the extreme point curvature analysis algorithm, a human presence can be determined. This system/mode will only be activated when the owner leaves, and there is no one in the monitored house. Therefore, any human detected, will be assumed as a trespasser. Trespasser can be detected by using the following conditions:

From the head detection and leg detection information obtained in Section 4.1, check:

IF (H_{center} = 1) OR (H_{right} = 1) OR (H_{left} = 1) OR ($L_{presence}$ = 1);
THEN trespasser is detected and alarm is activated;
ELSE trespasser is not detected;

This algorithm can differentiates between human and non-human objects because it detects head curve and leg curve which are not present in most non-human objects. The background model is also updated every 3s whenever a trespasser is not detected to overcome changing background problem such as illumination change.

4.3. Integrated Body Contours Algorithm

The integrated body contours algorithm is the combination of head curve, leg curve, ellipse fitting's ratio and orientation information to determine human postures. The postures include standing, bending, sitting and lying down. By detecting specific posture, the algorithm is able to reduce false alarm (e.g., detecting bend as lie posture) and increase the accuracy. Fainting can be detected by using the following steps:

Step 1: *Determine posture*: Based on the head detection, leg detection, ellipse fitting's ratio and orientation obtained in Section 4.1, the posture of the object blob can be determined according to the conditions stated in Table 5.

The conditions in Table 5 are obtained by monitoring the human body shapes when a person is performing stand, bend, sit and lie posture. The conditions are explained in Section 5.1.3.

Step 2: *Determine faint*: Check:

IF lie posture is detected for consecutive 5 frames;
THEN human faint is detected and alarm is activated;

Faint is assumed to have occurred when a lie posture is detected for more than 5 seconds. With an average processing time of 1 second per frame in the proposed system, 5 frames satisfy the fainting time period.

Table 5. Conditions for postures

Posture	Conditions
Stand	If (H_{center} = 1) & ($L_{presence}$ = 1);
	If (H_{center} = 1) & ($L_{presence}$ = 0) & ($R_{standMin}$ < R_{object} < $R_{standMax}$) & ($O_{standMin}$ < O_{object} < $O_{standMax}$)
Bend	If (H_{center} = 0) & ($L_{presence}$ = 1) & ($R_{bendMin}$ < R_{object} < $R_{bendMax}$) & ($O_{bendMin}$ < O_{object} < $O_{bendMax}$)
	If [(H_{right} = 1) or (H_{left} = 1)] & ($L_{presence}$ = 0) & ($R_{bendMin}$ < R_{object} < $R_{bendMax}$) & ($O_{bendMin}$ < O_{object} < $O_{bendMax}$)
Sit	If (H_{center} = 1) & ($L_{presence}$ = 0) & (R_{sitMin} < R_{object} < R_{sitMax}) & (O_{sitMin} < O_{object} < O_{sitMax})
Lie	If (H_{center} = 0) & (H_{right} = 0) & (H_{left} = 0) & ($L_{presence}$ = 0) & ($R_{lieMin1}$ < R_{object} < $R_{lieMax1}$) & ($O_{lieMin1}$ < O_{object} < $O_{lieMax1}$)
	If [(H_{right} = 1) or (H_{left} = 1)] & ($L_{presence}$ = 0) & ($R_{lieMin2}$ < R_{object} < $R_{lieMax2}$) & ($O_{lieMin2}$ < O_{object} < $O_{lieMax2}$)
	If (H_{center} = 0) & ($L_{presence}$ = 0) & ($R_{lieMin2}$ < R_{object} < $R_{lieMax2}$) & ($O_{lieMin3}$ < O_{object} < $O_{lieMax3}$)

5. EXPERIMENTAL RESULTS AND DISCUSSION

5.1. Parameters Optimization

Parameters optimizations are done to obtain the optimal results for the extreme point curvature analysis algorithm and integrated body contours algorithm. The parameters include background subtraction threshold, head & leg detection threshold, posture's conditions and ellipse fitting's ratio & orientation as stated in the following sections.

5.1.1. Background Subtraction Threshold

Background subtraction threshold (T_{BS}) is the minimum value of a pixel to be considered as a white pixel (value 1 in binary) in the resultant image (I_R) after the subtraction between pixels of current image (I_C) and pixels of background image (I_B), as mentioned in Step 3 Section 4. Background subtraction threshold greatly affects the outcome of the resultant image as to whether the acquired object blob is true to reality. Considering background subtraction threshold depends on the surrounding's brightness, the experiments for this work are done in a living room of a household with good

light condition. Background subtraction threshold is tested from the value of 0 to 255 (a step size of 10) on 1000 frames of images to observe the consistency of the threshold in providing a good object blob. The validation is determined by how similar the acquired object blob is to the real object. Figure 15 shows the results for the background subtraction threshold value:

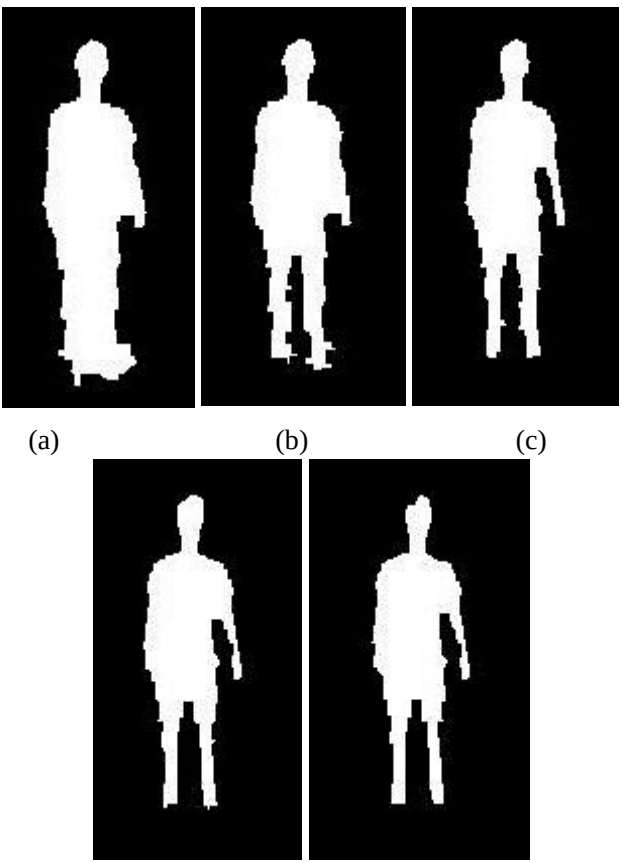


Figure 15. (a) T_{BS} = 40, (b) T_{BS} = 50, (c) T_{BS} = 60, (d) T_{BS} = 70, (e) T_{BS} = 80.

From Figure 15, we can observe that the object blob in image with T_{BS} = 40 has a deformed shape, where the leg and arm parts are totally covered by white pixels. Then, the object blob in image with T_{BS} = 50 still has its arm (right side) covered by white pixels and there are a lot of noises around the leg region. Object blob in image with T_{BS} = 60 resembles the actual object the

most. Lastly, the object blob in image with T_{BS} = 70 and 80 have smaller sizes around the arms, legs and head region as compared to the actual object. Hence. T_{BS} = 60 is selected as the ideal background subtraction threshold value.

5.1.2. Head and Leg Detection Threshold

The head and leg detection thresholds are the minimum values that a curve has to satisfy to be considered as a head detection or leg detection. There are a total of 4 thresholds to be determined which include center head position threshold ($C_{thresholdC}$), right head position threshold ($C_{thresholdR}$), left head position threshold ($C_{thresholdL}$) and leg curve threshold (C_{leg}). Head and leg detection in Section 4.1, are carried out using threshold values of 0.1 to 1.0 with a step size of 0.1. They are tested on a sample of 512 images to determine the best threshold value with the highest detection accuracy. The 512 images are comprised of 64 good images and 64 bad images each from 4 different individuals. Following are the results shown in graphs, and the explanations:

Figure 16 shows the graph of accuracy vs threshold for the center head position. It can be observed that the accuracy starts at the peak with 94.5% at threshold value of 0.1, which remains constant before dropping to 88.9% at threshold value of 0.3. The accuracy continues to drop before it remains constant at 50%, from threshold values of 0.6 to 1.0. Hence, 0.2 is selected as the center head position threshold ($C_{thresholdC}$) since it has the highest detection accuracy of 94.5%. Threshold value of 0.2 is selected instead of 0.1 because the higher the threshold, the lower the false alarm.

Figure 17 shows the graph of accuracy vs threshold for the right head position. It can be observed that the accuracy remains constant at 69.9%, from threshold value of 0.1 to 0.2, before rising up to 73.2% at threshold value of 0.3. Then, the accuracy starts to drop to 72.6% at threshold value of 0.4, and continues to drop before it remains constant at 50%, from threshold value of 0.6 to 1.0. Hence, 0.3 is selected as the right head position threshold ($C_{thresholdR}$) since it has the highest detection accuracy of 73.2%.

Figure 18 shows the graph of accuracy vs threshold for the left head position. It can be observed that the accuracy starts with 67.4% at threshold value of 0.1, before rising steadily to 68.2% at threshold value of 0.2, and 69.3% at threshold value of 0.3. Then, the accuracy begins to decrease to 63.9% at threshold value of 0.4, and continues to decrease before it remains constant with 50%, from threshold value of 0.6 to 1.0. Hence, 0.3 is selected as the left head position threshold ($C_{thresholdL}$) since it has the highest detection accuracy of 69.3%.

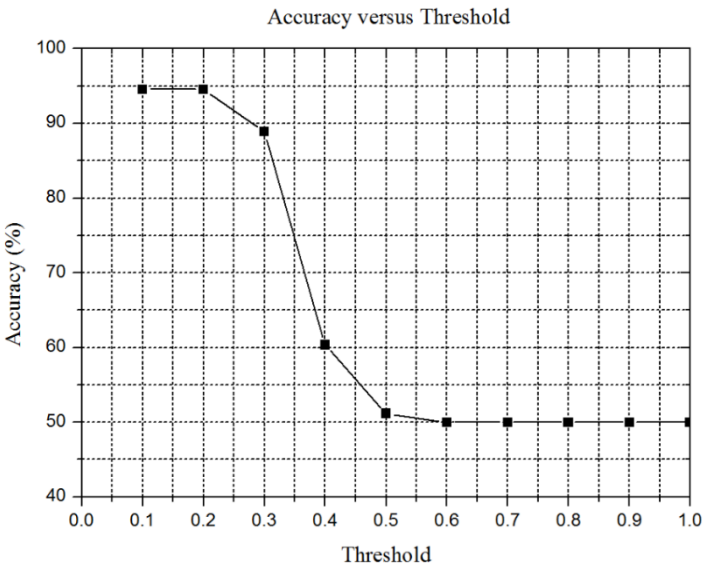


Figure 16. Graph of accuracy vs threshold for center head position.

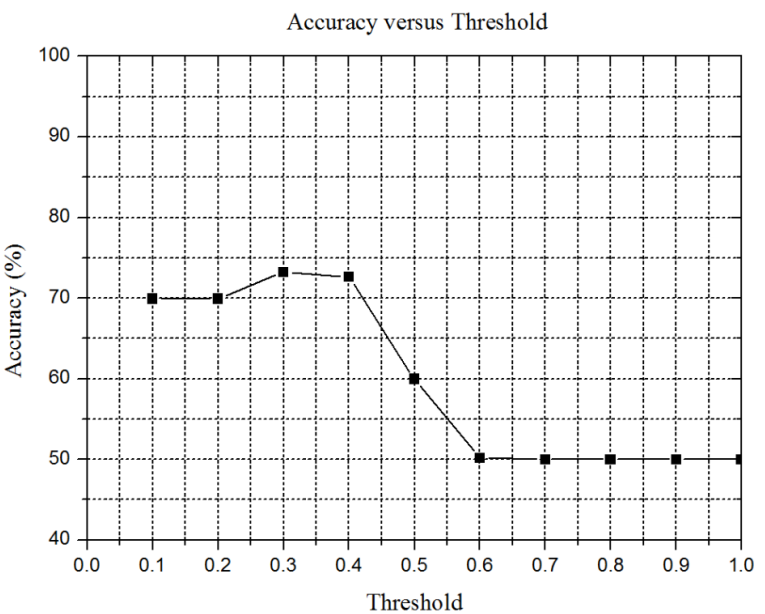


Figure 17. Graph of accuracy vs threshold for right head position.

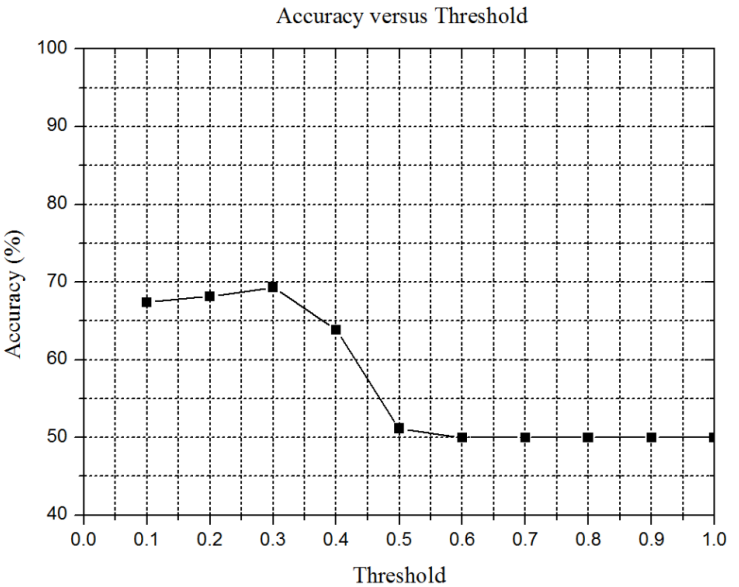


Figure 18. Graph of accuracy vs threshold for left head position.

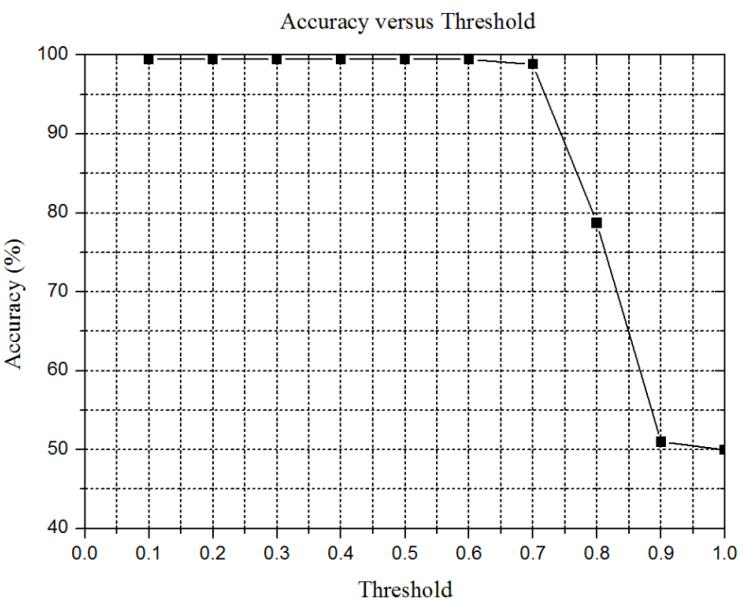


Figure 19. Graph of accuracy vs threshold for leg curve.

Figure 19 shows the graph of accuracy vs threshold for the leg curve. It can be observed that the accuracy starts at the peak with 99.4% at threshold value of 0.1, which remains constant before dropping to 98.8% at threshold value of 0.7. The accuracy continues to drop until the end, with accuracy of 50% at threshold value of 1.0. Hence, 0.6 is selected as the leg curve threshold (C_{leg}) since it has the highest detection accuracy of 98.8%. Threshold value of 0.6 is selected instead of 0.1 to 0.5 because the higher the threshold, the lower the false alarm.

5.1.3. Posture's Conditions

The posture's conditions set in Section 4.3 are derived by observing a set of human postures. Only the outline of the postures are being observed because the proposed method uses the object blob's boundary as stated in Step 5 Section 4. Following are the brief explanations on how the conditions are being derived:

Table 6. Explanation on posture's condition

Posture	Explanation
Stand	From the outline of shape shown in Figure 20(a), center head position and leg curve can be seen.
	From the outline of shape shown in Figure 20(b), center head position can be seen while leg curve is absent.
Bend	From the outline of shape shown in Figure 20(c), leg curve can be seen while center head position is absent.
	From the outline of shape shown in Figure 20(d), either right or left head position can be seen while leg curve is absent.
Sit	From the outline of shape shown in Figure 20(e) and Figure 20(f), center head position can be seen while leg curve is absent.
Lie	From the outline of shape shown in Figure 20(g), center head position, right head position, left head position and leg curve are absent.
	From the outline of shape shown in Figure 20(h), either right or left head position can be seen while leg curve is absent.
	This condition is a mitigation step in reducing false alarms. It is an extension of rule for the outline of shape shown in Figure 20(g). Hence, they have similar conditions where head and leg curve are absent.

* Note that Table 6 is a direct mirror to Table 5. Therefore, the explanation at specific row and column in Table 6 is appointed to the condition in the same row and column in Table 5.

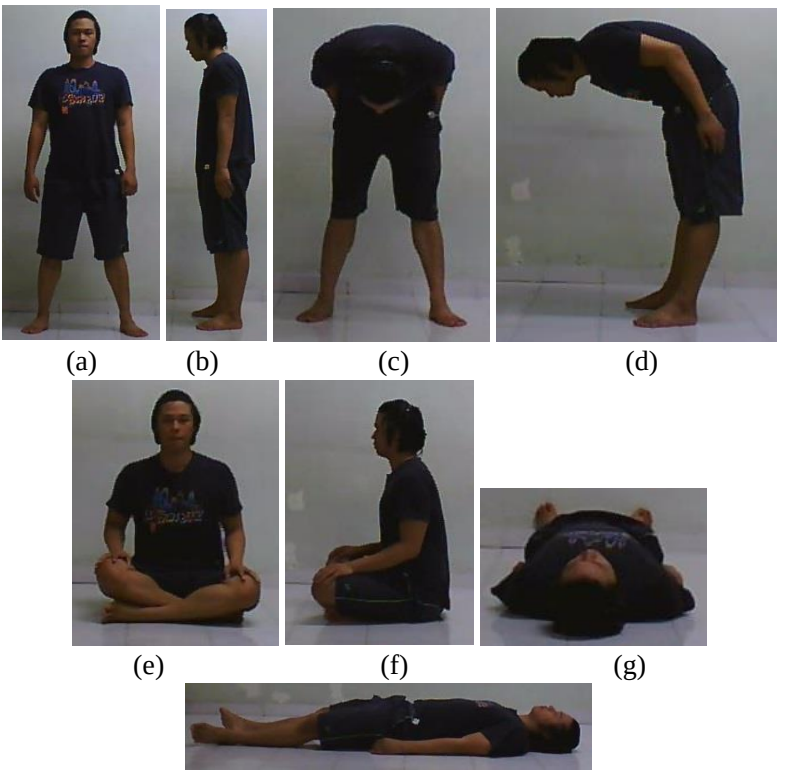
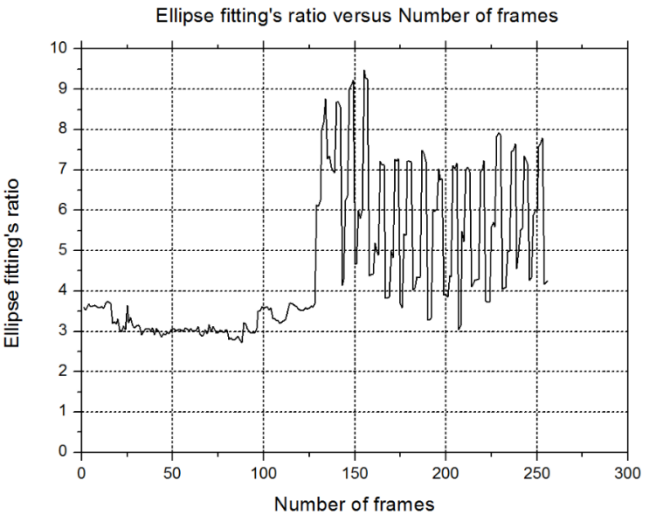


Figure 20. (a) Stand frontal, (b) Stand side, (c) Bend frontal, (d) Bend side, (e) Sit frontal, (f) Sit side, (g) Lie frontal, (h) Lie side.

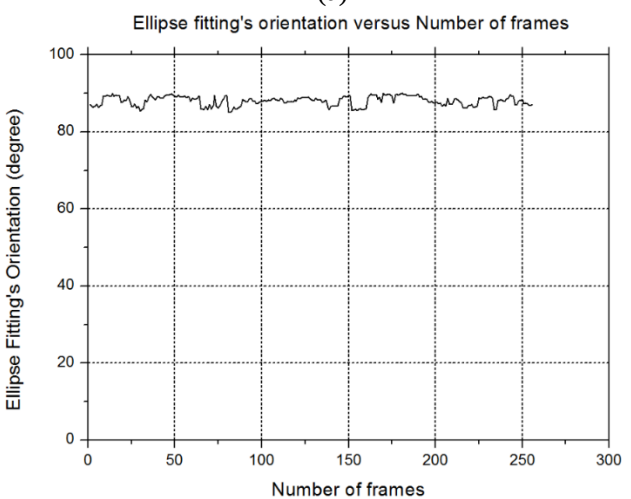
As for the R_{object} and O_{object} that are also included in the conditions in Section 4.3, the explanations will be in Section 5.1.4 below.

5.1.4. Ellipse Fitting's Ratio and Orientation

Test images for ellipse fitting's ratio and orientation experiment are collected from 4 different individuals with different heights, weights and body shapes. These individuals simulated 4 different poses which consist of stand, bend, sit and lie. These poses are simulated in 8 different angles from 0^0 (front facing the camera) to 45^0, 90^0, 135^0, 180^0, 225^0, 270^0 and 315^0. Then, each poses are separated into frontal (front or back facing the camera) and side (left or right side facing the camera) view to ease analysis. A total of 1024 images are being used for this analysis with 256 images for each poses. Following are the results of the experiment:



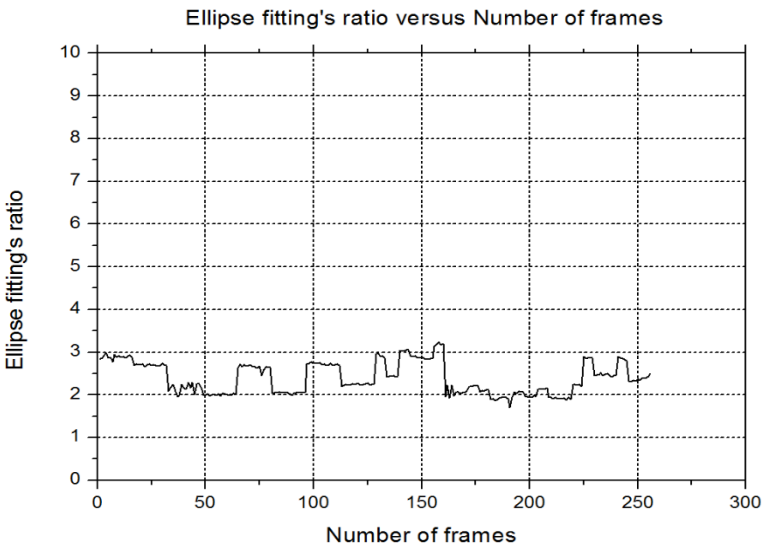
(a)



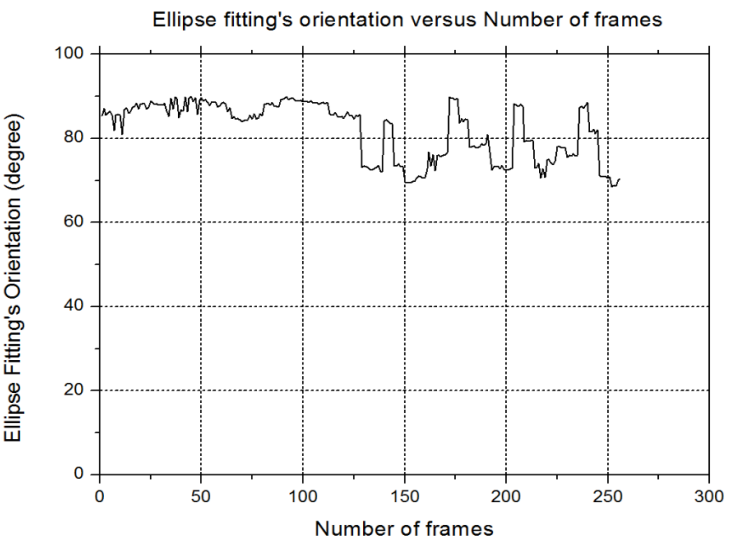
(b)

Figure 21. (a) Ratio graph for stand posture, (b) Orientation graph for stand posture.

The fluctuation shown in Figure 21(a) is due to the shifting of posture from frontal view to side view, causing an increase in the ellipse fitting's ratio. Based on Figure 21(a), the minimum stand ratio ($R_{standMin}$), is set as 2.5 and the maximum stand ratio ($R_{standMax}$), is set as 10. Based on Figure 21(b), the minimum stand orientation ($O_{standMin}$), is set as 80 and the maximum stand orientation ($O_{standMax}$), is set as 90.



(a)

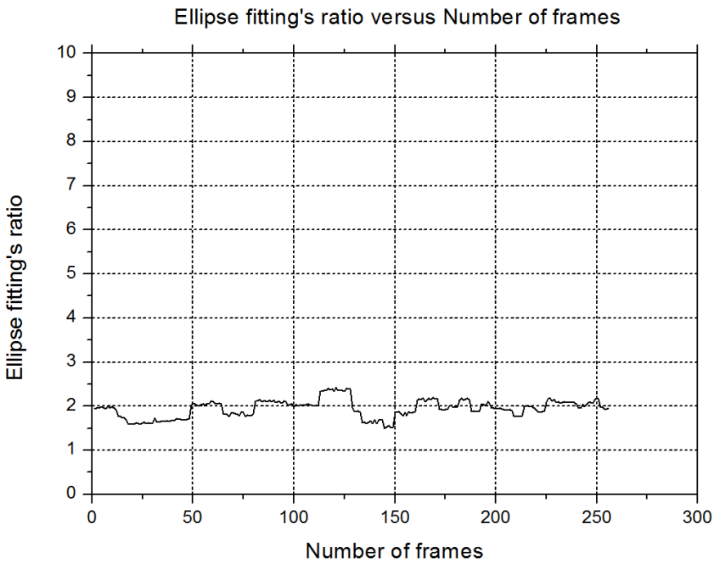


(b)

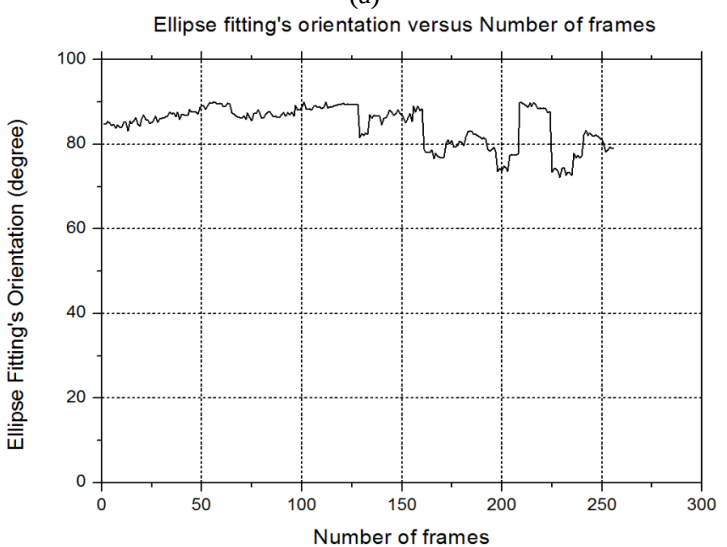
Figure 22. (a) Ratio graph for bend posture, (b) Orientation graph for bend posture.

Based on Figure 22(a), the minimum bend ratio ($R_{bendMin}$), is set as 1.5 and the maximum bend ratio ($R_{bendMax}$), is set as 3.5. Based on Figure 22(b), the

minimum bend orientation ($O_{bendMin}$), is set as 60 and the maximum bend orientation ($O_{bendMax}$), is set as 90.

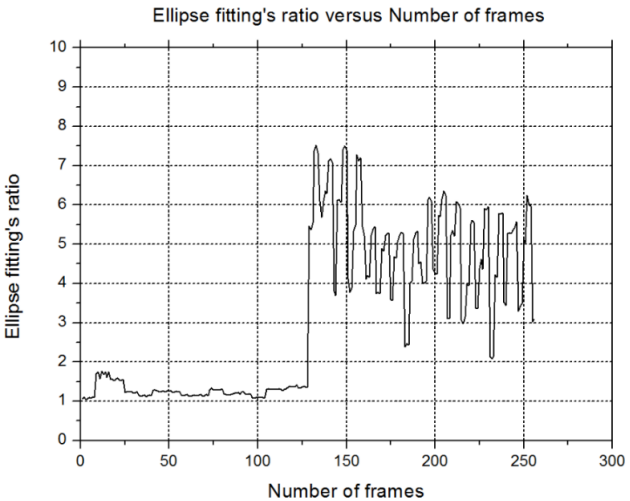


(a)

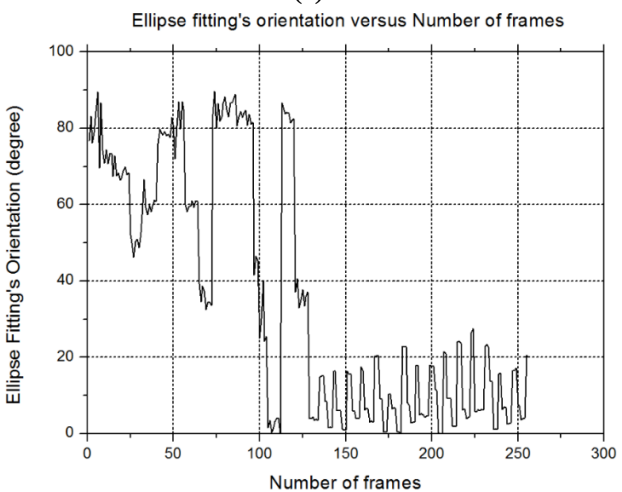


(b)

Figure 23. (a) Ratio graph for sit posture, (b) Orientation graph for sit posture.



(a)



(b)

Figure 24. (a) Ratio graph for lie posture, (b) Orientation graph for lie posture.

Based on Figure 23(a), the minimum sit ratio (R_{sitMin}), is set as 1 and the maximum sit ratio (R_{sitMax}), is set as 2.5. Based on Figure 23(b), the minimum sit orientation (O_{sitMin}), is set as 70 and the maximum sit orientation (O_{sitMax}), is set as 90.

The fluctuation shown in Figure 24(a) and Figure 24(b) are due to the shifting of posture from frontal view to side view, causing an increase in both

the ellipse fitting's ratio and orientation. Based on Figure 24(a), the minimum lie ratio 1 ($R_{lieMin1}$) is set as 0 and minimum lie ratio 2 ($R_{lieMin2}$) is set as 1.9. Meanwhile, the maximum lie ratio 1 ($R_{lieMax1}$) is set as 1.9 and maximum lie ratio 2 ($R_{lieMax2}$) is set as 8. The first set of ratios is for the lie posture in frontal position, while the second set of ratios is for the lie posture in side position. Based on Figure 24(b), the minimum lie orientation 1 ($O_{lieMin1}$) is set as 0, minimum lie orientation 2 ($O_{lieMin2}$) is set as 0 and minimum lie orientation 3 ($O_{lieMin3}$) is set as 5. Meanwhile, the maximum lie orientation 1 ($O_{lieMax1}$) is set as 90, maximum lie orientation 2 ($O_{lieMax2}$) is set as 5 and maximum lie orientation 3 ($O_{lieMax3}$) is set as 60. Similarly, the first set of orientations is for the lie posture in frontal position while the second set of orientations is for lie posture in side position. However, the third set of orientations is a special set built as a mitigation plan for better accuracy as stated in Table 6.

5.2. Extreme Point Curvature Analysis Algorithm

A total of 5136 images from 4 different individuals are used to test the extreme point curvature analysis algorithm. The evaluation is carried by using "Operator perceived Activity" (OPA) [39], where the operator compares the output result by the algorithm, with the actual condition in the image. Our approach is also compared (in terms of accuracy) with a couple of other works that are done in recent years, such as the size filter method [40] and head detection method [41]. Following are the results:

Table 7. Trespasser detection experimental results

Conditions	Extreme point curvature analysis algorithm Testing
Detected Correctly	5001
Not Detected Correctly	135
Accuracy (%)	97.34

Table 8. Trespasser detection comparison results

Metrics	Our approach	Size filter [40]	Head detection [41]
Accuracy (%)	97.34	64.98%	96.30%

The proposed method is able to achieve a detection rate of 97.34% on trespasser detection, and it also outperforms the other works, as shown in Tables 7 and 8.

5.3. Integrated Body Contours Algorithm

A total of 128 activities (64 fainting and 64 non fainting activities) performed by 4 different individuals are used to test the integrated body contours algorithm. The evaluation is also carried out by using OPA as stated in Section 5.2. Our approach is also compared (in terms of accuracy) with a couple of other works that are done in recent years, such as the modified aspect ratio method [42] and head detection method [43]. Following are the results:

Table 9. Faint detection experimental results

Conditions	Integrated body contours algorithm
Detected Correctly	118
Not Detected Correctly	10
Accuracy (%)	92.2

Table 10. Faint detection comparison results

Metrics	Our approach	Modified aspect ratio [42]	Head detection [43]
Accuracy (%)	92.20	90.60%	78.13%

The proposed method is able to achieve a detection rate of 92.2% on faint detection, and it also outperforms the other works, as shown in Tables 9 and 10.

CONCLUSION AND FUTURE RESEARCH DIRECTION

This work or chapter showed the possibility of having an efficient trespasser and faint detection with the use of minimal hardware. The proposed hardware uses only a simple wireless camera that is attached to a custom hyperbolic mirror. The attachment is done by using a custom bracket. Then, with the usage of a suitable unwarping method which is the log-polar mapping method, the mirror image can be easily translated into an omnidirectional panoramic view image. Finally, image processing can be easily done by using the proposed extreme point curvature analysis algorithm for trespasser detection and integrated body contours algorithm for faint detection. Both algorithms showed good results in achieving their intended purposes.

Currently, the proposed omnidirectional surveillance system is applied in indoor buildings especially in common households. In future, with an upgraded algorithm that can solve illumination changes, occlusion problem, and different height situation, the proposed system can be applied in application such as mobile robot surveillance. This mobile robot is able to move and monitor damages on structures situated in places which are inaccessible to a human. These topics will be addressed in the future works.

REFERENCES

[1] Gerald N. H. and Kathleen T. H. (2016) "The People's Law Dictionary", [online], Retrieved 2016 November 5 from http://legal-dictionary.thefreedictionary.com/Trespassers.

[2] Shannan C. (2010). "Victimization during Household Burglary", [online], Retrieved 2016 November 5 from http://www.bjs.gov/content/pub/pdf/vdhb.pdf.

[3] McGoey, C.E. (2012). *"Home Security: Burglary Prevention Advice"*. Aegis Books Inc. 2012.

[4] Miller, J. C., Smith M. L. and McCauley M. E. (1998). *"Crew Fatigue and Performance on U.S. Coast Guard Cutters"*. U.S. Coast Guard Research & Development Center, 1998.

[5] American Heritage, Dictionary of the English Language, Fifth Edition. (2011). "Health care". Houghton Mifflin Harcourt Publishing Company, [online], Retrieved 2016 November 5 from http://www.thefreedictionary.com/health+care.

[6] Random House Kernerman Webster's College Dictionary (2010). "Health care". Random House, Inc. [online], Retrieved 2016 November 5 from http://www.thefreedictionary.com/health+care.

[7] Adams, P. F., Martinez, M. E., Vickerie, J. L. and Kirzinger, W. K. (2011). *"Summary health statistics for the U.S. population"*, National Health Interview Survey, 2010, Vital and Health Statistics Series.

[8] Centers for Disease Control and Prevention (2016). "Falls Among Older Adult: An overview", [online], Retrieved 2016 November 5 from http://www.cdc.gov/HomeandRecreationalSafety/Fals/adultfalls.html.

[9] Stevenson, S. (2014). "10 Products You've Never Heard Of". [online], Retrieved 2016 November 5 from http://www.aplaceformom.com/blog/2014-6-1-cutting-edge-products-for-seniors/.

[10] Baker, A. (2016). "The Top 5 Safety Wearable Products for Seniors", [online], Retrieved 2016 November 5 from http://www.safewise.com/blog/top-safety-wearable-products-for-seniors/.

[11] Miller. J. T. (2016). "How to Keep Tabs On an Elderly Parent with Video Monitoring", [online], Retrieved 2016 November 5 from http://www.huffingtonpost.com/jim-t-miller/how-to-keep-tabs-on-an-el_b_8954044.html.

[12] Fischler, M. A. and Bolles, R. C. (1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Comm. of the ACM*, Vol. 24: p.p. 381-395.

[13] Corke, P., Strelow, D., and Singh, S. (2004). "Omnidirectional visual odometry for a planetary rover". International Conference on Intelligent Robots and Systems(IROS 2004), Vol. 4, p.p. 4007-4012.

[14] Durrant, W. H., and Bailey, T. (2006). "Simultaneous Localization and Mapping (SLAM): Part I The Essential Algorithms". *Robotics and Automation Magazine* Vol. 13: p.p. 99-110.

[15] Kawanishi, T., Yamazawa, K., Iwasa, H., Takemura, H., and Yokoya, N. (1998). "Generation of High-resolution Stereo Panoramic Images by Omnidirectional Imaging Sensor Using Hexagonal Pyramidal Mirrors", Proc. 14th Int. Conf. in Pattern Recognition, Vol. 1, p.p. 485-489.

[16] Ishiguro, H., Yamamoto, M., and Tsuji, S. (1992). "Omni-Directional Stereo", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 14, No. 2, p.p. 257-262.

[17] Huang, H-C., and Hung, Y. P. (1998). "Panoramic Stereo Imaging System with Automatic Disparity Warping and Seaming", Graphical Models and Image Processing, Vol. 60, No. 3, p.p. 196-208.

[18] Peleg, S., and Ben-Ezra, M. (1999). "Stereo Panorama with a Single Camera", Proc. IEEE Conf. Computer Vision and Pattern Recognition, p.p. 395-401.

[19] Shum, H., and Szeliski, R. (1999). "Stereo Reconstruction from Multi-perspective Panoramas", Proc. Seventh Int. Conf. Computer Vision, p.p. 14-21.

[20] Chen, S. E. (1995). "Quick Time VR: An Image-Based Approach to virtual Environment Navigation", Proc. of the 22nd Annual ACM Conf. on Computer Graphics, p.p. 29-38.

[21] Kumar, J., and Bauer, M. (2000). "Fisheye lens design and their relative performance", *Proc. SPIE*, Vol. 4093, p.p. 360-369.

[22] Padjla, T., and Roth, H. (2000). "Panoramic Imaging with SVAVISCA Camera- Simulations and Reality", Research Reports of CMP, Czech Technical University in Prague, No. 16.

[23] Oh, S. J., and Hall, E. L. (1987). "Guidance of a Mobile Robot Using an Omnidirectional Vision Navigation System", Proc. of the Society of Photo-Optical Instrumentation Engineers, SPIE, 852, p.p. 288-300.

[24] Kuban, D. P., Martin, H. L., Zimmermann, S. D., and Busico, N. (1994). "Omniview Motionless Camera Surveillance System", United States Patent No. 5, 359, 363.

[25] Nalwa, V. (1996). *"A True Omnidirecdtional Viewer"*, Technical Report, Bell Laboratories, Homdel, NJ07733, USA.

[26] Akihiko T., Atsushi I. (2004). "Panoramic Image Transform of Omnidirectional Images Using Discrete Geometry Techniques", in Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'04).

[27] Jeng, S.W. and Tsai, W.H. (2007). "Using pano-mapping tables for unwarping of omni-images into panoramic and perspective-view images", in *IET Image Process.*, 1, (2), pp. 149–155.

[28] Jurie, F. (1999). "A new log-polar mapping for space variant imaging: Application to face detection and tracking", *Pattern Recognition, Elsevier Science*, 32:55, p.p. 865-875.

[29] Pua, W. S., Wong, W. K., Loo, C. K. and Lim, W. S. (2013). "A Study of Different Unwarping Methods for Omnidirectional Imaging", *Computer Technology and Application 3*, pp. 226-239.

[30] Huang, D. S., Wunsch, D.C., Levine, D.S., Jo, K-H. (2008). Advanced intelligent computing theories and applications: with aspects of theoretical and methodological issues in 4th International Conference on Intelligent Computing, ICIC 2008, Shanghai, China, September.

[31] Hampapur, A., Brown, L., Connell, J., Ekin, A., Haas, N., Lu, M. et al. (2005). "Smart Video Surveillance", *IEEE Signal Processing Mag.*, p.p. 39-51.

[32] George, W., Siavash, Z. (2000). "Robust Image Registration Using Log-Polar Transform", in Proc. of IEEE Intl. Conf. on Image Processing.

[33] Traver, V. J., Alexandre B. (2010). "A review of log-polar imaging for visual perception in robotics" in *Robotics and Autonomous Systems* 58, p.p. 378-398.

[34] Jos´e S. V., Alexandre B. (2003). "Vision-based Navigation, Environmental Representations and Imaging Geometries", in VisLab-

TR 01/2003, Robotics Research, 10th International Symposium, R. Jarvis and A. Zelinsky (Eds), Springer.

[35] Yu, M., Rhuma, M., Naqvi, S. M., Wang, L. and Chambers, J. (2012). "A Posture Recognition-Based Fall Detection System for Monitoring and Elderly Person in a Smart Home Environment", *IEEE Transactions on Information Technology in Biomedicine*, Vol. 16, No. 6, November 2012.

[36] Pratt, W. (2001). Digital Image Processing, 3rd ed. Hoboken, NJ: Wiley.

[37] Rougier, C., Meunier, J., St-Arnaud, A. and Rousseau, J. (2007). "Fall detection from human shape and motion history using video surveillance," in Proc. 21st International Conference on Advanced Information Networking and Applications Workshops, (AINAW"07), Niagara Falls, ON, Canada, pp. 875–880.

[38] Jain, A., (1989). *Fundamentals of digital image processing.* Prentice Hall, Englewood Cliffs, New Jersey.

[39] Owens, J., Hunter, A., and Fletcher, E. (2002). "A fast model-free morphology-based object tracking algorithm", British Machine Vision Conference, vol.2, p.p. 767-776.

[40] Hafiz, F., Shafie, A. A., Khalifa, O. and Ali M. H. (2010). "Foreground segmentation-based human detection with shadow removal," International Conference on Computer and Communication Engineering (ICCCE 2010), pp.1-6, May, 11-13, 2010.

[41] Wong, W. K., Chew, Z. Y., Loo, C. K. and Lim, W. S. (2010). "An Effective Trespasser Detection System Using Thermal Camera," 2nd International Conference on Computer Research and Development, p.p.702 –706.

[42] Vaidehi, V., Ganapathy, K., Mohan, K., Aldrin, A. and Nirmal, K. (2011). "Video based automatic fall detection in indoor environment," IEEE International Conference on Recent Trends in Information Technology, ICRTIT, pp. 1016-1020.

[43] Wong, W. K., Poh, Y. C., Loo, C. K. and Lim, W. S. (2010). "Wireless Webcam Based Omnidirectional Health Care Surveillance System," 2nd International Conference on Computer Research and Development, pp. 712-716, May, 7-10, 2010.

In: Surveillance Systems
Editor: Roger Simmons
ISBN: 978-1-53610-703-6
© 2017 Nova Science Publishers, Inc.

Chapter 2

TRACKING MOVING OBJECTS IN VIDEO SURVEILLANCE SYSTEMS WITH KALMAN AND PARTICLE FILTERS – A PRACTICAL APPROACH

Grzegorz Szwoch*
Gdansk University of Technology,
Department of Multimedia Systems, Gdansk, Poland

ABSTRACT

The development and tuning of an automated object tracking system for implementation in a video surveillance system is a complex task, requiring understanding how these algorithms work, and also the experience with choosing proper algorithm parameters in order to obtain accurate results. This Chapter presents a practical approach to the problem of a single camera object tracking, based on the object detection and tracking with Kalman filters and particle filters. The aim is to provide practical guidelines for specialists who design, tune and evaluate video surveillance systems based on the automated tracking of moving objects. The main components of the tracking system, the most important parameters and their influence on the obtained results, are discussed. The described tracking algorithm starts with the detection phase which

* greg@sound.eti.pg.gda.pl.

identifies areas in each video image that represent moving objects, employing background subtraction and morphological processing. Next, movement of each detected object is tracked on a frame-by-frame basis, providing a 'track' of each object. First, the Kalman filter approach is presented. Implementation of a dynamic model for the filter prediction, methods of obtaining the measurement for updating the filter, and the influence of the noise variance parameters on the results, are discussed. Tracking with Kalman filters fails in many practical situations when the tracked objects come into a conflict due to the object occlusion and fragmentation in the camera images. Another method presented here is based on particle filters which are updated using color histograms of the tracked objects. This method is more robust to tracking conflicts than the Kalman filter, but it is less accurate in describing the object size, and it is also much more demanding in terms of computation. Therefore, a combined approach for resolving the tracking conflicts, is proposed. This algorithm uses Kalman filters for the basic, non-conflict tracking, and switches to the particle filter for resolving cases of occlusion and fragmentation. A methodology of evaluation of tracking algorithms is also presented, and an example of testing the three presented tracking algorithms on a sample test video is shown.

INTRODUCTION

Smart cameras are the current trend in video surveillance systems [Nab01]. The number of cameras installed in modern video monitoring solutions still increases, and a human operator is not able to notice every important event that occurs in the surveyed areas. Therefore, automated video content analysis (VCA) algorithms are implemented as a 'helper' for the operators [Lin11]. Traditionally, such algorithms require powerful workstations to run complex video analysis in real time. However, this situation started to change in the recent years, with a development of powerful and energy-efficient computing platforms, such as GPUs, FPGAs, DSPs, etc. Such devices may be implemented within embedded camera systems, forming smart devices that combine video sensors with processors running VCA algorithms, within a single device.

Various complex VCA algorithms may be implemented in surveillance systems, performing detection of specific events, such as abandoned luggage, robbery, traffic law violations, etc. However, most of these algorithms are built upon two basic operations: object detection and object tracking [Czy11]. The former extracts moving objects from video, and the latter tracks

movement of individual objects as long as they are present in the camera view. These 'tracks' of moving objects may be then used for a high-level analysis of the object behavior. Therefore, the developer of VCA solutions has to ensure that these two basic operations are performed with a sufficient accuracy. If the object is lost or the tracker is assigned to a wrong object, further event analysis becomes impossible. Therefore, this Chapter focuses on performing the object tracking stage in a way that accurate tracks, useful for further analysis, are obtained.

Object detection identifies individual objects in static images (single video frames), and produces data needed for object tracking. There are two main approaches to object detection. The first one aims to detect a specific class of objects, employing an algorithm trained with sample images of the desired class. Such algorithms include the Viola-Jones detector [Vio01], commonly used for face detection, and histograms of oriented gradients (HoG), usually applied to human detection [Dal05]. The other group of methods is based on *background subtraction* and it usually employs a statistical background model in order to separate moving objects from the background. Such a model has to be constantly updated in order to adapt to varying conditions. An algorithm of this type which is commonly applied for the VCA is based on Gaussian mixture models (GMM), as proposed by Stauffer and Grimson [Sta99], and later extended by Zivkovic [Ziv06]. There are also other background subtraction algorithms, e.g., the Codebook algorithm which utilizes a layered background model [Kim05]. The main drawback of the background subtraction approach is that it works only in fixed view cameras.

For successful tracking, the position of each object has to be established in each analyzed video frame. A collection of the extracted object data constitutes a *track* of the object. The main challenge in object tracking is related to *conflict situations* in which different moving objects overlap in the camera view (*occlusion*) or they are divided into separate objects. If the frequency of such conflicts is relatively low and individual objects may be detected most of the time, algorithms that track the results of object detection (image regions, often called *blobs*) are usually employed. Such algorithms work on the 'prediction-update' principle. The most notable algorithm of this type is based on *Kalman filters* [Wel04]. It is often used in VCA applications because it is computationally efficient and accurate as long as the conflicts are relatively rare and short-term [Czy11]. *Particle filters* [Aru02, Ris04] are an alternative approach which is significantly more robust to tracking conflicts, but it is also much more demanding in terms of processing resources, and less accurate in tracking the object size. Therefore, particle filters are less common

in VCA systems than Kalman filters, but their advantages were utilized in a number of published works. For example, Isaard and Blake [Isa98] proposed the Condensation algorithm based on particle filters, for tracking object contours in a cluttered environment. Nuumiaro et al. [Num03] used particle filters based on color histograms for tracking moving objects in video. Czyz et al. [Czy06] extended the particle tracker with automatic object detection and tracking multiple objects.

The tracking methods described above are not suitable for crowded scenes, with a large number of objects and a high occurrence of object occlusions. Such scenarios require a different approach to tracking. For example, the CamShift algorithm [Bra98] searches for a specified object in the image with a sliding window, using the histogram back-projection method. Optical flow methods (e.g., Lucas-Kanade [Luc81] and Horn-Schunck [Hor81]) are often used for tracking objects in busy scenes. Dense optical flow methods detect the movement by analyzing all image pixels, while sparse optical flow algorithms analyze the movement of key points (corners), detected e.g., with the Shi-Tomasi algorithm [Shi94]. The optical flow approaches are computationally demanding and because of this, their implementation in VCA systems is still a challenge.

This Chapter focuses on the first type of object tracking algorithms, namely on Kalman and particle filters. A theory of these algorithms may be found in many publications, there are also reports on implementation of these approaches to object tracking in video. However, developers of VCA systems still face two important problems. The first one is related to obtaining accurate measurements of positions and sizes of the tracked objects, required for updating their tracker. It is easy to do if the object is clearly identified in the camera image, but in case of tracking conflicts, obtaining a valid measurement is not trivial The second problem is related to the parameters tuning in the object detection and tracking algorithms, in order to obtain accurate object tracks. Despite the abundance of publications on object tracking in video with these methods, it is not easy to find a clear solution to both problems. This Chapter has therefore two main aims. First, it attempts to fill the abovementioned gap, by describing the influence of the algorithm parameters on the obtained results, and also presenting the problem of obtaining accurate measurements for updating the tracking filter in presence of conflicts. Second, a novel approach that combines Kalman filters with particle filters, is proposed. This dual-type tracker uses the simpler Kalman filter when there are no conflicts, and the more demanding particle tracker only for resolving these conflicts.

The rest of the Chapter is organized as follows. The next Section describes the background subtraction procedure based on the GMM, and discusses the influence of its parameters on the accuracy of object detection. Next, the Kalman filter algorithm which tracks objects using data obtained from the object detection stage, is presented. A relationship between the algorithm parameters and the tracking accuracy, as well as problems related to tracking conflicts, are discussed. Then, a tracker utilizing a particle filter is presented. In the subsequent Section, the combined tracker is proposed in order to obtain more accurate measurements for updating the Kalman filter in case of conflicts, by utilizing the advantages of both filter types. The final Sections present a method of evaluating the performance of tracking algorithms, discuss the results of tests of the presented algorithms, and finish the Chapter with the Conclusion.

OBJECT DETECTION

Performing object tracking in a video stream requires that data on the position and size of each moving object, in every analyzed video image, is obtained first. Therefore, the task of the object detection procedure is to analyze the individual video frames, and to extract image regions representing important moving objects that should be tracked. In this Section, the classic approach based on background subtraction with the GMM algorithm, is presented. The obtained results are post-processed, and then connected components (blobs) representing moving objects are extracted, forming data suitable for tracking.

Background Subtraction

The first stage of object detection is the *background subtraction*. Removing background from the image means that pixels belonging to moving objects (humans, vehicles, etc.) have to be detected and separated from the background. A naïve approach to background subtraction would be based on differencing or averaging the video frames. However, such methods do not work reliably in practical situations, because values of background pixels vary constantly due to the camera noise, changes in light, etc. Additionally, there is a problem of ‘ghosting,’ i.e., pixels left behind moving objects would also be detected as the foreground. In order to avoid such errors, a *statistical*

background model is usually employed. Most of the algorithms of this kind assume that the values of each background pixel, observed for a long time, form a Gaussian distribution, with a mean value μ and a standard deviation σ. Such a model may be learned from a number of initial video frames. Various algorithms for background modeling were proposed, but the approach based on Gaussian mixture models (GMM) [Sta99, Ziv06] is the one most widely used in video analysis.

Assuming a single-channel image, the pixel having value x is described by a background model with values (μ, σ) if:

$$|x-\mu| \le b\sigma \tag{1}$$

where b is a factor determining the maximum distance between the pixel value and the model mean, usually b is 2.0 to 3.0. A separate background model has to be constructed for each image pixel. For each analyzed video frame, every pixel is compared with its model and assigned to either the background and the foreground. In practice, three-channel (RGB) images are analyzed, so the background models have to store μ and σ for each color channel, separately. The pixel belongs to the background if Eq. 1 is fulfilled for all three channels.

A pre-learned background model would become invalid if, for example, light in the scene changes (e.g., sun comes out from the clouds). Therefore, the background models need to be constantly updated. In the GMM algorithm, each model is initialized by setting its mean to the value of the first observed pixel, and the initial variance to a predefined, high value. For each analyzed image, means and variances of the models are updated if the pixel was assigned to the background, using the following equations [Sta99]:

$$\mu_t = \mu_{t-1} + \rho(x_t - \mu_{t-1}) \tag{2}$$

$$\sigma_t^2 = \sigma_{t-1}^2 + \rho\left[(x_t - \mu_t)^2 - \sigma_{t-1}^2\right] \tag{3}$$

where x is the pixel value, t is time index and ρ is the *background update rate*. Higher values of ρ result in a model that adapts quickly to changes, but frequent changes in the background may prevent the model from becoming stable. Lower values cause the model to react slowly to changes, resulting in a more stable model which needs more time to ‘re-learn.’ During the adaptation phase (before these changes are fully incorporated into the model), a large

number of false positive detections may be expected. Typical values of ρ range from 10^{-5} to 10^{-2}, but there is no universal value, this parameter has to be selected according to the frequency and intensity of the expected background variations.

The background model described above is unimodal. When a change in the background scene persists for a prolonged time and the model is updated, the original background is lost. For example, a vehicle stopped on the traffic light may be incorporated into the background, and when it leaves, the model needs to be re-learned. In order to avoid such problems, the GMM algorithm uses a mixture of weighted Gaussians [Sta99]. Each pixel model consists of k Gaussians, usually k is 3 to 5. The weights of all Gaussians sum up to one. Initially, a single Gaussian with an unit weight is created. For each video frame, the algorithm searches for the first matching Gaussian. If it is found, its parameters are updated according to Eqs. 2 and 3. If none of the Gaussians were matched, a new one is created (replacing the one with the lowest weight, if a maximum number of Gaussians is reached). Next, the weights of all Gaussians are updated:

$$\pi_t = \pi_{t-1} + \alpha(o_t - \pi_{t-1}) - \alpha c_T \tag{4}$$

where π is the weight, o is one if the Gaussian was matched and zero otherwise, α is the *learning ratio* which has similar meaning to ρ (often, identical values are used for both parameters). The parameter c_T was introduced by Zivkovic [Ziv06] in order to reduce the influence of older (not updated recently) Gaussians on the detection. When c_T is 0, the classic GMM algorithm is used. Additionally, the modified algorithm chooses k dynamically. After the update, the weights are normalized to the unit sum, and the Gaussians are ordered by a decreasing ratio of the weight to the variance.

The decision whether the pixel belongs to the foreground or to the background is made as follows. The number of distributions that describe the background is given by [Sta99]:

$$B = \arg\min_b \left(\sum_{k=1}^{b} \pi_b > T \right) \tag{5}$$

where T is the threshold, usually set to 0.5-0.7. This approach limits the actual background model to a number of Gaussians with sufficiently high weights. If

a matching Gaussian was found and it is one of B Gaussians with the highest weights, the pixel is assigned to the background, otherwise it is classified as the foreground.

Background subtraction with the GMM may be interpreted as follows. Initially, the Gaussian modeling a given pixel has the mean equal to the value of this pixel, and a high variance. If the Gaussian is matched in the consecutive video frames, its variance decreases gradually, with the coefficient ρ determining the speed of this adaptation. It is useful to limit the minimum variance value in order to prevent 'overfitting.' The variance becomes low for a stable background and it is higher in case of e.g., frequent light changes or the camera noise.

The background model is usually initialized with the first received video frame, so the initial model includes moving objects present in this image. Therefore, the model initialization should be performed when no moving objects are present, or the model has to be learned for a defined time before the actual detection is started. Alternatively, the learning rates ρ and α may be set to much higher values during the initialization phase, for example:

$$\alpha' = \max\left(\alpha \frac{T_{init}}{t},\ 0.5\right) \text{if } t < T_{init} \tag{6}$$

where t is the frame index and T_{init} is the number of frames used for initialization. With this approach, the learning parameters start with high values and they gradually decrease towards the target values. As a result, the detection accuracy is improved during the initialization.

Background subtraction requires a significant amount of processing time and it has high memory requirements. The following considerations should be taken into account when this algorithm is implemented.

Video resolution. Each image pixel is analyzed independently. Therefore, the video resolution has the largest impact on the processing time and on the memory usage. A powerful processing hardware is required for a real-time analysis of high resolution video streams. In case that limited resources are available, the video resolution may be decreased by downscaling each image, e.g., by a factor of 2. This operation reduces the processing requirements significantly, but at the same time, the analysis resolution is decreased. It should also be noted that the algorithm is well suited for parallel implementation, e.g., on GPU platforms [Szw15, Szw16].

The number of Gaussians. The minimum number of Gaussians per model in practical applications is 3. Increasing this number may be helpful if it is necessary to remember a larger number of background modes, but it increases both the processing time and the memory usage, so it should be avoided if possible.

Post-Processing and Object Extraction

As a result of the background subtraction, a *foreground mask* – a binary image denoting foreground and background pixels, is obtained. Usually, this mask has to be post-processed (cleaned) before regions of foreground pixels, representing moving objects, are extracted. These stages are summarized below.

Shadow removal. Shadows of moving objects are normally detected as foreground pixels and they disrupt the object detection, hence they should be removed from the mask. Various approaches to this problem have been proposed in the literature. Most of them incorporate the shadow removal procedure into the background subtraction stage, basing on the observation that shadows have similar color to the background, but their brightness is lower. An example of such an algorithm may be found in the publication by Horprasert et al. [Hor99].

Morphological processing. The result of background subtraction is rarely clean. Usually, it is contaminated by a ‘pixel noise’ and gaps, resulting from imperfections of the background subtraction algorithm. Such errors are usually reduced by means of morphological processing [Xu05]. First, the morphological opening reduces noise, then the morphological closing fills small holes. Therefore, a following sequence of morphological operators is applied: erosion, dilation, dilation and erosion [Czy11]. Typically, a 3×3 square kernel is used in these operations. Larger kernels may be useful in removing larger artifacts (especially in high resolution videos), but they may distort the contours of objects. Figure 1 shows the background subtraction mask without and with morphological processing. It may be observed that the post-processing removed some defects (small contours and holes inside large contours).

Figure 1. Example of post-processing of the background subtraction results. Top: the analyzed image with moving objects. Middle: the result of background subtraction (black color denotes the foreground pixels, noise and gaps are visible). Bottom: the result of morphological processing (the noise is removed and small gaps are filled).

Blob extraction. Connected components of foreground pixels are commonly named *blobs*. Each blob may represent a single moving object, a group of objects (occlusion) or a part of the object (fragmentation). Blobs are usually extracted from binary images using the border following algorithm [Suz85]. Analysis of the mask with these methods is usually destructive, so a copy of the mask should be preserved for further analysis. A blob is described by its *contour* – a list of edge points. A simplified blob descriptor is its *bounding rectangle*, represented by the center point (x_c, y_c), width w and height h:

$$\mathbf{b} = [x_c, y_c, w, h] \tag{7}$$

Alternatively, **b** may be seen as an ellipse with axes *w* and *h*, and the center point as above.

A set of vectors **b**, extracted during the object detection stage, constitutes data on the positions and sizes of moving objects detected in the current video frame. It should be noted that any error made at this stage is propagated to the actual tracking phase. Therefore, it is necessary to reduce the number of background subtraction errors as much as possible, mainly by tuning the algorithm parameters.

Typical Problems in Background Subtraction

The most important problems that may occur during the background subtraction stage, and possible approaches to avoid at least some of them, are summarized below.

Changes in the scene lighting. This problem is important in both the outdoor scenes, when clouds move across the sky, and the indoor scenes, when the natural or artificial light changes. Frequent variations in the scene lighting require a re-adaptation of the model, and during this phase, object detection is practically impossible because of an excessive number of false-positive results. In some situations, this problem may be reduced by increasing the learning rate and using a larger number of Gaussians per pixel. Special algorithms performing *backlight compensation* may also be useful.

Objects incorporated into the background. Objects that remain stationary for a prolonged time may eventually be learned as the background. An example is a vehicle stopping at the red light. When this object moves, it leaves a ‘background hole’ which, without further analysis, would be detected as a moving object. This problem may be reduced by lowering the learning rate, which contradicts the solution to the previous problem. Background holes may be detected by an analysis of the pixel values in the neighborhood of the blob contour. In case of holes, there should be a smooth transition of the pixel values, while for the actual objects, edges should be detected alongside the contour [Szw10].

Camouflaging. This is one of the most common problems in background subtraction. When color of a moving object is very similar to the background (e.g., a person in a white shirt on a light gray wall), some pixels of the moving

object will be assigned to the background, resulting in false negative detections and fragmentation of the blob. This problem is extremely difficult to avoid. It may be, to some degree, reduced by decreasing *b* in Eq. 1 to 2.0 or less, reducing the detector sensitivity, but it may also result in false positive decisions in other areas. Figure 2 shows the influence of *b* on the background subtraction results. The value of *b* set too low results in an excessive noise, while too high *b* causes a blob fragmentation (many false negative results).

Figure 2. Influence of the *b* parameter (Eq. 1) on the background subtraction accuracy: *b* = 1.5 (too many false positives); *b* = 2.5 (optimal); and *b* = 4.5 (some false negatives).

Occlusion and fragmentation. Ideally, a single blob should represent one moving objects. If objects occlude each other in the camera view, they are detected as a single blob. On the other hand, an object may be divided into more than one blob. It may happen when some pixels are incorrectly assigned to the background (e.g., because of camouflaging), or because a stationary object in the scene (e.g., a post, a tree) partially occludes the object. Figure 3 illustrates the occlusion and fragmentation problem. Such errors need to be coped with during the tracking phase, by intelligent splitting and merging of blobs.

Reflections. Shadow removal has already been mentioned. Another problem is caused by reflective surfaces, such as floor, windows, glass walls, etc. Reflections in such areas are detected as moving objects and they cannot be removed from the foreground mask using the same methods as for the shadows.

Night scenes. During the night, background subtraction becomes very difficult (sometimes even impossible) because of an extreme chrominance noise and intensive highlights, e.g., from vehicle headlights. Noise removal is necessary during the preprocessing stage, but it also blurs the contours of moving objects.

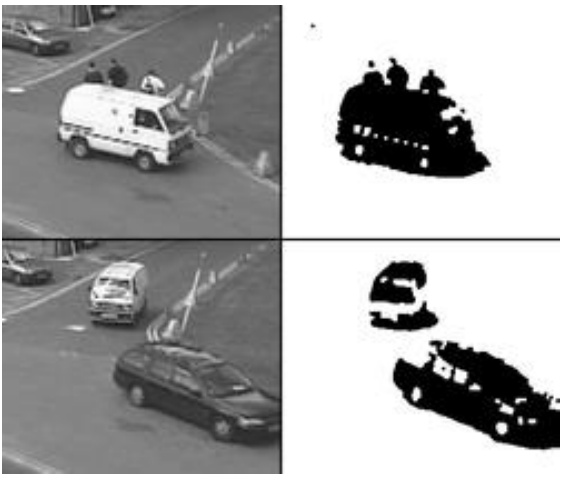


Figure 3. Examples of typical object detection problems. Top: occlusion, a vehicle and three persons are represented with a single, merged blob. Bottom: fragmentation, the white vehicle is fragmented into two blobs due to an inaccurate background subtraction.

Camera settings. Finally, some settings of a video source may be tuned in order to decrease the risk of errors. The video compression ratio should be set as low as possible, because compression artifacts (such as pixel blocking) in highly compressed video streams may decrease the detection accuracy (but at the same time, a sufficient frame rate has to be preserved for the tracking phase). Sensor sensitivity (ISO) should not be set too high in order to keep the camera noise at a reasonable level. Automatic white balance and automatic camera brightness should be turned off, because they change the overall brightness and color of the image and force the background model to readjust. A fixed profile of white balance (indoor or outdoor, depending on the scene) is recommended.

OBJECT TRACKING WITH KALMAN FILTERS

Once the object detection phase is completed, the resulting data, describing the positions and sizes of the detected moving objects, may be used for object tracking. The purpose of this stage of video analysis is to combine the data obtained from each analyzed video frame, in order to track movement of each individual object, as long as it is visible in the camera. A *tracker* is a structure which keeps the current and the past states of a single moving object.

A collection of states gathered from all analyzed video frames constitute a *track* of the object (Figure 4). In a trivial case of a single moving object, its states could be obtained directly from the object detection stage. However, in practical scenarios, multiple moving object are present simultaneously in the camera view, and *tracking conflicts* (e.g., occlusions) may happen. As long as individual objects can be detected in most of the analyzed frames (the conflicts are sufficiently infrequent and short-term), each object may be tracked separately, by a tracker assigned to it. Most of the object tracking algorithms work in two stages. The *prediction* stage computes an estimate of the current tracker state (the prior) which may be compared e.g., with the object detection results. The *update* phase calculates the current state (the posterior) based on a *measurement*. In this Section, an algorithm based on Kalman filters is used, an alternative approach employing particle filters will be presented later in the Chapter.

Figure 4. A track of the moving object – positions of the tracked object, detected in the analyzed video frames, are marked with white dots. The top images present the examples of tracking results (the bounding box of the white van is shown).

State Variables and the Dynamic Model

Kalman filters (KF) are commonly used for tracking data from noisy sensors [Wel04], and they are applicable to object tracking in video [Czy11]. It is assumed that both the dynamic process and the measurements are contaminated with noise having a normal distribution. Tracking moving objects may be performed using a first order or a second order dynamic model

(more complex models are also possible) [Li03]. A vector of the *tracker state* variables for the first order model is given by:

$$\mathbf{s} = \left[x, y, \dot{x}, \dot{y}, w, h, d_w, d_h\right]^T \tag{8}$$

where the variables denote: the position of the center point of the object's bounding box, the object velocity (in x and y directions), the size of the bounding box, and the size change. Alternatively, a single scale factor may be used instead of the two size change variables, or they may be omitted from the vector (assuming that the tracker size is constant). For the second order dynamic model, the acceleration is also included:

$$\mathbf{s} = \left[x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}, w, h, d_w, d_h\right]^T \tag{9}$$

The model state is observed in discrete moments, corresponding to the analyzed video frames. Assuming the common case of a constant video frame rate, it is useful to use frame numbers as the time indices, so that the time difference between two consecutive frames is equal to one. The state propagation from time t-1 to t, using the first order model, is given by:

$$\begin{aligned} x_t &= x_{t-1} + \dot{x}_{t-1} & y_t &= y_{t-1} + \dot{y}_{t-1} \\ \dot{x}_t &= \dot{x}_{t-1} & \dot{y}_t &= \dot{y}_{t-1} \\ w_t &= w_{t-1} + d_{w,t-1} & h_t &= h_{t-1} + d_{h,t-1} \\ d_{w,t} &= d_{w,t-1} & d_{h,t} &= d_{h,t-1} \end{aligned} \tag{10}$$

and for the second order model:

$$\begin{aligned} x_t &= x_{t-1} + \dot{x}_{t-1} + 0.5\ddot{x}_{t-1} & y_t &= y_{t-1} + \dot{y}_{t-1} + 0.5\ddot{y}_{t-1} \\ \dot{x}_t &= \dot{x}_{t-1} + \ddot{x}_{t-1} & \dot{y}_t &= \dot{y}_{t-1} + \ddot{y}_{t-1} \\ \ddot{x}_t &= \ddot{x}_{t-1} & \ddot{y}_t &= \ddot{y}_{t-1} \end{aligned} \tag{11}$$

where the size equations are the same as in the first order model.

It is not trivial to justify the choice of either model. The first order model assumes a constant velocity, which is not true in practical situations, so this model is oversimplified. On the other hand, accurate modeling of the velocity

changes due to both the actual acceleration and deceleration of the object, and the camera perspective effect, with the second order model is problematic, and also a larger number of the state variables affects the filter performance. The second order model may be useful if the tracked objects are expected to slow down and accelerate very often (e.g., vehicles at traffic lights), so that the trackers do not lose their objects. In simpler scenarios (e.g., persons inside a building), the first order model should be sufficient.

Prediction and Update with Kalman Filters

Mathematical background of the KF is given in [Wel04] and a practical tutorial may be found in [Lab15]. In the prediction (time update) phase, an estimate of the tracker state is computed:

$$\mathbf{s}_t^- = \mathbf{A}\mathbf{s}_{t-1} \tag{12}$$

where **A** is the process matrix. In the scenario described here, the matrix **A** for the first and the second order models may be determined from Eqs. 10 and 11:

$$\mathbf{A}_I = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{A}_{II} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

Next, an estimate of the *error covariance matrix* is computed:

$$\mathbf{P}_t^- = \mathbf{A}\mathbf{P}_{t-1}\mathbf{A}^T + \mathbf{Q} \tag{14}$$

The matrix **P** is initialized with the starting (usually high) variance and it is updated internally by the filter. The matrix **Q** describes the *process noise covariance* and it represents the uncertainty in the dynamic model. Usually, in tracking moving objects in cameras, it is a diagonal matrix, with values on the

diagonal representing the variance of each state variable. Often, the same value is used for the variance of all variables, but also separate variances may be set e.g., for estimation of the position and the size.

From the predicted state variable vector, a predicted tracker state may be obtained by selecting the variables related to the position and the size:

$$\mathbf{o}_t^- = \left[x_t^-, y_t^-, w_t^-, h_t^-\right] \tag{15}$$

The main problem in tracking with the KF is obtaining valid measurements for updating the trackers in case of conflicts In the algorithm presented here, the predicted states of all trackers are compared with the object detection results (the detected blobs). For this task, a *relationship matrix* is constructed by comparing predicted trackers **o** with the detected blobs **b** [Czy08]. Here, it is assumed that a tracker and a blob are related if their bounding boxes (rectangles) overlap. Therefore, the resulting matrix is binary – regions are either related or not. Alternatively, ellipse regions may be used, or an actual coverage area may be calculated. First, a simplified case will be discussed: each predicted tracker state is related to at most one blob, and each blob is related to at most one tracker (a 1-1 relation). In this situation, **b** may be used as a measurement for updating the matched tracker. Other cases of the tracker-blob relations cause the tracking conflicts, and they will be discussed later.

Updating the KF with a measurement is given by [Wel04]:

$$\mathbf{K}_t = \mathbf{P}_t^- \mathbf{H}^T \left(\mathbf{H}\mathbf{P}_t^- \mathbf{H}^T + \mathbf{R}\right)^{-1} \tag{16}$$

$$\mathbf{s}_t = \mathbf{s}_t^- + \mathbf{K}_t \left(\mathbf{z}_t - \mathbf{H}\mathbf{s}_t^-\right) \tag{17}$$

$$\mathbf{P}_t = \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}\right)\mathbf{P}_t^- \tag{18}$$

The matrix **K** represents the KF gain, **z** is the measurement vector, **H** is the measurement matrix that relates tracker state variables to the measurement vector, given by:

$$\mathbf{H}_I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad \mathbf{H}_{II} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{19}$$

and **R** is the measurement noise covariance matrix which, in case of the described system, is a diagonal matrix, containing values of the measurement noise variance on the diagonal. As a result of the update stage, the posterior state of the tracker is calculated. From this state, variables representing the object position and size may be extracted, stored in the track, and they may be later used to perform further analysis (such as event detection).

It should be noted that the measurement vector may only contain data that can be measured directly with the sensor. In the case presented here, a camera is the sensor, and the object detection algorithm is a method of measuring the position and size of the tracked objects. One may be tempted to ‘measure’ the velocity (and acceleration) by differencing data from the current and the previous video frame. However, this approach is not valid, because there is no sensor that measures these variables and their estimation should be performed by the filter.

Selection of Kalman Filter Parameters

The three noise variances are the most important parameters that affect the tracker performance in tracking moving objects in the camera. Similarly to the background subtraction algorithm, there are no universal values for these parameters and they should be tuned for a specific tracking scenario. However, some guidelines may be provided in order to help the algorithm developers in optimizing the tracking system.

Error variance (**P**). Only the initial values have to be provided, the matrix is later updated by the filter. These values should be high because of an uncertainty of the initial state. For example, we know the initial position and the size of the object, but we don’t know its velocity, so the latter may get a higher noise variance. The error variance will decrease as the filter converges. A value of 1 is often used for the initial error variance for all variables. However, since the initial estimates of the velocity, acceleration and size change are inaccurate, it is justified to set a larger variance for these variables.

Process noise (**Q**). Variance of the process noise determines the balance between the deterministic and the random component of the modeled process. A low variance means that we expect the dynamic model to describe the process accurately. For example, if we know that the tracked objects always move with a constant velocity, then we can rely on the first order dynamic model, by setting a low variance of the process noise. In practice, the velocity of moving objects is not constant. If the noise variance is set to a low value and the object stops, the tracker may overshot and lose the object. Therefore, a higher variance value is needed in this case. On the other hand, the variance set too high will cause the filter not to trust the dynamic model and to assume that the movement is more random in nature. As a result, random changes of the tracker state may result in losing the tracked object. Therefore, tuning the process noise variance requires finding a value that balances both cases. For example, if a camera shows pedestrians on a sidewalk, small velocity changes may be expected, so lower variance values may be sufficient. If a camera observes a busy road intersection, higher variance values will be needed. Additionally, different values may be used for separate variables, e.g., velocity changes may be more random than size variations.

Measurement noise (**R**). Variance of the measurement noise defines a confidence on the measurements provided to the tracker for updating its state. If the measurements are less accurate (more noisy), a higher variance should be set. This parameter sets the balance between the state predicted by the filter and the measurements. For example, a low variance of the measurement noise means that we are confident in the measurement accuracy, so the predicted state will be largely ignored in the update phase. Conversely, a high variance means that the predicted state is more important than the measurement. In practice, this variance cannot be set too high, because the filter will ignore the measurements and the tracked object may be lost if it changes its velocity or the direction of movement. On the other hand, the variance that is set too low may disrupt the tracker by incorporating inaccurate measurements. Similarly to the process noise, different values of the measurement noise may be set for individual variables.

One may argue that measurements made with the object detection algorithm are accurate, so the variance of measurement noise should be low. This is not the case. In an ideal situation, a specific point of the object should be tracked. However, in the discussed scenario, a center point of the blob is tracked and the position of this point within the object may change on a frame-by-frame basis. For example, when a walking person is tracked, the shape of the blob changes in different phases of the movement, so the position of the

blob center point fluctuates, increasing the measurement inaccuracy. Therefore, the measurement noise variance is usually set to a higher value when the tracked objects are mostly humans, and to a lower one in a scene where rigid objects (such as vehicles) are tracked. Additionally, the object detection algorithm is a 'noisy sensor' because of background subtraction imperfections, as described in the previous Section.

It can be concluded that setting proper values of the noise variances in order to achieve the desired tracking accuracy is a time consuming process. In practical applications, it should be performed by means of analysis of video recorded from the same source that will be used for the tracking. A number of representative object tracks should be extracted from the video with the background subtraction and object detection, and corrected manually. The tracking algorithm may be then run on the recordings, and the predicted states may be compared with the ground truth data. For example, a mean squared error for the position may be computed for a single track:

$$e = \frac{1}{N}\sum_{k=1}^{N}\left(x_{p,k} - x_k\right)^2 + \left(y_{p,k} - y_k\right)^2 \tag{20}$$

where (x_k, y_k) is the ground truth position of the object in k-th video frame, $(x_{p,k}, y_{p,k})$ is the predicted object position obtained from the tracker, N is the number of the analyzed states. The error value may be computed for all analyzed tracks. In order to obtain an optimal set of parameters, a grid search may be performed by repeating this procedure for different sets of parameters, usually on a logarithmic scale (e.g., 10^{-5}, 10^{-4}, …, 10^{-1}). With this method, optimal values resulting in a minimized error may be found and used for the tracking, and later tuned if necessary.

In practice, it is not always required to set the noise variance values in **Q** and **R** independently. In many cases, only two variance values are defined: one for the process noise and another for the measurement noise. Therefore, **Q** and **R** are diagonal matrices with constant values on each diagonal. In this case, the results of Kalman filtering depend only on the ratio of both variances, not on their absolute values. The result obtained for the variances set to e.g., (10^{-1}, 10^{-3}) will be identical to those obtained with the variances (10^{-5}, 10^{-7}). This feature of KFs (which is rarely mentioned in the literature) may simplify the process of the algorithm tuning, as the number of parameters is practically reduced to a single ratio. However, in some more difficult scenarios, it may be

beneficial to use separate variance parameters for the position, size, velocity, etc.

Tracking Conflicts

So far, a simplified case of tracking objects that do not interact witch each other, was considered. In practice, the detected blobs often merge together, forming a single blob (an occlusion in the camera view) and they also may split (e.g., a person leaving their luggage). Resolving such cases are the most challenging problem in object tracking in video, and this problem does not have a definitive solution. The KF is not able to handle such cases by itself, it is the job of the tracking algorithm to provide an accurate measurement that takes tracking conflicts into account. In this section, some attempts to solve this problem with the KF will be discussed and a more sophisticated approach, using particle filters, will be discussed later in the Chapter.

Figure 5. An example of a tracking conflict. Left: a white van is tracked without a conflict. Center: a single blob that represents two vehicles (in occlusion) is used for updating the KF. Right: the tracker state after updating the KF with an incorrect measurement.

When the predicted positions of all trackers are compared with the blobs detected in the current video frame using the background subtraction and object detection algorithms, a relationship matrix is constructed [Czy08]. If a tracker/blob is related to more than one blob/tracker, a *tracking conflict* occurs. As a result, it is not possible to obtain a direct measurement for updating the KF, because there is no single blob that represents the whole tracked object, and only that object. An example is shown in Figure 5. The object that was previously tracked in a non-conflict situation, now enters the conflict, because the predicted position of the object is related to a blob that represents two objects, so the measurement obtained from the object detection algorithm is inaccurate. Therefore, the updated state of the tracker is

contaminated with a wrong measurement. If the conflict persists for a large number of frames, the tracker loses the original object and starts to track the merged blob.

The following conflict situations may occur in practice.

A merge due to a temporary occlusion. This situation happens very often when paths of two or more moving objects cross. From the perspective view of a camera, these objects are in the same position (they overlap), so a single blob is detected. If the common blob is used as the measurement, the tracker state, and especially the variables related to the object size, are distorted (Figure 6). Resolving this type of conflict requires finding a region of the blob that represents the object. It is problematic if the object is mostly or even completely occluded by another one. The KF may help in resolving short-term occlusions. The blob containing merged objects may be used as a measurement, but since it does not represent a single moving object, the measurement error may be very high. Therefore, if such type of conflict is detected, the measurement error variance needs to be increased, in order to let the filter rely mostly on the dynamic model. Due to the inaccuracy of these measurements and possible changes in the velocity of objects movement, the filter is likely to diverge after a large number of analyzed frames in which the conflict persists. More advanced approaches to this problem attempt to search for the image region representing the tracked object within the merged blob, usually employing algorithms based on color histograms, such as CamShift [Bra98].

Figure 6. An example of the occlusion. Left: two vehicles are tracked separately. Center: the same blob is used for the (inaccurate) measurement for both objects. Right: the tracker resumes correct tracking of the vehicle after a short-term occlusion.

A split due to separating objects. This situation occurs when two or more objects that were moving together, become separated, for example: persons that were walking in a group, but at some point walked away from each other; a person leaving luggage and walking away, etc. When objects split, each of them should get their own tracker (Figure 7, center). Such a case may be

detected by examining changes in tracker size, because when these objects move away from each other, the tracker size increases significantly and the space between the blobs increases. If such a situation is detected, new trackers should be created for the separated objects.

A split due to fragmentation. In this situation, a single object is split into two or more blobs (Figure 7, right). Fragmentation may result from occlusion by the environment (e.g., a vehicle driving behind a post), but it often results from imperfections of the background subtraction algorithm (such as the camouflaging, Figure 3). Contrary to the previous case, a bounding box containing all the related blobs should maintain the size of the complete object. Therefore, a merged set of all related blobs can be used as the measurement for the filter update.

Figure 7. Examples of object splitting. Left: a single tracker for a group of objects moving together. Center: a split caused by the separated objects (a vehicle and three persons start to move independently). Right: a split caused by the fragmentation (two blobs for a single object, because of the background subtraction errors, as shown in Figure 3).

Tracking conflicts may become much more complicated. For example, a fragmented object may be at the same time occluded by another object, so a combined ‘merge-split’ conflict occurs, reflected by a complex relationship of multiple trackers to multiple blobs. Resolving such conflicts becomes an even more difficult challenge. The only type of conflicts that may be controlled to some degree are the splits caused by the fragmentation. A countermeasure to this problem is tuning the background subtraction parameters in order to reduce the blob fragmentation as much as possible.

The approach presented here is a simple one and it will not work reliably in busy scenes, with a large number of persisting conflicts. It is also possible to use the predicted state from the KF as a noisy measurement [Szw11], but it only works for short-term conflicts, as the KF state degenerates quickly. Some more complex solutions were proposed, for example, in the algorithm by Bose et al. [Bos07], a graph of trackers assigned for each blob is constructed,

coherency of tracks is computed and tracks are merged or split according to the results.

Computational Complexity and Final Remarks

The KFs are relatively simple in terms of computation, as they rely on basic linear algebra routines with small (and often sparse) matrices. Therefore, the tracking algorithm is significantly less resource-demanding than the background subtraction. Implementation of the KF algorithm should utilize a computing framework with an efficient implementation of the linear algebra routines, such as BLAS. In terms of accuracy, the KFs work well in scenes with objects moving mostly separately, with infrequent and short-term occlusions. A parking space is a good example of such a scene. As the number of conflicts increases, the performance of the KF trackers deteriorates. In very busy scenes, in which it is difficult to identify separate objects because of constant merging and splitting (for example, a street in a centre of a large city, during rush hours), it is practically impossible to track objects with KFs, so other algorithms should be used. Methods based on the optical flow algorithm [Luc81, Hor81] are commonly used for this task, but they are computationally complex and difficult to implement for a real-time video analysis.

The main drawback of KFs is that they assume a normal distribution of the tracked variables and they are able to provide only a single hypothesis of the tracked object state, which is not sufficient for resolving complex conflicts. Also, the KFs model only linear dynamic processes. Some modifications to the KF algorithm, namely the extended Kalman filter (EKF) and the unscented Kalman filter (UKF), were proposed [Wan00] in order to remove some of these limitations. They have their place in object tracking algorithms, although they are more complex and difficult to implement.

TRACKING WITH PARTICLE FILTERS

Particle filters (PF) are sequential Monte Carlo methods based on point mass (or ‘particle’) representations of probability densities, suitable for modeling nonlinear and non-Gaussian systems [Aru02, Ris04]. Application of PFs for object tracking may help to avoid some problems present in the KF approach, related to tracking conflicts, at the cost of a significantly higher computational time. Specifically, while the KF models only a single

hypothesis of the state of a tracked object, PFs provide multiple, weighted hypotheses, which may be useful e.g., in resolving tracking conflicts. PFs are used for object tracking, e.g., in radar applications [Gus02]. They are less common in object tracking in video, mostly because of their high computational complexity, which translates to a long processing time and higher energy consumption. Example implementations of the PF in video tracking were presented by Nummiaro [Num03] and Czyz [Czy06]. The PFs are also used in more complex tracking scenarios, such as the Condensation algorithm, used for tracking curves in a dense visual clutter [Isa98]. A detailed statistical background theory on PFs may be found in [Aru02] and [Ris04], and a practical tutorial is presented in [Lab15]. A short description of the algorithm is presented below.

Particle Filters

A particle tracker models the probability density function of the object state with a set of *N particles*. Each particle is described by its weight π and a vector of state variables, similarly to the KF. In the example presented here, the same variables as for the KF algorithm, will be used (Eq. 8). The sum of weights of all particles is always equal to one. Similarly to the KF, two main stages – the prediction and the update – are performed. The former begins with resampling the particle set. A new set of *N* particles is created by randomly selecting *N* particles from the original set, with a replacement. Each particle may be selected zero, one or more times, and particles with higher weights have a higher chance to be drawn. The resampled set contains mostly particles with high weights, representing 'more probable' states of the object. The resampling is performed by calculating the cumulative weights of the particles, by generating *N* random numbers from an uniform distribution ranging from 0 to 1, and selecting the first particle with a cumulative weight equal to or higher than this random value [Isa98]. Next, the state of all particles in the resampled set is propagated using a dynamic model. In the algorithm described here, the same dynamic model as in the Kalman tracker (either the first or the second order one) is used. In the next step, random values (a 'drift') are added to the predicted state:

$$\mathbf{s}_t = \mathbf{A}\mathbf{s}_t^- + \mathbf{w}_t \tag{21}$$

where **s** is the predicted state, $\mathbf{s}^-$ is the state after resampling, **w** is the vector of random values from a normal distribution with a zero mean and with variances $\boldsymbol{\sigma}^2$. Therefore, if the same particle was selected multiple times from the original set, the dynamic model propagates all particles to the same state, and the random process adds uncertainty to the prediction phase, resulting in dispersion of these particles. The variance values define a spread of each variable. Similarly to the KF, a higher variance is needed if movement of the object is expected to deviate significantly from the dynamic model.

The measurement phase recalculates the weights of all particles and normalizes them to the unit sum. In the tracking application, the particle weight should reflect the similarity between the predicted and the measured state. An example using color histograms for computing weights will be described in the next Section. Finally, an estimate of the object state may be computed as a weighted mean of the particle set (Figure 8):

$$\bar{\mathbf{s}}_t = \sum_{i=1}^{N} \pi^{(i)} \mathbf{s}_t^{(i)} \tag{22}$$

Figure 8. Tracking a white van with the particle filter. Left: predicted states of the particles, representing the object position, are shown as dots. Center: the updated particles (only the particles with sufficiently high weights are retained). Right: the mean state of the tracker, shown as a bounding box.

Measuring Particle Weights with Color Histograms

While the tracking algorithm based on PFs is conceptually simple, the main challenge is obtaining the measurements for updating the particle set. In the KF, this procedure was performed with background subtraction and object detection. As it was discussed previously, it is not possible to obtain a reliable measurement with this approach in case of tracking conflicts. The PFs provide multiple hypotheses of the object state, which may be useful for conflict

resolving, but a method for measuring the object position and size is needed. In order to compute a similarity between the tracked object and the predicted region in the video frame, color histograms may be used [Num03]. The target object histogram is stored in the tracker. For each particle, the histogram of the image region described by the particle state is computed and normalized, and its similarity to the target histogram is calculated and used for the weight updating. Efficient histogram calculation requires that histograms are invariant of brightness changes (so they allow tracking objects in scenes with different lighting), but at the same time, they are specific for a given object (i.e., objects that look different also have significantly different histograms).

Color histograms may be computed in a variety of color spaces (RGB, normalized RG, HSV, HLS, etc.). In the algorithm presented here, the improved HLS (iHLS) color space, introduced by Hanbury [Han03], was used. The main advantage of this color space is that it removes the dependency of the saturation on the brightness. Therefore, it is better suited for the analysis of camera images [Bla06].

Various methods of histogram calculation in the iHLS color space are possible: three separate 1D histograms, a combined 1D histogram, a 2D histogram with only selected channels, etc. For example, Hanbury proposed a merged histogram of the hue and the brightness, with bin values weighted by the saturation [Han03]. However, in the preliminary experiments on the tracking algorithm presented here, this approach did not work well and it resulted in losing the tracked objects. It was found that a 2D histogram constructed from the hue and the saturation channels in the iHLS color space works with a good accuracy. Values of the hue range from 0 to 360 degrees, values of the saturation range from 0 to 1. The number of histogram bins should be chosen so that the histogram reflects significant differences between different objects, but at the same time, the histogram is not too detailed. The number of bins which is too high increases the computation time and memory requirements, and makes the histogram comparison more difficult (the values are spread among too many bins). As a good compromise, a histogram consisting of 64 ranges for the hue and 8 ranges for the saturation, was chosen. Because the hue is meaningless for low saturation (almost gray) pixels, it is replaced with the brightness values, scaled to 0-360 range, for pixels falling into the bins representing the lowest saturation values. After computing the histogram for all pixels in the image region described by a given particle, it is normalized to the unit sum.

The position of an object is represented by a center point (x, y) and the size (w, h). The image region represented by a given particle state may be

therefore visualized as an ellipse with the center point (x, y) and the axes (w, h), and all pixels within this ellipse are used for the histogram computation. The ellipse is not rotated (its axes are always parallel to the image borders). An incorrect estimation of the object size may result in including the background pixels into the histogram computation. When too many background pixels are included, the risk of 'sticking' the tracker to the background increases. In order to avoid such problem, it is possible to weight the pixels used for the histogram computation. A similar approach was proposed by Nummiaro [Num03]. Pixels close to the ellipse border receive lower weights, so that the background pixels on the edges of the analyzed region have a smaller contribution to the histogram. The histogram weight r of each pixel is computed from its distance from the ellipse center, as follows:

$$r = \max\left(1 - \frac{(x_p - x_c)^2 + (y_p - y_c)^2}{\left(\frac{w}{2}\right)^2 (x_p - x_c)^2 + \left(\frac{h}{2}\right)^2 (y_p - y_c)^2}, 0\right) \tag{23}$$

where (x_c, y_c) is the ellipse center, (x_p, y_p) are the pixel coordinates, (w, h) are the ellipse axes lengths. The value of r is one for the ellipse center, it decreases towards zero when the distance from the center increases, and it is zero for pixels outside the ellipse.

For calculation of the similarity between the histogram computed from an image region and the target histogram stored in the tracker, various metrics may be used, e.g., correlation, intersection, Chi-Square, Bhattacharyya, Hellinger, quadratic, etc. [Zha14]. It was found during the experiments that comparable results were obtained with different metrics. Therefore, for performance reasons, the Bhattacharyya metric was used. The distance between two histograms H_1, H_2 consisting of N bins is:

$$d = \sum_{i=1}^{N} \sqrt{H_1(i) H_2(i)} \tag{24}$$

The particle weights are calculated from the histograms distance using an exponential weighting function [Num98]:

$$\pi = \exp\left(\frac{-d^2}{2s^2}\right) \tag{25}$$

Here, the value of s^2 serves as a parameter controlling the weighting function shape. Increasing this value allows for larger distances between the histograms. In the experiments presented here, this value was empirically set to 0.3. After the computation of weights for all particles is done, the weights are normalized to the unit sum (therefore, the normalizing factor of the Gaussian function was omitted from Eq. 25).

After the measuring stage is finished, the mean tracker state may be computed from all particle states, using Eq. 22. For the final step, the target histogram of the object should be updated:

$$H_{T,n} = (1-\alpha)H_{T,(n-1)} + \alpha_p H_\mu \tag{26}$$

where $H_{T,n}$ is the target histogram stored in the tracker in frame n, H_μ is the histogram computed from the image region corresponding to the current mean tracker state, α_p is an update factor which determines the histogram update ratio. The target histogram is computed when the tracker is initialized. Updating the target histogram is necessary in order to take changes of the object appearance into account, e.g., if the object changes its orientation relative to the camera, or it moves to an area with different light. In order to prevent distorting the target histogram with incorrect results, e.g., when the tracked object is partially occluded, this operation should be performed only if the distance between histograms $H_{T,n-1}$ and H_μ is below the threshold (typically in the range of 0.25 to 0.4).

Selection of the Tracker Parameters

Performance of the PF used for object tracking in video is affected mainly by two parameters. The *number of particles* per single tracker determines how many samples of the probability distribution function are taken. The choice of this value depends on the number of state variables, because the particles have to cover a range of variation of each variable. The proposed state vector contains 8 or 10 variables, depending on the dynamic model, so a high number of particles is needed to fill the variable space. If too few particles are used,

the state prediction becomes inaccurate and unstable (a large variation between frames is observed). A higher number of particles increases the tracking accuracy, but it also significantly affects the processing time. In practical situations, 500 to 2000 particles per one tracker should be used, depending on the variability of object movement and the frequency of tracking conflicts. A value of 500 is recommended for the initial experiments.

Noise variance determines a spread of the particle states during the prediction phase, this value should be set in order to cover the expected deviations from the dynamic model. For example, the variance of the position noise should be set so that all particles cover the image region in which the object may appear, but they do not extend significantly beyond this region. Similarly to the KF, a low variance means that the object is expected to move according to the dynamic model. If the value is too low, the actual position of a tracked object may not be fully covered by any particle, resulting in tracking errors. On the other hand, the variance that is set too high causes the particles to spread too wide, increasing the risk of taking a wrong image region as a measurement and losing the tracked object. Different noise variances may be used for each variable type (the position, velocity, size, etc.). The values of variance in PFs are usually higher than in the KF. It is recommended to set a higher variance to the position and velocity noise (e.g., 1.0 – 10.0) and a much smaller variance for the position noise (e.g., 10^{-6}). These values need to be tuned for a specific scenario and techniques similar to those described for KFs (e.g., a grid search) may be employed.

The *weighting function parameter* s^2 in Eq. 25 sets the required similarity of the object histograms. With higher values, larger differences between the histograms are allowed. Lower values may help in separating objects with a similar look, but they may result in losing the tracked object if its appearance changes. A value of 0.3 may be used as a starting one.

The *histogram update threshold* defines the maximum distance between the computed and the target histogram that allows for updating the latter. This value should not be too low, because the target histogram will not be updated if the object appearance changes, (e.g., when its orientation relative to the camera is changed) and it should also not be too high, because the target histogram may be corrupted when the tracked object is occluded. Typical values are 0.3 to 0.5. The *histogram update rate* (α_p in Eq. 27) controls the speed of adaptation of the target histogram to changes in the object appearance and it is typically set to 0.01 – 0.05.

Implementation Considerations

The PFs are significantly more demanding in terms of the processing power and memory requirements than the KFs. The processing time depends mainly on the number of particles per one tracker, and the number of tracked objects. The prediction and updating phases need to be performed for each particle. In case of the tracking algorithm presented here, computing the color histogram for each particle and comparing it with the target histogram results in a high computational load. Therefore, a real-time implementation of the PF is problematic even for a single object. Parallel processing systems (e.g., GPUs) may reduce the time of prediction and update phases, but some elements of the histogram computation (normalization, reduction to the sum, etc.) are sequential, and the processing threads often become stalled at the synchronization points, so the gain in the processing time is not as high as it may be expected.

The main advantage of employing the PF for object tracking in camera video is that it can be used without the background subtraction phase. Therefore, it may be applied in scenarios in which performing background subtraction is not possible because of a non-static camera view. An example of such application is object tracking in video obtained from cameras on UAVs (‘drones’). The limitation of this approach is that it is not easy to initialize the tracker automatically, it has to be done either manually, or by providing a target template. However, there are some works on automatic initialization of PF trackers, e.g., by Czyz et al. [Czy06]. Another possibility is to detect an object in a fixed camera and to obtain its color histogram and use it as the object template. The tracking algorithm may then monitor the selected entry areas in the non-fixed camera view, compute their color histograms and compare them with this template. If a sufficiently high histogram similarity is detected, this region may be used for initializing the tracker. The same procedure may be used for ‘restarting’ a tracker if the tracked object temporarily leaves the camera view [Num03].

Comparison with Kalman Filters

A decision whether a potentially improved accuracy of the PF tracker compared with the KF justifies the increased demands on computation resources, depends on the average numbers of simultaneously tracked objects and the frequency of observed tracking conflicts. If the conflicts are rare and

short-term, KFs are expected to perform reasonably well and employing PFs in this case is not going to provide a significant performance boost. However, if object occlusions, splitting and fragmentation are frequent, the PFs may improve the tracking accuracy in a significant way. For example, in case of object occlusions (blob merges), the KF provides a single prediction of the object state, and it is not possible to verify the accuracy of this hypothesis and to obtain a better measurement for the tracker update. On the other hand, the PF provides multiple hypotheses that may be verified e.g., by comparing color histograms, and the particle weights reflect the accuracy of these predictions. Therefore, the particle tracker is able to find the optimal estimate of the object state. Compared with algorithms such as CamShift [Bra98] that perform a 'blind' search for a region with the best matching histogram, the particle tracker uses a dynamic model to predict the most probable object state. Of course, this approach will not work well if an object is completely occluded for a prolonged time, or when it is camouflaged in the background. However, if the occlusion is partial and temporary, it will result in high dispersion of the particles during the conflict, but after the object becomes fully visible, the particle set should be able to refocus on it. An example of a successful tracking in the described case is presented in Figure 9. When the tracked object is partially occluded, the particle set is still focused on the visible part of the object, and when the occluding object moves away, the tracker readjusts itself to the tracked object. The tracker is even able to handle short-term full occlusions, provided that the occluding object has a sufficiently different color histogram from the target. Obviously, there is a risk of 'stealing' the tracker by another object with a similar appearance. This effect cannot be fully eliminated, but it may be reduced by tuning the algorithm parameters.

Figure 9. Phases of tracking the dark (parked) vehicle that is partially occluded by the white van moving in front of it. Top: a 'swarm' of particles modeling the object position (after the filter update). Bottom: the mean tracker state as a bounding box.

Object fragmentation is usually not an issue with the PFs, because they are able to recover the image region containing the object. On the other hand, permanent object splitting is not handled by the particle tracker itself. In case of splitting the object into two or more separate ones, the tracker will stick to the object which has the highest similarity with the target histogram. The remaining objects will be lost, so a dedicated procedure is needed to handle the split by assigning a new tracker to the separated objects.

The initialization of the tracker for new objects that appear on the scene, and for objects left behind after the split, is an obvious problem with this approach. One solution is to use the background subtraction and blob extraction procedures to find objects not assigned to any tracker and initialize their tracking with this data. The drawback of such an approach is that the computation time increases significantly, because now two computationally complex algorithms: background subtraction and particle filtering, are employed. However, incorporating the background subtraction procedure into the PF tracker has an additional advantage of removing the influence of the background pixels on the tracking accuracy. In a standalone particle tracker presented here, color histograms are calculated from all pixels inside the ellipse determined by the tracker. This may also include the background pixels, for example, from the area between legs of a walking person. As a result, the histograms are distorted by the background pixels, which increases the risk of losing the tracked object if a tracker sticks to the background (it may happen e.g., when the object is mostly occluded). In this case, the background subtraction stage may be used to mask out the background pixels, removing them from the histogram calculation. Therefore, this modification may increase the tracking accuracy.

A COMBINED TRACKING ALGORITHM

As it was previously discussed, the KF-based tracking algorithm provides a sufficient accuracy in the non-conflict cases. The KFs are also able to cope with infrequent and short-term conflicts, but with a decreased accuracy. Using the PFs for the non-conflict cases seems to be unnecessary, because the computation time is high and there is no substantial gain in the tracking accuracy. However, the situation changes when the tracking conflicts are more frequent and they last longer. In these cases, the PFs are expected to perform better than the KFs in locating the tracked object within the conflict area. In practical video surveillance scenarios, a mixed situation is often observed:

individual objects may be identified in most stages of their movement, but there are also conflict situations that the KFs are unable to handle. Therefore, based on the discussion presented in the previous Sections, a mixed tracker that combines the useful features of both the KF and the PF, is proposed. An overview of the system is presented in Figure 10. The idea is to use the KF for tracking the object as long as no complex conflicts occur, and to employ a simplified PF when such conflicts have to be resolved. Each moving object is represented by a tracker which consists of three components: the KF, the PF and the target color histogram. Additionally, the tracker stores a flag indicating whether a conflict was detected in the previously analyzed frame, and the tracker state saved from the latest non-conflict frame. A theoretical background for each component has been provided earlier in the text.

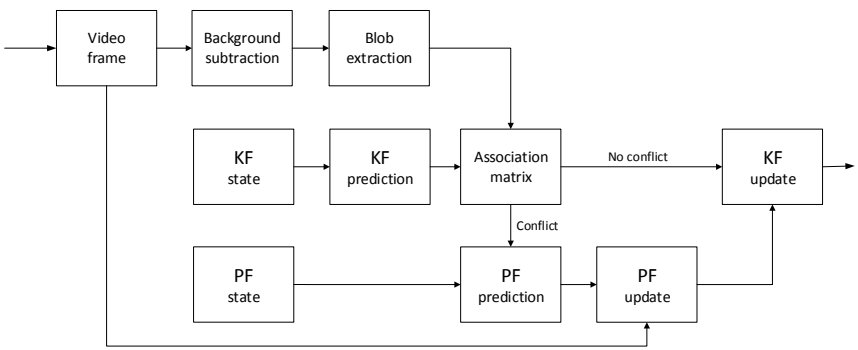


Figure 10. Block diagram of the proposed combined object tracking algorithm.

Tracking Objects without Conflicts

The analysis of a video frame begins with the background subtraction. In order to reduce the computation time, the image may be downscaled. The foreground/background mask is morphologically processed and blobs of foreground objects are extracted. Next, the association matrix is constructed by comparing the bounding boxes of the detected blobs with the predicted tracker states, obtained from the KFs. For each blob that is not related to any tracker and has a sufficiently large size, a new tracker is created. The KF is initialized with the detected position and size of the blob, and with zero velocity and size change. The initial target histogram is computed from the blob area. The PF is initially inactive. If there is a non-conflict relationship between one blob and one tracker, this blob is used as a measurement for updating the KF.

Additionally, the histogram of the blob and its distance from the target histogram, are calculated. If the distance exceeds a certain threshold, it means that the blob does not represent the tracked object. Therefore, the tracker is removed. If the histogram distance is sufficiently low, the target histogram is updated (Eq. 26). The PF is inactive during the non-conflict tracking.

Handling Fragmentation and Splitting

A relationship of one tracker to more than one blob indicates either the fragmentation, or the object splitting into two or more separate tracks. For the analysis of this case, a bounding box encompassing all the related blobs is calculated. If the conflict results from a fragmentation, the size of the combined bounding box remains similar to the object size stored in the tracker, and the distance between the histogram computed from all the related blobs and the target histogram is also small. Therefore, the combined bounding box is used as a measurement for the KF update. It is recommended to increase the measurement noise variance for this case, because fragmentation makes the measurement inaccurate by nature. The target histogram is not updated.

Tracker splitting into separate objects may be detected by observing that the size of the combined bounding box of the matched blobs increases in the successive frames, and also the distance between the individual blobs increases. When the size of the combined blob is larger than the original tracker size by a factor exceeding a threshold, a split is detected. Histograms and sizes of the partial blobs are compared with the tracker data obtained before the conflict occurred. If a match is found between the merged blob and a single tracker, this blob is used for updating the tracker and new trackers are created from the remaining blobs. An additional analysis is also needed in order to detect splitting and fragmentation occurring at the same time.

Handling Occlusions and Complex Conflicts

The remaining conflict situations represent cases in which more than one tracker is related to one or more blobs. Such a situation occurs during the occlusion, and also in complex cases of the occlusion coexisting with the fragmentation and splitting. These are the most difficult conflicts to resolve. Generally, the matched blob (or the combined blob) is larger than the individual tracker states, so the main problem is to find a region inside the

blob that contains a given tracked object, and to use this region for the tracker update. The PF is used for this task, because it allows for verification of the predicted state (by computing the distance between the color histograms).

When this type of conflict is detected, the tracker activates the PF. A simplified state vector, containing only the position and the velocity, is used in particle tracking:

$$\mathbf{s} = [x, y, \dot{x}, \dot{y}]^T \tag{27}$$

Initial values of these variables are copied from the last KF state. The object size and its changes are not taken into account, because it is not possible to obtain the size measurement when a tracked object is occluded. It is therefore assumed that the object size does not change significantly during the conflict. Reducing the dimension of the state vector allows for using a smaller number of particles per tracker. The prediction phase is also simplified:

$$\begin{aligned} \dot{x}_t &= \dot{x}_{t-1} + v_x \quad & \dot{y}_t &= \dot{y}_{t-1} + v_y \\ x_t &= x_{t-1} + \dot{x}_t \quad & y_t &= y_{t-1} + \dot{y}_t \end{aligned} \tag{28}$$

where v_x and v_y are noise values from the independent normal distributions. The process noise variance should allow for expected deviations from the dynamic model, but it should also be sufficiently low, in order to keep the particles within the blob borders. Since it is known that the object is inside the blob (assuming that the results of background subtraction are accurate), the particles should not extend beyond the blob borders. Therefore, a proper choice of the noise variance keeps the spread of particles within the blob limits.

Verification of the hypotheses (calculation of the particle weights) is performed as before, by comparing color histograms computed for each particle, with the target histogram stored in the tracker. If the overlap between the tracked objects is small and these objects differ in appearance, it may be expected that the particles having the highest weight in the set describe the image region containing the tracked object. The computation of weights is done according to Eqs. 23-25, but the particles having the estimated position beyond the blob limits automatically receive a zero weight.

After the update phase is finished, the mean posterior state of the PF is computed (Eq. 22) and used as a representation of the object position, with the size retained from the original state (before the conflict). This result is then

used as a measurement for updating the KF, possibly with an increased value of the noise variance. Tracking with the PF continues until a non-conflict state is detected and the tracker switches back to the KF only. The target histogram remains unaffected during tracking with the PF.

Tracking Objects with the Combined Tracker

The complete tracking procedure (Figure 10) may be summarized with the following steps.

1) The background subtraction and object detection stages are performed, and bounding boxes of the blobs are obtained.
2) Tracker prediction using the KF is performed. The association matrix is constructed from the detected blobs and the predicted KF states. Groups of related trackers and blobs are obtained. For blobs without a matched tracker and having a sufficient size, new trackers are initialized with the KF, and the initial target histograms are computed.
3) Trackers that are in the conflict are found. The current state of each tracker (a conflict or no conflict) is compared with the flag stored in the tracker. If the state changes from ‘no conflict’ to ‘conflict,’ the PF is initialized with the last state of the KF.
4) In case of a conflict, the PF is used to obtain the mean state by performing the prediction phase, computing the color histograms of the predicted regions, and their distances from the target histogram. The mean state from the PF is used as the measurement. If there is no conflict, the matching blob (or a merged blob, in case of fragmentation) is used as the measurement.
5) The KF is updated with the measurement obtained either way. The posterior state of the tracked object is obtained from the updated KF.
6) If there was no conflict in the current frame, the color histogram of the posterior state is computed and used for updating the target histogram in the tracker, if the distance between the histograms is below the threshold. Also, the current state of the KF is stored in the tracker for future reference.

This process is repeated for all the tracked objects, and then for the consecutive video frames.

EXPERIMENTS

In order to perform a thorough evaluation of any object tracking algorithm, it has to be tested on a large number of object tracks, obtained from a set of video recordings representing various tracking scenarios, with varying complexity of objects movement. In practical applications, the tracking system has to be verified also on real recordings obtained from the target surveillance system. For each video, the ground truth data describing the exact position and size of all objects in each video frame, has to be available. Creation of such a set requires a substantial amount of work. Finding a ready to use benchmark set suitable for object tracking evaluation is also problematic. Therefore, in this Section, a simplified test procedure utilizing only one video recording and a single object, will be presented. The aim is to illustrate the testing procedure, and to provide an overview on the accuracy of the tracking methods presented here. However, it is by no means an exhaustive performance evaluation of these algorithms.

For a quantitative analysis of the tracker performance, various metrics are used [Ceh16]. Commonly utilized region-based metrics may be calculated by comparing a coverage of two image regions: the one obtained from the tracking algorithm, and the ground truth data. These regions are usually described with rectangles, denoted as **t** and **g** for the tracker and the ground-truth, respectively. Pixels situated inside these rectangles may be classified as:

- true positive (TP) results: $\mathbf{t} \cap \mathbf{g}$
- false positive (FP) results: $\mathbf{t} \cap \sim\mathbf{g}$
- false negative (FN) results: $\sim\mathbf{t} \cap \mathbf{g}$

Recall describes the part of **g** that was detected correctly. Lower recall corresponds to a higher number of FNs.

$$recall = \frac{TP}{TP + FN} = \frac{area(\mathbf{t} \cap \mathbf{g})}{area(\mathbf{g})} \tag{29}$$

Precision describes how much of **t** belongs to **g**, so a lower precision occurs when the number of FP is higher.

$$precision = \frac{TP}{TP + FP} = \frac{area(\mathbf{t} \cap \mathbf{g})}{area(\mathbf{t})} \tag{30}$$

Finally, accuracy describes the part of **t** that represents the correct detections. Both FPs and FNs contribute to a lower accuracy.

$$accuracy = \frac{TP}{TP + FP + FN} = \frac{area(\mathbf{t} \cap \mathbf{g})}{area(\mathbf{t}) + area(\mathbf{g}) - area(\mathbf{t} \cap \mathbf{g})} \tag{31}$$

Values of all these measures range from 0 (the worst) to 1 (the best), they may also be expressed in percents. None of these metrics is exhaustive, so all three of them need to be calculated and provided in the report. It should be pointed out that in general, it is not possible to lower the number of both FPs and FNs at the same time, by tuning the algorithm parameters. The precision and recall metrics are usually related with each other, and altering some parameters of the algorithm (e.g., the noise variances in the KF) often leads to changing both the number of FPs and FNs in the opposite direction. Therefore, when the precision increases, the recall may decrease, and vice versa. This effect is often visualized by plotting the precision and recall values in a single graph, as a function of the tested parameter, forming a receiver operating characteristic (ROC). The ROC curve is useful in finding a proper balance between the number of FPs and FNs.

Another useful measure is the distance error, measured as a distance in pixels between the center position of the tracker and the center point of the ground truth rectangle. Obviously, this measure does not take the size of the object into account, but it is useful for assessment of the object location accuracy. It is measured as an average of the squared distances obtained from N analyzed video frames:

$$derr = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\left(x_{t,i} - x_{g,i} \right)^2 + \left(y_{t,i} - y_{g,i} \right)^2 \right)} \tag{32}$$

where ($x_{t,i}$, $y_{t,i}$) is the center point of the tracker in i-th frame, and ($x_{g,i}$, $y_{g,i}$) is the center point of the object rectangle in the ground truth data. This measure is therefore a root-mean-square error (RMSE).

In the experiments described here, the most complex operations – the background subtraction and the particle filtering – were implemented using C++ code. The OpenCV library was used for input/output operations. The remaining procedures were implemented in Python. The experiments were performed on a standard desktop PC. A video from the PETS 2001 dataset [Pet01] was used for the evaluation, specifically the dataset 1, testing set, camera 1 (encoded as ds1_ts_c1). The recording consists of 3064 frames in 704×576 px resolution, 15 fps. Unfortunately, the ground truth data for this dataset is no longer available. Therefore, the data was created manually for one of the moving vehicles, using the free Viper-GT software [Mar02]. The tracking scenario is as follows. A white mini-van is tracked, its track is shown in Figure 4. The object appears in the camera view, entering from the left (frame #697), it moves to the right, passing a recently parked green vehicle (#806-882), then it stops on the right edge of the screen (#1002). It also passes by several walking persons, with a partial occlusion. Next, it reverses into the side alley (#1595-1960). Another (black) car enters the scene and turns into the same area, partially occluding the tracked vehicle (#2264-2494). Next, the tracked car moves toward the left side (#2485) and finally leaves the area (#2687).

The processing of each video frame started with the background subtraction, using the GMM algorithm described earlier. Five Gaussians per pixel were used, and the algorithm parameters were: $b = 0.5$ (Eq. 1), $\rho = 10^{-6}$ (Eqs. 2 and 3), $\alpha = 10^{-6}$ and $c_T = 0.05$ (Eq. 4), $T = 0.5$ (Eq. 5). The initial variance of each pixel model was set to 60 and the minimum variance was fixed at 5. The background subtraction mask was processed with the morphological opening followed by the morphological closing, both performed with a square, 3×3 kernel. From the obtained mask, blobs were extracted. In the next stage, one of three examined tracking algorithms was applied.

For the Kalman tracker, the predicted state was computed first, then blobs that intersected with the predicted rectangle were found. The first-order dynamic model was used. In case of an occlusion, the whole matching blob was used for updating the tracker. The same value of the prediction noise variance was used for all the predicted variables, and similarly, the same value of the measurement noise variance was used for all the measured variables. As it was discussed before, only the ratio of the noise variances influences the obtained results. Therefore, the performance measures were evaluated for several values of the variance ratio, defined as the prediction-to-measurement noise variance ratio. The results are presented in Table 1. Although the

differences are not large, it may be observed that all the metrics are lower for low values of the variance ratio, and for ratios larger than 10, no significant change in the results is observed. Since the differences were small, there was no point in plotting a ROC curve in this case. Based on the analysis, a ratio of 1, i.e., identical values for both noise variances, was chosen as the optimal value for this case. However, in practical scenarios, the described evaluation procedure should be performed on a large number of tracks obtained in varying conditions, in order to obtain a meaningful result.

Table 1. Performance metrics (%) obtained for the object tracked with the Kalman filter, as a function of the ratio of the process noise variance to the measurement n. v.

Variance ratio	Accuracy	Recall	Precision	Dist. error
10^{-3}	71.26	96.38	72.61	17.39
10^{-2}	73.02	98.48	73.65	16.35
10^{-1}	73.91	99.38	74.25	15.09
1	74.21	99.59	74.46	14.75
10^{1}	74.31	99.66	74.53	14.68
10^{2}	74.33	99.67	74.54	14.67
10^{3}	74.33	99.67	74.54	14.67

In case of the *particle filter* tracker, the analysis was performed using a PF containing 512 particles in the set. In the prediction phase, the noise variance was equal to 1 for the position and the velocity, and 10^{-6} for the size. These optimal values were found experimentally. The measurement phase was performed by computing the distances between color histograms, as presented earlier. Values: $s^2 = 0.3$ (Eq. 25), $\alpha_p = 0.05$ (Eq. 26) and the histogram update threshold equal to 0.3, were used. Finally, the proposed *combined tracker* was evaluated, using the same values of the KF and PF parameters as above. Additionally, the metrics were measured for the PF with a varying number of particles.

Table 2 presents the obtained results averaged for the complete object track, and Figure 11 shows the plots of all metrics vs. the frame number, illustrating how each tracker performs in different situations (a conflict or no conflict). The KF tracker has a near perfect recall, but the overall precision is below 75% because of the conflicts, during which the area covered by the tracker is larger than the actual object (a result of inaccurate measurements

during occlusions), so the precision decreases significantly. Therefore, the total accuracy of the KF is only 74.21%. The metrics obtained for the particle tracker are much worse, except for the precision, which is slightly higher than for the KF. However, the recall is only about 50%. Since the PF tracker does not utilize data on the detected blobs, the object is extracted from the image using color histograms only. When an object has mostly uniform color, it happens that a part of the object has a similar histogram to the whole object, and the particle representing this partial region receives a high weight, because of a small histogram distance. As a result, the tracker sticks to a section of the tracked object. This effect is sometimes described as a 'shrinking' of the tracker. As a result, the overall accuracy of the PF is only about 42%. When the PF is applied to object tracking in video, it should be expected that prediction of the object size is characterized by a significant error. However, even when only the object position is considered, the PF has a higher distance error than the KF. Despite the low metric values, the PF was still able to track the moving object without losing it. From Figure 11 it may be observed that the PF performs better than the KF in case of occlusions: all metrics, except for the recall, are higher. It should also be added that when the size noise variance was increased to about 10^{-4}, the tracked object was lost because at some point, the size of the tracked area started to grow in an unpredictable manner, because of introducing too much randomness into the dynamic process.

Table 2. Performance metrics (%) obtained for the three tracking algorithms: KF, PF and the proposed, combined algorithm, with a varying number *N* of particles per tracker

Tracker	Accuracy	Recall	Precision	Distance error
Kalman filter	74.21	99.59	74.46	14.75
Particle filter, N = 512	42.46	50.22	77.37	20.67
Combined tracker, N = 512	79.87	97.11	81.89	7.65
Combined tracker, N = 256	79.86	97.05	81.89	7.72
Combined tracker, N = 128	79.05	97.01	80.70	8.66
Combined tracker, N = 64	77.40	95.58	79.53	10.26
Combined tracker, N = 32	76.27	94.64	78.55	12.14

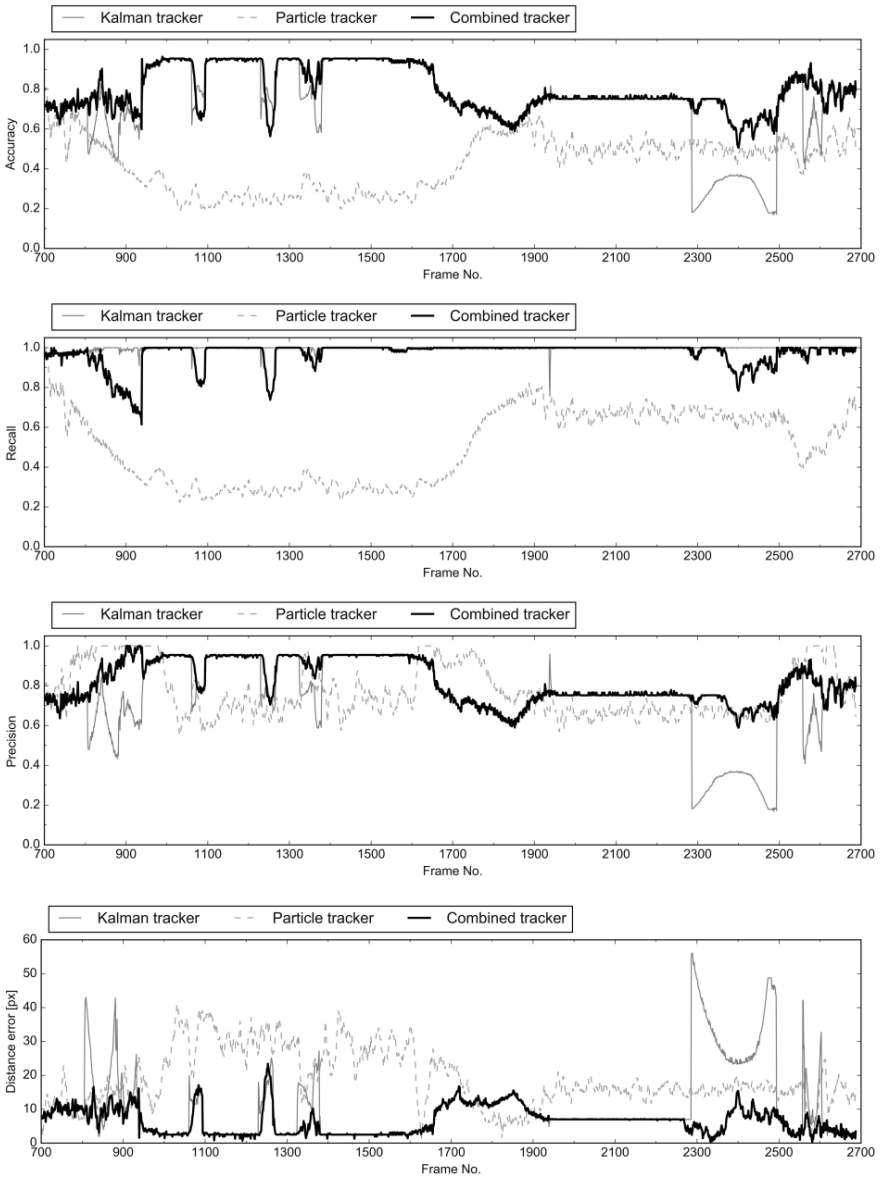


Figure 11. The performance metrics calculated for each frame of the tested video. Top to bottom: accuracy, recall, precision, and the distance error in pixels.

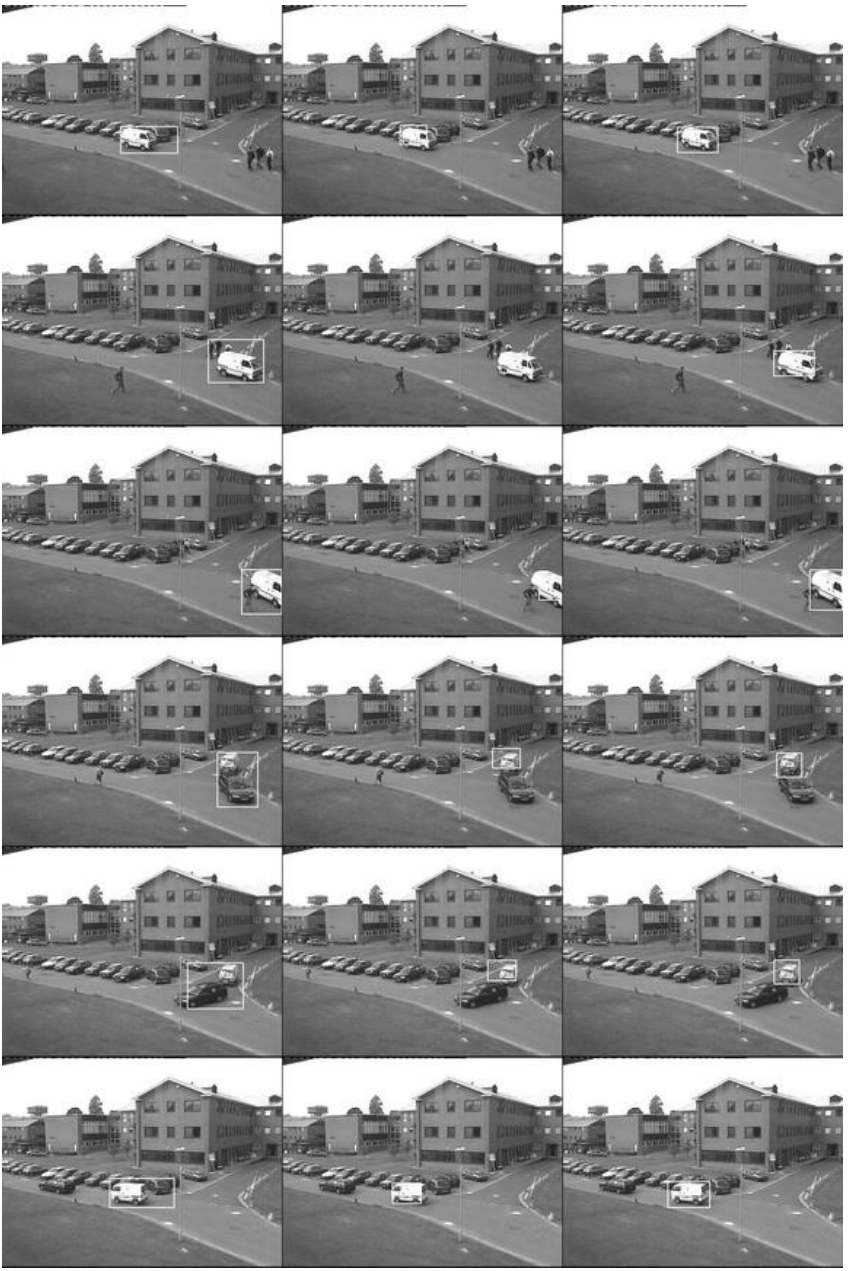


Figure 12. Tracking results in sample frames from the analyzed video set. Columns from left to right: the Kalman tracker, the particle tracker and the proposed, combined algorithm.

As it can be concluded from Figure 11, the KF provides accurate tracks when no conflicts occur, but the metrics decrease significantly during the conflicts, because there are no algorithms in the KF tracker able to cope with the conflicts. On the other hand, the PF tracker provides a more uniform performance, with higher metric values during the conflicts, but lower overall values than the KF. The main disadvantage of the PF tracker is its inability to track sizes of objects, even in the non-conflict cases. Therefore, an obvious conclusion is to merge the advantages of both methods and to employ the PF for resolving conflicts that occur while tracking with the KF. It was expected that the combined approach will increase the performance scores during the conflicts, and, as a result, the overall metric values will also increase. The experiments confirmed this assumption. The best scores were obtained for the combined tracker employing the PF with 512 particles, resulting with a 5.66 percentage point gain in the accuracy and a 7.4 p.p. gain in the precision, compared to the KF tracker. The distance error was reduced almost twice. A 2.3 p.p. decrease in the recall was observed, but the value is still high (97.11%). Of course, the obtained gain may depend on the scene and the frequency of conflicts. From the plots it may be observed that the combined tracker provides comparable scores to the KF when no conflicts are present, but it outperforms both the KF and the PF during the conflicts.

Selected frames from the analyzed video (presenting mostly the conflict cases), and the tracking results obtained from all the tested algorithms, are presented in Figure 12. The results provided in this Section serve only as an illustration of how the tracking algorithms are evaluated. In the presented scenario, employing the combined tracker resulted in improved accuracy scores. In real life applications, similar tests should be performed on a large number of tracks collected from the analyzed system. It should also be noted that incorporating the PF into the tracking algorithm increases the computation time. The computational load may be reduced by lowering the number of particles used per filter, at a cost of a reduced accuracy and tracker stability (as shown in Table 2), and also an increased risk of losing the object. Using 128 particles per filter still produces satisfactory metric values, but it is not recommended to use a lower number of particles.

CONCLUSION

The approach to object tracking in video surveillance systems presented here utilizes various algorithms: background subtraction, Kalman filters and

particle filters. Each of these algorithms has a number of parameters affecting their performance, and sometimes these parameters depend on each other. Therefore, developing an automated object tracking system is a complex task, involving both the system design and the parameters tuning. There are no universal settings for these algorithms, the parameters have to be chosen by the system designers based on two factors: the target scenario (the camera view, the type of moving objects, the character and intensity of movement, a frequency of conflicts, etc.) and their own experience. The aim of this Chapter was to collect knowledge on the algorithms that form the tracking system, and to provide some guidelines on influence of the algorithm parameters on the tracking accuracy in typical real life scenarios, in order to help the system designers in gaining the experience mentioned above. A methodology of performance evaluation of tracking algorithms was also described.

The algorithm that is most often used for object tracking, is based on the Kalman filter. The accuracy of this algorithm decreases during conflicts, such as occlusion or fragmentation. An alternative approach based on particle filters provides a higher tracking accuracy during short-term conflicts, but the overall performance is much worse than for the KF, and the computation time is significantly higher. Therefore, a combined approach was proposed. The KF is used for the basic object tracking, and the PF is utilized for resolving the conflicts. An example of a typical tracking scenario presented here proved that the proposed approach provides higher overall performance scores than both the KF and PF. Therefore, this solution may be recommended for typical tracking scenarios, provided that an evaluation of the tracking algorithms in a specific surveillance setup is performed.

The approach to the object tracking problem presented here is intended for scenarios with a moderate number of moving objects. For busy scenes, other methods, related to a ‘crowd tracking’ (e.g., algorithms based on optical flow) should be used. Object tracking in video is a vast topic and many other scenarios were not presented in this Chapter. For example, there are cameras with a non-fixed field of view, including pan-tilt-zoom systems, moving cameras mounted on UAVs, cameras mounted inside vehicles, etc. In these cases, the background subtraction approach cannot be applied, but the particle filters are very useful. Object tracking in multi-camera setups, including multiple cameras with an overlapping field of view, stereoscopic setups, coupled systems consisting of a fixed and a movable (pan-tilt-zoom) camera, systems consisting of cameras with non-overlapping fields of view, is also a separate topic. The latter case requires an object re-identification procedure, based on the calculated object descriptors (color histograms may be used here,

as described earlier). There are also specific sensors, such as infra-red, thermographic, time of flight, etc. Tracking moving objects in such systems using video analysis methods is possible, but it is a separate problem that deserves its own chapter in a book.

ACKNOWLEDGMENT

This work has been funded by the Artemis JU as part of the COPCAMS project under GA number 332913.

REFERENCES

[Aru02] Arulampalam, A.; Maskell, A.; Gordon, N.; Clapp, T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Processing*. 2002, 50, 174-188.

[Bla06] Blauensteiner, P.; Wildenauer, H.; Hanbury, A.; Kampel, M. On colour spaces for change detection and shadow suppression. *Computer Vision Winter Workshop*. 2006, 87-92.

[Bos07] Bose, B.; Wang, X.; Grimson, G. Multi-class object tracking algorithm that handles fragmentation and grouping. *IEEE Conf. Computer Vision and Pattern Recognition*. 2007, 1-8.

[Bra98] Bradski, G. R. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*. 1998, Q2, 214-219.

[Ceh16] Cehovin, L.; Kristan, M. Visual object tracking performance measures revisited. *IEEE Trans. Image Processing*. 2016, 25, 1261-1274.

[Czy06] Czyz, J.; Ristic, B.; Macq, B. A particle filter for joint detection and tracking of color objects. *Image and Vision Computing*. 2006, 25, 1271-1281.

[Czy08] Czyzewski, A.; Dalka, P. Examining Kalman filters applied to tracking objects in motion. *9th Int. Workshop on Image Analysis for Multimedia Interactive Services*. 2008, 175-178.

[Czy11] Czyzewski, A.; Szwoch, G.; Dalka, P.; Szczuko, P.; Ciarkowski, A.; Ellwart, D.; Merta, T.; Łopatka, K.; Kulasek, L.; Wolski, J. Multi-stage video analysis framework. Video surveillance; Lin, W.; Ed.; InTech: *Rijeka*, 2011, pp. 147-172.

[Dal05] Dalal, N. Histograms of oriented gradients for human detection. 2005 *IEEE Comp. Soc. Conf. Computer Vision and Pattern Recognition*. 2005, 1, 886-893.

[Gus02] Gustafsson, F. Particle filters for positioning, navigation and tracking. *IEEE Trans. Signal Processing*. 2002, 425-437.

[Han03] Hanbury, A. A 3D-polar coordinate colour representation well adapted to image analysis. *Proc. Scandinavian Conf. Image Analysis* (SCIA). 2003, 804–811.

[Hor81] Horn, B.; Schunck, B. Determining optical flow. *Artificial Intelligence.* 1981, 17, 185–203.

[Hor99] Horprasert, T.; Harwood, D.; Davis, L. S. A statistical approach for real-time robust background subtraction and shadow detection. *IEEE ICCV Frame-Rate Workshop*, 1999, 1-19.

[Isa98] Isard, M.; Blake, A. CONDENSATION – Conditional density propagation for visual tracking. *Int. J. Computer Vision.* 1998, 29, 5-28.

[Kim05] Kim, K.; Chalidabhongse, T. H.; Harwood, D; Davis, L. Real-time foreground-background segmentation using Codebook model. *Real-time Imaging.* 2005, 11, 167-256.

[Lab15] Labbe, R. (2015) Kalman and Bayesian filters in Python. 2015. https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python.

[Li03] Li, X. R.; Jilkov, V. P. A survey of maneuvering target tracking Part I: dynamic models. *IEEE Trans. Aerospace and Electronic Systems*. 2003, 39, 1333-1364.

[Lin11] Lin, W. Video surveillance; InTech: *Rijeka;* 2011.

[Luc81] Lucas, B. D.; Kanade, T. An iterative image registration technique with an application to stereo vision. *Proc. Imaging Understanding Workshop.* 1981, 121-130.

[Mar02] Mariano, V.Y.; Min, J.; Park, J.-H.; Kasturi, R.; Mihalcik, D.; Li, H.; Doermann, D.; Drayer, T.. Performance evaluation of object detection algorithms. *Int. Conf. Pattern Recognition.* 2002, 965-969. http://viper-toolkit.sourceforge.net/.

[Nab01] Nabil, A. Smart cameras; Springer-Verlag: New York, NY, 2001.

[Num03] Nummiaro, K.; Koller-Meier, E.; Van Gool, L. An adaptive color-based particle filter. *Image and Vision Computing*. 2003, 21, 99-110.

[Pet01] PETS (2001). Performance Evaluation of Tracking and Surveillance. http://www.cvg.reading.ac.uk/slides/pets.html.

[Ris04] Ristic, B.; Arulampalam, A.; Gordon, N. Beyond the Kalman filter: Particle filters for tracking applications. Artech House, Boston:,MA, 2004.

[Shi94] Shi, J; Tomasi, C. Good features to track. *9th IEEE Conf. Computer Vision and Pattern Recognition.* 1994, 593-600.

[Sta99] Stauffer, C; Grimson, W. E. L. Adaptive background mixture models for real-time tracking. *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR).* 1999, 246-252.

[Suz85] Suzuki, S. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics and Image Processing*. 1985, 30, 32-46.

[Szw10] Szwoch G.; Dalka, P.; Czyzewski, A. A framework for automatic detection of abandoned luggage in airport terminal. *Intelligent Interactive Multimedia Systems and Services*. 2010, 9, 13-22.

[Szw11] Szwoch, G.; Dalka, P.; Czyzewski, A. Resolving conflicts in object tracking for automatic detection of events in video. *Elektronika.* 2011, 52, 52-55.

[Szw15] Szwoch, G. Performance evaluation of parallel background subtraction on GPU platforms. *Elektronika.* 2015, 56, 23-27.

[Szw16] Szwoch G.; Ellwart, D.; Czyzewski, A. Parallel implementation of background subtraction algorithms for real-time video processing on a supercomputer platform. *J. Real-Time Image Processing.* 2016, 11, 111-125.

[Vio01] Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. *Proc. IEEE Comp. Soc. Conf. Computer Vision and Pattern Recognition*. 2001, 1, 511-518.

[Wan00] Wan, E. A.; van der Merwe, R. The unscented Kalman filter for nonlinear estimation. *IEEE Symp. Adaptive Systems for Signal Processing, Communications and Control*. 2000, 153-158.

[Wel04] Welch, G.; Bishop, G. (2004). An introduction to the Kalman filter. Technical report TR-95041, University of North Carolina, 2004. https://www.cs.unc.edu/~welch/kalman/kalmanIntro.html.

[Xu05] Xu, L.-Q.; Landabaso, J. L.; Pardas, M. Shadow removal with blob-based morphological reconstruction for error correction. *IEEE Conf. Acoustics, Speech & Signal Processing*. 2005, 729-732.

[Zha14] Zhang, Q.; Canosa, R. L. A comparison of histogram distance metrics for content-based image retrieval. *Proc. SPIE Imaging and Multimedia Analytics in a Web and Mobile World.* 2014, 9027, 90270O.

[Ziv06] Zivkovic, Z.; Van der Heijden, F. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters.* 2006, 27, 773-780.

BIOGRAPHICAL SKETCH

Grzegorz Szwoch, PhD

Affiliation: Gdansk University of Technology, Department of Multimedia Systems, Gdansk, Poland

Education:

1996 – 2000: Gdansk University of Technology, Ph.D. studies: a Ph.D. degree in Telecommunication.

1991 – 1996: Gdansk University of Technology: a M.Sc. degree in Telecommunication

Address: Gdansk, Poland

Research and Professional Experience:

Grzegorz Szwoch has joined the research staff at the Gdansk University of Technology, Multimedia Systems Department, in 1996. Since then, he has been working in multiple research projects involving audio and video analysis and processing. Since 2008, he has been focusing on projects related to intelligent video content analysis, with the aim of an automated detection of important events, using object detection and tracking techniques.

In 2008-2011, he participated in the project within Polish Platform of Internal Security, related to crime identification and prevention. While working in this project, he has gained experience in object detection, object tracking with Kalman filters, and detection of basic events (such as intrusion) by means of analyzing the collected object tracks.

From 2009 to 2014, he was a researcher in the large European project INDECT – Intelligent information system supporting observation, searching and detection for security of citizens in urban environment. In this project, he continued his work on automated object tracking and event detection, focusing on more complex events, such as detection of unattended luggage or vehicles blocking an intersection. For this purpose, he developed a multilayer detection model, based on the Codebook background subtraction algorithm, adapted to the event detection needs. He also worked on object detection in multi-camera setups, especially in a dual-camera system consisting of a fixed and a pan-tilt-zoom camera.

In another, Polish scientific project MAYDAY Euro 2012 (2010-2012), concerning context analysis of multimedia data streams on a supercomputer platform for identification of specific objects and security threats, his goal was to implement computationally complex video content analysis algorithms on

the supercomputer cluster. The main algorithm that was implemented on the parallel computing platform, was based on the Codebook background subtraction method, supplemented with object tracking with Kalman filters and the object detection module. The algorithm was implemented within a web service.

In the European project ADDPRIV – Automatic Data Relevancy Discrimination for a Privacy - sensitive Video Surveillance (2011-2014), he worked on automatic detection of unattended luggage in public spaces. Within this project, the detection algorithm utilizing a multi-layer model based on the modified Codebook method, was developed. The system was tested in a real-life scenario in Milan-Linate airport.

In the European project COPCAMS – Cognitive and Perceptive Cameras (2013-2016), the research was focused on parallel processing of multimedia streams for application in smart camera systems. During this project, he worked on parallel implementation of the object detection and tracking algorithms on GPU platforms, with CUDA and OpenCL. He developed an object tracking algorithm based on particle filters, intended for implementation in systems equipped with non-fixed cameras, e.g., unmanned aerial vehicles. He also proposed a combined object tracking algorithm, employing both Kalman and particle filters, for an improved resolving of a difficult tracking cases.

His professional interests include audio, image and video processing and analysis, programming (Python, C++) and web technologies. He is particularly interested in employing parallel processing platforms (such as GPUs) and mini-computers (e.g., Raspberry Pi) for the analysis of multimedia data.

He is also an academic teacher on the topics of sound synthesis, computer graphics, audio measurement, and applications of digital signal processor.

Professional Appointments:

since 2004: Gdansk University of Technology, Assistant Professor
2000-2004: Gdansk University of Technology, Research Assistant

Publications Last Three Years:

1. Szczodrak, M., Szwoch, G. An Approach to the Detection of Bank Robbery Acts Employing Thermal Image Analysis. Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA) 2013, Poznan, 2013, 297-301.

2. Kotus, J., Dalka, P., Szczodrak, M,., Szwoch, G., Szczuko, P., Czyzewski, A. Multimodal Surveillance Based Personal Protection System. Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA) 2013, Poznan, 2013, 100-105.
3. Czyzewski, A., Bratoszewski, P., Ciarkowski, A., Cichowski, J., Lisowski, K., Szczodrak, M., Szwoch, G., Krawczyk, H. Massive surveillance data processing with supercomputing cluster. Information Sciences, 296 (1), 2014, 322-344, DOI: 10.1016/j.ins.2014.11.013.
4. Dalka, P., Ellwart, D., Szwoch, G., Lisowski, K., Szczuko, P., Czyzewski, A. Selection of Visual Descriptors for the Purpose of Multi-camera Object Re-identification. In: U. Stanczyk and L.C. Jain (eds.), Feature Selection for Data and Pattern Recognition. Studies in Computational Intelligence, 584, Springer 2014, 263-303, DOI: 10.1007/978-3-662-45620-0_12.
5. Szwoch, G., Dalka, P. Detection of vehicles stopping in restricted zones in video from surveillance cameras. In: A. Dziech, A. Czyzewski (eds.), Multimedia Communications, Services and Security. Communications in Computer and Information Science, 429, Springer 2014, 242-253, DOI: 10.1007/978-3-319-07569-3_20.
6. Lech, M., Dalka, P., Szwoch, G., Czyzewski, A. Examining Quality of Hand Segmentation Based on Gaussian Mixture Models. In: A. Dziech, A. Czyzewski (eds.), Multimedia Communications, Services and Security. Communications in Computer and Information Science, 429, Springer 2014, 111-121, DOI: 10.1007/978-3-319-07569-3_9.
7. Szwoch, G. Parallel background subtraction in video streams using OpenCL on GPU platforms. Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA) 2014, Poznan, 2014, 54-59.
8. Szwoch, G. Performance evaluation of parallel background subtraction on GPU platforms. Elektronika: konstrukcje, technologie, zastosowania, 2015 (4), 2015, 25-29, DOI: 10.15199/13.2015.4.4.
9. Szwoch, G., Ellwart, E., Czyzewski, A. Parallel implementation of background subtraction algorithms for real-time video processing on a supercomputer platform. Journal of Real-Time Image Processing, 11 (1), 2016, 111-125, DOI: 10.1007/s11554-012-0310-5.
10. Szwoch, G. Extraction of stable foreground image regions for unattended luggage detection. Multimedia Tools and Applications, 75 (2), 2016, 761-786, DOI: 10.1007/s11042-014-2324-4.

In: Surveillance Systems
Editor: Roger Simmons
ISBN: 978-1-53610-703-6
© 2017 Nova Science Publishers, Inc.

Chapter 3

PERFORMANCE EVALUATION OF SINGLE OBJECT VISUAL TRACKING: METHODOLOGY, DATASET AND EXPERIMENTS

***Juan C. SanMiguel*[*], *José M. Martínez*[†] and *Mónica Lozano*[‡]**
Universidad Autónoma de Madrid, Madrid, Spain

Abstract

Performance evaluation of visual tracking approaches (trackers) based on ground-truth data allows to determine their strengths and weaknesses. In this paper, we present a methodology for tracker evaluation that quantifies performance against variations of the tracker input (data and configuration). It addresses three aspects: dataset, performance criteria and evaluation measure. A dataset with ground-truth is designed including common tracking problems such as illumination changes, complex movements and occlusions. Four performance criteria are defined: parameter stability, initialization robustness, global accuracy and computational complexity. A new measure is proposed to estimate spatio-temporal tracker accuracy to account for the human errors in the generation of

[*]E-mail address: juancarlos.sanmiguel@uam.es.
[†]E-mail address: josem.martinez@uam.es.
[‡]E-mail address: monica.lozano@uam.es.

ground-truth data. Then, such measure is compared with the related state-of-the-art showing its superiority to evaluate trackers. Finally, the proposed methodology is validated on state-of-the-art trackers demonstrating their utility to identify tracker characteristics.

Keywords: Visual tracking, Performance evaluation, Dataset design, Tracker accuracy and Parameter analysis

1. Introduction

Visual tracking has received enormous attention by the research community during the past years, resulting in a wide variety of approaches (trackers) [55][27]. In this situation, selecting the optimum tracker for each application requires to evaluate tracker performance (i.e., determine its strengths and weaknesses) under different challenges that affect trackers such as noise, clutter, illumination changes and occlusions.

Common performance evaluation of trackers analyzes the obtained results through a methodology defined by a *dataset* (a set of sequences), the *ground-truth* data (manual annotations of the ideal result) and the *measures* (to quantify the performance) [27]. Its design is challenging as it has to cover, with enough variability, the situations and problems of interest [15] and correctly estimate performance. Although there are approaches not based on ground-truth [52][41], most of the literature measure performance as the (spatial and temporal) deviation between the tracker output and the ground-truth data [7][25][45]. However, current approaches partially address this evaluation as they do not systematically cover the tracking problems [6][35][24] or use small datasets [28] (as generating ground-truth is tiresome limiting the dataset variability and size). Moreover, many measures exist making difficult which one to use as there are no comparisons [7], thus increasing the complexity to design a proper methodology. In summary, these limitations have restricted the wide acceptance of a common tracker evaluation approach and motivated the recent interest in major conferences [53][34] and the organization of challenges[1].

In this paper, we propose a methodology for performance evaluation of single-object tracking. It provides an evaluation framework including the dataset, the evaluation measures and the aspects to understand the advantages and drawbacks of the tracker nder evaluation. Each tracker is modeled as a

[1]IEEE Workshop on Visual Object Tracking Challenge, http://www.votchallenge.net/

black box with two inputs (visual data and configuration parameters) and one output (target estimations). Tracking performance is measured against variations of the visual data (problems affecting tracking) and configuration parameters (inaccurate initialization, non-optimum settings) by comparing its results with ground-truth data. A dataset is designed to represent the relevant tracking problems with different complexity levels via synthetic and real data (126 sequences, ˜23000 frames). Then, four evaluation criteria are defined: parameter stability, initialization robustness, global accuracy and computational complexity. A novel spatio-temporal performance measure is proposed to counteract the ground-truth errors made by the annotators. Finally, experiments are presented to validate the proposed methodology[2]. We compare existing accuracy measures, showing the benefits of the proposed measure under inaccurate ground-truth data. Then, we apply the methodology to classical and recently proposed trackers determining their strengths across a wide variety of testing conditions.

The structure of this paper is as follows. Section 2 presents the related work. The proposed methodology is overviewed in Section 3. The dataset and performance criteria are described, respectively, in Sections 4 and 5. Then, Section 6 presents the experimental results. Finally, Section 7 summarizes the main conclusions.

2. Related Work

Performance evaluation for visual tracking can be categorized as low (high) level based on the application's independency (dependency) [27]. Evaluation is also for single (SOTE) and multiple object tracking (MOTE) [9]. MOTE is often simplified to SOTE after associating the estimated and ground-truth targets [7]. In this section, we briefly discuss recent advances in visual tracking and review the SOTE low-level approaches focusing on the performance evaluation scores, the benchmark datasets and evaluation frameworks.

2.1. Visual Tracking

Single-object visual trackers can be roughly categorized into generative and discriminative [58]. *Generative trackers* employ region-based search to find similar areas to the target model such as the Mean-shift (MS) [16], Color Particle

[2]The software and dataset are available at http://www-vpu.eps.uam.es/SOVTds/

Table 1. Popular benchmark datasets for visual tracking. (Keys. P:Person. F:Face. C:Car. CM:Complex Movement. IC:Illumination Change. SC:Scale Change. SO:Similar Objects. OC:Occlusion. LoC:Levels of Complexity. GT:Ground-Truth available.)

Dataset	# Seq	View **distance**	Target **type**	Problems **described**	Identified problems **CM**	**IC**	**SC**	**SO**	**OC**	LoC	GT	Purpose
SPEVI [46] (Sing.)	5	Medium	F	Yes	√	√	√			No	Yes	Indoor tracking
SPEVI [46] (Mult.)	3	Medium	F	Yes		√		√	√	No	Yes	Indoor tracking
ETISEO [29]	86	Far	P, C	No	√	√	√	√	√	No	Yes	Indoor/outdoor surveillance
PETS2000 [36]	1+1	Far	P, C	No		√	√	√		No	No	Outdoor tracking
PETS2001 [35]	5+5	Far	P, C	No	√	√			√	No	No	Outdoor tracking
PETS2006 [35]	28	Far	P	No		√	√	√	√	No	No	Abandoned object
PETS2007 [35]	1+9	Far	P	No	√	√	√	√	√	No	No	Loitering, abandoned object
PETS2010 [37]	1+3	Far	P	No	√	√	√	√	√	No	No	Crowd outdoor activities
CAVIAR [14]	17	Far	P	No		√	√	√	√	No	Yes	People tracking in a mall
VISOR [48]	6	Medium	F	No	√		√	√	√	No	No	Indoor tracking
i-Lids [3]	7	Medium	P, C	No		√	√		√	No	No	Abandoned object/vehicle
Clemson [10]	16	Close	F	Partial	√	√	√	√	√	No	Yes	Indoor tracking
VOTD [39]	16	Medium	P	Yes			√		√	No	Yes	Outdoor tracking
OOT [53]	50	All	P, F,C	Partial	√	√	√	√	√	No	Yes	In/Outdoor tracking
ALOV [45]	300	All	P, F,C	Yes	√	√	√	√	√	No	Yes	In/Outdoor tracking

Filter [32] and Lucas-Kanade [5] trackers. These approaches fail in presence of objects similar to the target or occlusions by other objects. Recent proposals address these limitations via adaptive strategies to update the target model such as incremental PCA [40], continuous outlier detection [61] and scale-orientation adaptations of MS [31][49]. Combination of rigid and deformable generative models can be done via superpixels to increase robustness against occlusions [33]. Local information can be used to increase the target model accuracy such as the MS extension for background correction [30] and the FFT-based tracker [59]. *Discriminative trackers* focus on developing classifiers to distinguish between the target and its background, being sensitive to sudden changes in the surrounding background. For example, the TLD tracker [23] combines PN learning and a tracker to exploit the spatio-temporal structure of the data. Target-background dissimilarity can be also measured via superpixels [54]. Due to the high computational cost of previous approaches, fast discriminative trackers are proposed focused on compressive sensing which updates a set of weak classifiers via sparse factorization [58] and on adaptive dimensionality reduction of color attributes based on their discriminative power [17].

Multiple trackers or models can be combined to overcome the limitations of each tracker such as selecting relevant data to update the target model [21],

imposing smoothness constraints in the combined trajectory [4] and determining spatio-temporal relationships among trackers [20]. Moreover, motion of nearby targets can be exploited to improve performance via structural constraints [60]. However, it cannot be applied to single-target tracking. Finally, the robustness to initialization error has been also recently tackled via visual saliency [56]. The wide variety of existing trackers motivates the development of methodologies to evaluate their performance across challenging visual data.

2.2. Performance Evaluation Scores

Tracker evaluation scores are computed at frame or trajectory level [6]. The former test each frame individually whereas the latter check the similarity between the estimated and ground-truth object trajectories. For an in-depth discussion, the reader may refer to [27][7].

Frame-level evaluation assess the spatial accuracy of the estimated target location. By modeling it as a classification problem, standard measures of precision and recall are applied at pixel [24] or object level [6][11][42]. Both are extended by the *spatial overlap (SO)* between the estimated and the ground-truth data [15][25][28][18][29]:

$$SO(x_f^E, x_f^{GT}) = \frac{2 \cdot \left| A_f^E \cap A_f^{GT} \right|}{\left| A_f^E \right| + \left| A_f^{GT} \right|}, \tag{1}$$

where x_f^E and x_f^{GT} are the estimated and ground-truth targets for frame f, $|A_f^E \cap A_f^{GT}|$ is their spatial overlap (in pixels); $|A_f^E|$ and $|A_f^{GT}|$ represent their area (in pixels). Unlike centroid-based measures [11][43], SO considers the errors in the estimated target size and saturates to one [27]. Evaluating the target estimation can be also considered via its error (i.e., the non-overlapping region) at pixel [15] or region level [13]. Finally, other approaches compute such ground-truth similarity using Euclidean [6] or Mahalanobis [19] distances.

Trajectory-level evaluation quantifies spatio-temporal accuracy of the estimated target tracks. For example, the mean over all the frames of each sequence can be taken for SOs [15][25][29] or centroid distances [11][43]. In addition, [11] computed the positive and false-matches using point-wise ground-truth for measuring the rate of correct, wrong and missing targets. [38] focused on the ability of the tracker to maintain the same identifier for each detected target. Extending the previous approaches, [57] thresholded the SO for determining

correct target detection and derived a set of track-based measures for measuring their fragmentation as well as their spatial and temporal closeness to ground-truth tracks. More recently, [28] defined the loss of target as the number of SOs for an entire track that are below a threshold. Then, this measure is computed for a set of predefined thresholds and accumulated to obtain the performance score of each track.

As a conclusion, several approaches are available to measure tracker performance given a particular tracker initialization and configuration, being not clear which measure to use. Hence, it would be desirable to analyze which measure is more efficient and to systematically study the performance variation under different configurations and initializations (as in [28]).

2.3. Benchmark Datasets

Unlike self-made videos, benchmark datasets provide public sequences to compare trackers [15][53]. Existing datasets are oriented to low or high level evaluation.

Low-level datasets provide ground-truth data on a frame basis to measure tracker accuracy. Although there are datasets with various target types [41][45] or multiple views [22], they usually focus on single views and specific targets such as faces [48] [10] [46], people [37] [39] or cars [36]. They have limitations as the existing problems are not described [48] [10] [37] or not covered with different complexities [41] [28] [48] [10] [46] [37] [39] [53] [45]. The evaluation is often restricted to small size datasets as ground-truth generation is time consuming [41][28] [48] [10].

High-level datasets regard the tracking-application and often present real situations without describing the related problems [35] [29] [14] [3] [47]. Few of them contain ground-truth at frame-level [29] [14]. Unlike low-level datasets, their size is usually large having several tracking problems. The high-level datasets are widely used for many video analysis stages.

Many current datasets for tracking have been collected for high-level purposes without describing the tracking challenges. Table 1 lists the reviewed datasets showing that they do not properly cover the requirements for the test data. Thus, it would be interesting to create a dataset with several tracking problems of varying complexity allowing to compare trackers.

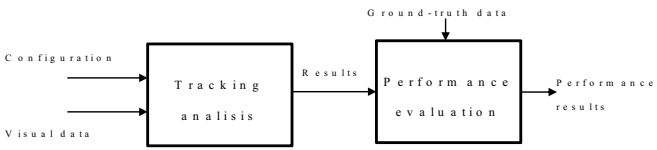


Figure 1. Proposed evaluation methodology for visual tracking.

2.4. Comparison to Other Evaluation Frameworks

Compared to recent evaluation frameworks, the proposed methodology distinguishes a wide variety of challenges unlike [6][52][28][51] that are focused on metrics [6][52], tracker input robustness [28] and comparisons of many trackers [51]. We avoid using many measures to evaluate tracker accuracy (as done in [6][53]) which often derives in poor qualitative and complex analysis of the results. Although the proposed dataset shares some similarities with [53], it is categorized into different complexities allowing a better understanding of the tracker limitations. In [34], the authors focused on analyzing variability of recently published performance rankings instead of the tracker capabilities. Similarly to [53], trackers are evaluated using a large-scale dataset describing tracking-related problems as indicated in [45]. However, we additionally include the complexity of each problem modeled in the dataset and analyze other tracker characteristics(complexity, initialization, parameter variability).

In summary, the previously mentioned frameworks do not handle inaccurate ground-truth data, study the stability of the tracker parameters, measure the impact of target types (cars, people, faces) on performance or compare the tracker complexity. The proposed methodology addresses these shortcomings of tracker evaluation. Moreover, it also evaluates recently proposed approaches, thus extending these frameworks which compare trackers up to 2012.

3. Methodology Overview

The proposed methodology for evaluating single-object trackers is depicted in Fig. 1. It is composed of two stages: tracking analysis and performance evaluation.

The first stage models the tracker to evaluate as a black box with two inputs (visual data and configuration) and one output (results). The *visual data* is the

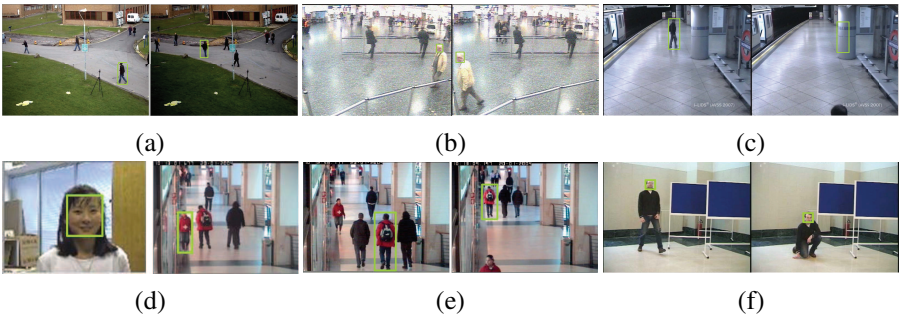


Figure 2. Sample frames of the selected tracking problems on standard datasets: (a) global illumination change [37], (b) image noise [47], (c) occlusion [3], (d) similar objects [10], (e) scale change of red man [14] and complex movement [48]. Targets are represented by green squares.

video sequence with the targets to track. Evaluating tracker accuracy requires to use data covering the tracking problems (e.g., occlusions, scale changes). The *configuration* describes all the tracker parameters that can be manually set (e.g., window search). The *results* are the estimated target locations defined by their bounding boxes (center position and size).

The second stage formalizes the tracker evaluation by comparing their results with ground-truth data. We propose such evaluation as changing the tracker input and then, analyzing its accuracy. Hence, visual data and configuration variations are modeled by, respectively, sequences with complexity-variable tracking problems (requiring to design a new dataset) and the two principal aspects to configure a tracker (initial target location and parameters). Tracker performance is evaluated through four criteria: parameter stability, initialization robustness, global accuracy and computational complexity (detailed in Sec. 5).

4. Dataset Design

We propose a new dataset, named SOVTds (Single-Object Video Tracking dataset), composed of synthetic and real data selected from publicly available benchmarks. It covers common problems and situations of tracking, having a total of 126 sequences (~23000 frames) where ground-truth data is generated for each frame as the target bounding box (center and size). The detailed de-

scription of situations, selected sequences and the annotated ground-truth can be downloaded at http://www-vpu.eps.uam.es/DS/SOVTds/. In this section, we describe the covered tracking problems, the estimation of their complexity and the modeled situations.

4.1. Considered Tracking Problems

Several problems have to be taken into account that corresponds to real-world situations. Fig. 2 shows an example of such problems. In the proposed dataset, we have modeled the following tracking-related problems:

Complex or Fast Movement. The target changes its trajectory unexpectedly or increases its speed abruptly and, thus, it might exceed the tracker search area.

Gradual (and Global) Illumination Changes. Sequences may suffer slow changes on the mean frame intensity, outdating the target model and making tracking hard.

Abrupt (and Local) Illumination Changes. As the target moves, it can enter in areas with variable illumination. Hence, the tracker might be confused losing the target.

Noise. It appears as random variations over the values of the image pixels and can significantly degrade the quality of the extracted features for the target model.

Occlusion. It is defined when an object moves between the camera and the target. It can be partial or total if, respectively, a region or the whole target are not visible.

Scale Changes. It happens when a target moves during the sequence and increases or decreases its size due to changes in its distance from the camera.

Similar Objects. It considers objects with similar features to those of the target (e.g., color, edges) as the tracker might be confused and track them.

Table 2. Criteria and factors to measure the complexity of the tracking problems for each sequence

Problem	**Criteria** (**factors**)
Complex Movement	The target changes its **speed** (pixels/frame) abruptly in consecutive frames
Gradual Illumination	The average intensity of an area changes gradually with time until a **maximum intensity difference** is reached
Abrupt Illumination	The average intensity of an area changes abruptly regarding its neighbor (**maximum intensity difference**)
Noise	It includes natural (snow) or white Gaussian noise which is manually added with **varying deviation valu**e
Occlusion	Objects in the scene occlude **a percentage of the target**
Scale Changes	The target changes its size with a **maximum relative change** regarding its original size.
Similar Objects	An object with **similar average color to the target** appears in the neighborhood of the target

4.2. Complexity Factors

After describing the problems covered by the dataset, we define the criteria to evaluate their complexity (Table 2). These criteria include objective (illumination change, occlusion and scale change) and subjective (complex movement, noise and similar objects) factors. Some factors can be artificially generated (noise and illumination changes) allowing to create synthetic sequences or modify real ones with any required complexity.

4.3. Modeled Situations

Trackers operate in several situations ranging from controlled (e.g., synthetic data) to uncontrolled (e.g., real-world data). We estimate the complexity of the dataset sequences using the criteria in Table 2 and categorize them into the following four situations.

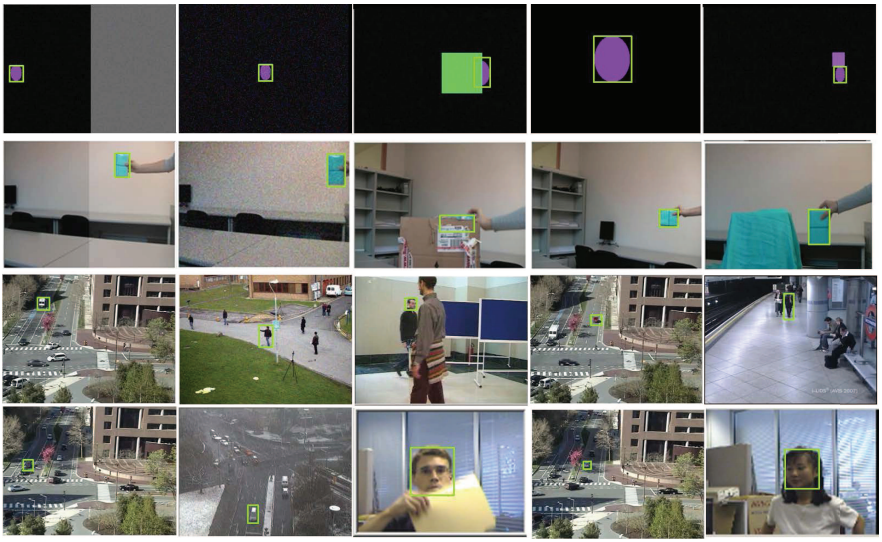


Figure 3. Sample frames for the situations of the proposed dataset (from top row to bottom row): synthetic (S1), laboratory (S2), Simple real (S3) and Complex real (S4). In addition, samples of some tracking-related problems are also presented for each column (from left to right): abrupt illumination change, noise, occlusion, scale change and (color-based) similar objects. Target are represented by green squares.

4.3.1. Synthetic Sequences (S1)

It is composed of synthetic sequences that provide controlled testing conditions allowing to isolate each problem. They consist on a moving ellipse in a black background that can contain squares of the same or different color (acting as, respectively, similar or occluder objects). We have created sequences for all the selected problems with five complexity degrees each one. In total, 35 sequences are generated (~3500 frames). Sample frames are shown in the first row of Fig. 3.

4.3.2. Laboratory Sequences (S2)

It extends S1 by representing real data in a laboratory setup under controlled conditions. Video sequences are created for all selected problems (with three complexity levels each one) using a single-color object as target. For some

problems (noise, gradual and abrupt illumination changes), the problems are artificially introduced. S2 contains 21 sequences (˜6500 frames). Sample frames are shown in the second row of Fig. 3.

4.3.3. Simple Real Sequences (S3)

It includes real data from public datasets under non-controlled situations. We have extracted clips from the original data containing isolated tracking problems. As each target has different properties [27], we have grouped the sequences into three target-dependent categories: cars (from MIT Traffic [2] and Karlsruhe [1] datasets), faces (from TRECVID2009 [47], CLEMSON [10] and VISOR [48] datasets) and people (from PETS2000 [35], CAVIAR [14], TRECVID2009 [47], PETS2009 [37] and i-Lids [3] datasets). For each target type and problem, three sequences with different complexities are identified making a total of 53 sequences (˜8500 frames). Sample frames are shown in the third row of Fig. 3.

4.3.4. Complex Real Sequences (S4)

It contains the most complex data, which are clips from other datasets mixing problems. It covers real situations where it is hard to find isolated problems. Similarly to S3, we distinguish three types of targets. For each sequence, the existing problems have been estimated and classified according to Table 2. The data are extracted from the MIT Traffic [2] (for cars), CLEMSON [10] (for faces) and PETS2009 [37] (for people) datasets. S4 has 15 sequences (˜4500 frames). Sample frames are shown in the fourth row of Fig. 3.

5. Performance Evaluation

We describe the proposed evaluation measure and the four criteria to assess tracker performance.

5.1. Performance Measure

Many evaluation approaches rely on *Sequence Frame Detection Accuracy* (SFDA) [25] that performs a temporal averaging of frame-based SOs (eq. 1) for each sequence. SO is limited as it computes pixel errors between the estimated and ground-truth data disregarding their location (i.e., errors around the center

or borders of the target are equally penalized). However, ground-truth annotations are manually generated and prone to human error, which is often noticed at the borders of the target location [26]. We extend SFDA by weighting the overlapped pixels (between the estimated and ground-truth data) according to their spatial position. A new measure, *Accumulated Weighted Spatial Overlap* (AWSO), is defined for each sequence as:

$$AWSO = \frac{1}{F_{GT}} \sum_{f=1}^{F_{GT}} WSO(f), \tag{2}$$

where WSO is the *Weighted Spatial Overlap*:

$$WSO(f) = \frac{\sum_{i=1}^{N_O} k_E\left(o_f^i\right) + \sum_{i=1}^{N_O} k_{GT}\left(o_f^i\right)}{\sum_{i=1}^{N_E} k_E\left(l_f^i\right) + \sum_{i=1}^{N_{GT}} k_{GT}\left(g_f^i\right)}, \tag{3}$$

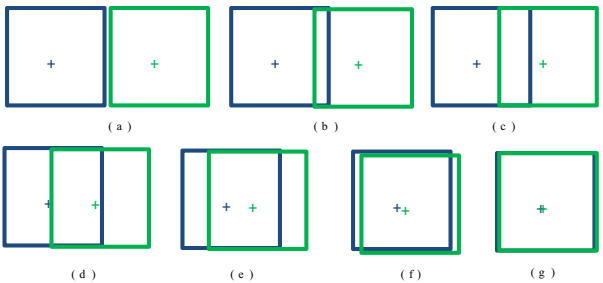
where F_{GT} is the number of frames with ground-truth data; o_f^i, l_f^i and g_f^i are pixel coordinates of, respectively, the overlapped, the estimated and ground-truth locations for frame f; N_O, N_E and N_{GT} are the number of overlapped, estimated and ground-truth pixels; $k_E(\cdot)$ and $k_{GT}(\cdot)$ are two kernels to weight each pixel inversely proportional to its distance from, respectively, the estimated (l_f^c) or ground-truth (g_f^c) center locations. Both are defined as:

$$k_E\left(o_f^i\right) = 1 - \left(d\left(o_f^i, l_f^c\right)/d_{max}\left(o_f^i, l_f^c, l_f^{1\ldots N_E}\right)\right)^n, \tag{4}$$

$$k_{GT}\left(o_f^i\right) = 1 - \left(d\left(o_f^i, g_f^c\right)/d_{max}\left(o_f^i, g_f^c, g_f^{1\ldots N_{GT}}\right)\right)^n, \tag{5}$$

where $d(\cdot,\cdot)$ is the Euclidean distance between each pixel coordinate (o_f^i, l_f^i or g_f^i) and the center of the estimated (l_f^c) or ground-truth (g_f^c) target; $d_{max}(\cdot,\cdot,\cdot)$ gets the maximum distance determined by the furthest ground-truth point g_f^i along the line formed by o_f^i and g_f^c (similar for target estimation using l_f^c and l_f^i instead of g_f^c and g_f^i); n controls the importance given to pixels close to the center location.

As a summary, we measure performance for each frame by combining the weighted coverage of the spatial overlap for both the estimation and the ground-truth location. Values close to one (zero) indicate high (low) tracker spatial accuracy. Fig. 4 shows an example of SO and WSO measures.



	(a)	(b)	(c)	(d)	(e)	(f)	(g)
Overlap	0%	10%	25%	50%	75%	90%	100%
SO	.00	.10	.25	.50	.75	.90	1
WSO	.00	.04	.18	.50	.81	.96	1

Figure 4. Sample results for standard SO and the proposed WSO measures for different spatial overlaps. Estimations and ground-truth targets are depicted by, respectively, blue and green squares.

5.2. Performance Criteria

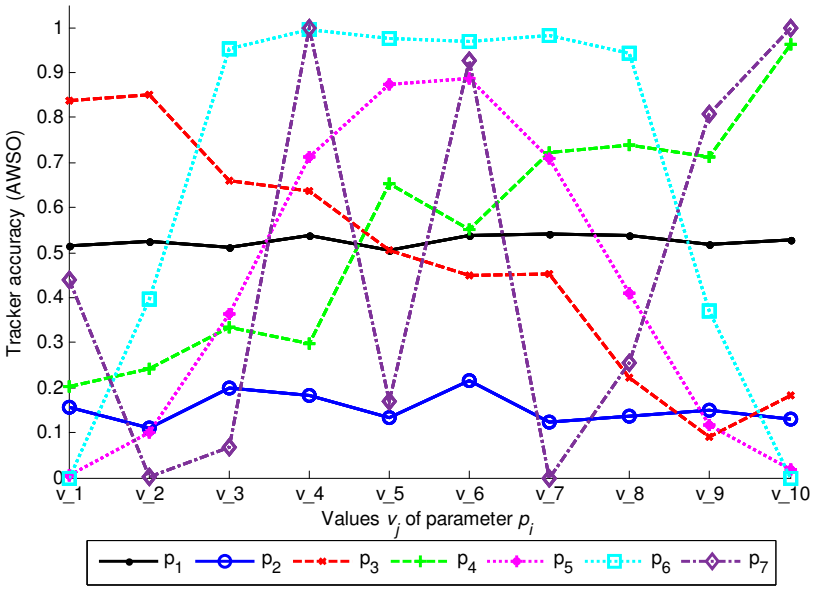
5.2.1. Parameter Stability

Trackers frequently have many parameters, making difficult their optimal configuration. We study the parameter impact on tracker accuracy to determine which ones are more reliable or require fine tuning. First, the relevant parameters (p) are identified as well as their testing values. Then, a default setting is defined and, for each p, the sequences are processed varying their values. Parameter sensitivity is computed as the tracker accuracy variance:

$$\sigma_p = \frac{1}{N_s}\sum_{s=1}^{N_s}\sqrt{\frac{1}{N_p}\sum_{v=1}^{N_p}(AWSO_{s,v} - \mu_{s,p})^2}, \tag{6}$$

where N_s and N_p are, respectively, the number of sequences and test values of p; and $AWSO_{s,v}$ and $\mu_{s,p}$ are, respectively, the results accuracy of sth sequence for the value v of p and the mean for all v (computed as eq. 2).

Detecting stable parameters requires defining the stability concept, which often depends on the application. As a first approach, we threshold σ_p using a maximum allowed deviation (σ_{max}) to accept stability ($\sigma_p \leq \sigma_{max}$). However, the opposite condition ($\sigma_p > \sigma_{max}$) does not imply instability as results may be



	p_1	p_2	p_3	p_4	p_5	p_6	p_7
σ_p	.012	.035	.266	.258	.354	.422	.425
η_p	.010	.031	-.668	.765	.010	.001	.566
γ_p	1	.556	.333	.444	.111	.556	.000

Figure 5. Sample results for parameter stability. The curves represent the result's variation of seven tracker parameters (all with 10 test values) for one sequence. For $\sigma_{max} = 0.05$, only p_1 and p_2 are stable. A predominant decreasing and increasing trend is observed for, respectively, p_3 and p_4 (high $|\eta_p|$). p_5 and p_6 have partial stability (high γ_p). The most unstable is p_7 (low γ_p and high σ_p).

partially stable (see Fig. 5). To detect it, we measure properties of $AWSO_{s,v}$ results using the mean accumulated difference ($\eta_p \in [-1, 1]$) and the ratio of consecutive stable values ($\gamma_p \in [0, 1]$):

$$\eta_p = \frac{1}{N_s} \sum_{s=1}^{N_s} \left(\sum_{v=2}^{N_p} (AWSO_{s,v} - AWSO_{s,v-1}) \right), \tag{7}$$

$$\gamma_p = \frac{1}{N_s}\sum_{s=1}^{N_s}\left(\frac{1}{N_p}\sum_{v=2}^{N_p}(\Delta AWSO < \sigma_{max})\right), \tag{8}$$

where $\Delta AWSO = AWSO_{s,v} - AWSO_{s,v-1}$.

High (low) γ_p values mean stable (unstable) ranges. For η_p, stability is reached for values close to 0. Both measures are required for stability as only using a single one is not sufficient. Fig. 5 shows examples for seven parameters. Note that thresholding η_p and γ_p is not straightforward and, therefore, the analysis is limited to qualitative comparisons (one parameter is less sensitive than others). Finally, optimum parameter value is obtained as the most frequent maximum-performance value over all sequences.

5.2.2. Robustness to Target Initialization

Trackers under evaluation are usually initialized with ground-truth data. However, tracking-based applications may use automatic initialization whose performance is lower than the ground-truth one [27]. The impact of inaccurate initialization on the results should be analyzed. Similarly to [28], we modify the ground-truth (initial) target location in three aspects: size (preserving ground-truth center), center (keeping the ground-truth size) and both. These initializations are applied to the tracker whose accuracy is measured using AWSO. Fig. 6 depicts some examples.

Differently from [28], we distinguish three target types (persons, faces and cars) and three overlap factors between ground-truth and modified initializations (90%, 75% and 50%). It allows to measure performance decrease for variable inaccuracy (overlap) and target types (rigid vs non-rigid). For each target, we generate 10 random initializations for overlap and modification (making a total of 90).

5.2.3. Global Accuracy

For each problem described in the dataset, the tracker global accuracy is computed as:

$$gAWSO = \frac{1}{F_T}\sum_{s=1}^{N_s} F_{GT}^{s} \cdot AWSO_{s,v^*}, \tag{9}$$

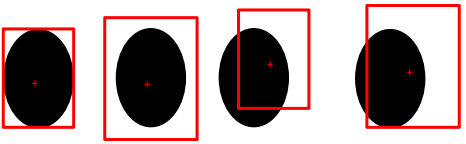


Figure 6. Target initialization modifications: (a) the ground-truth initialization, (b) changes in size (center is the ground-truth one), (c) change in position (size is the ground-truth one) and (d) changes in size and position. Target location and initialization are, respectively, solid black ellipses and red squares.

where s indicates the sth sequence and v^* the optimum values of tracker parameters as computed in sec. 5.2.1; $AWSO_{s,v*}$ is its accuracy value computed as eq. 2; and F^s_{GT} and F_T are the number of frames of, respectively, each sequence and problem (with N_s test sequences).

5.2.4. Computational Complexity

Visual tracking, where real-time constraints may apply [27], often involves intensive processing. Hence, the computational complexity of tracking has to be estimated. It can be theoretically determined via the big-O notation [44]. However, current trackers contain many stages limiting the use of such notation. In practice, this complexity can be approached as the mean time (or memory) required for tracking. Note that both measures depend on the implementation and the testing machine but they are accepted to approximate algorithmic complexity [8]. We extend this complexity analysis by calculating such time as a function of the target size.

6. Experiments

We present two experiments for the proposed approach. First, we evaluate the proposed measure *AWSO* (Sec. 5.1) and related ones for tracker accuracy. Then, we apply the proposed evaluation methodology (Sec. 5.2) to selected trackers using the SOVTds dataset (Sec. 4). A standard PC (P-IV 2.8GHz and 2 GB RAM) is used.

Table 3. Evaluation results for different tracking errors. Results are for CBWH (low), MS (med) and SOAMST (high) trackers using selected sequences from Karlsruhe [1], PETS2010[37], CLEMSON [10], I-LIDS[3] and VISOR [48] datasets.

Error case	Test sequence	Mean evaluation result					
		SFDA	ATA	iATE	iAUC	TC	AWSO
Low	S3_cars_nM	.741	.741	.840	.758	1	.867
Low	S3_cars_nL	.753	.753	.849	.779	1	.825
Low	S3_faces_nM	.797	.797	.797	.791	1	.894
Low	S3_faces_nL	.804	.804	.804	.799	1	.899
Low	S3_people_nH	.829	.829	.829	.824	1	.909
Low	S3_people_nM	.837	.837	.817	.831	1	.920
Low	S3_people_nL	.841	.841	.831	.836	1	.925
Med	seq_bb	.429	.592	.466	.426	.608	.488
Med	seq_jd	.353	.638	.384	.351	.545	.325
Med	seq_mb	.642	.694	.558	.637	.922	.679
Med	seq_ms	.510	.685	.510	.507	.758	.594
Med	seq_sb	.427	.675	.363	.424	.631	.453
Med	seq_villains2	.553	.567	.535	.549	.920	.617
High	AB_Easy_man	.275	.376	.534	.272	.615	.385
High	mv2_002_redcar	.225	.394	.557	.223	.542	.300
High	mv2_005_scar	.205	.426	.448	.203	.451	.105
High	visor2_head	.120	.628	.132	.119	.167	.148
High	visor5_head	.035	.572	.039	.035	.062	.032
High	visor6_head	.084	.392	.181	.083	.172	.130

6.1. Comparison of Performance Evaluation Measures

We compare the proposed *AWSO* measure with relevant approaches based on temporal mean of SOs (Sequence Frame Detection Accuracy, *SFDA* [25]), SOs above zero (Average Tracking Accuracy, *ATA* [15]), wrong target pixels (Average Tracking Error, *iATE=1-ATE*) [15], inverse target loss ratio (*iAUC=1-AUC* [28]) and correct target detections as *SO¿0.30* (Track Completeness, TC [57]). All range from zero (low accuracy) to one (high accuracy). We analyze three error degrees (low, medium and high), which are obtained after a visual inspection of the following tracker's results: MeanShift (MS) [16], Background-corrected MS (CBWH) [30] and Scale Adaptive MS (SOAMST) [31].

Table 3 summarizes the obtained results. As it can be observed, SFDA presents a strong correlation with iAUC for all cases indicating that both are similar. ATA wrongly performs for medium and low error cases having high values as it requires a spatial overlap greater than zero for its computation. TC obtains disperse values as can be observed for medium (sequences *seq_bb* and *seq_ms*) and low (sequences *S3_faces_nM* and *S3_people_nM*) errors. Moreover, TC easily saturates to 1 for the low error case. An example of the TC inconsis-

tency is depicted in Fig. 7(a) where both results have similar SFDA. The results of SFDA and AWSO show that both measures provide reliable estimation for all error cases. As shown in Fig. 7(b), low SFDA values correspond to correctly tracked targets as most of the estimated locations are very close to the target center. Hence, AWSO provides a better evaluation.

As a conclusion, SFDA correctly represents tracker performance for all cases. Although the theoretical SFDA range is [0, 1], according to the results, the real range is almost [0, 0.8], thus reducing its variability. iAUC is very similar to SFDA. ATA and TC are not consistent for, respectively, high and all error cases. The proposed AWSO addressed the annotation errors allowing a fully range coverage, thus improving SFDA.

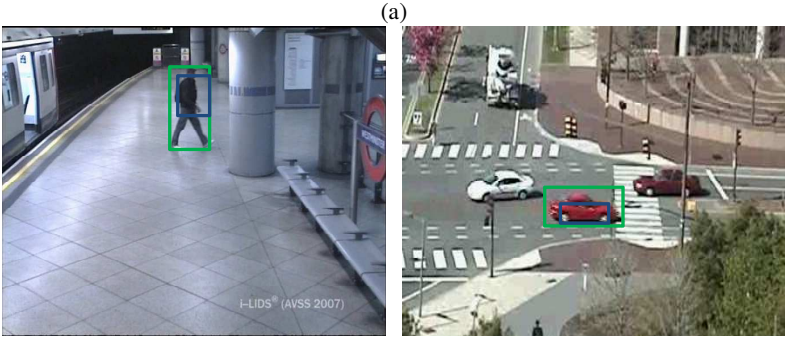
6.2. Application of the Evaluation Methodology

We evaluate a total of 14 trackers, half classic and half recently proposed trackers. The selected classical trackers are: MeanShift (MS) [16], standard Template Matching (TM) [12], TM based on affine transformations (LK) [5], Color-based Particle Filter (PFC) [32] and Incremental Visual Tracker (IVT) [40]. The selected trackers which have been recently proposed are: Background-corrected MS (CBWH) [30], Scale Adaptive MS (SOAMST) [31], Track-Learn-Detect (TLD) [23], sparse multi-hypothesis tracker (ST) [50], Adaptive Color Attributes (ACA) [17], Compressive Tracker (CT) [58], Pyramid-based Sparse Representation Mean Transform (PSRMT) [61], Spatio-Temporal Context learning (STC) [59] and Probability Continuous Outlier Model (PCOM) [49]. For all trackers, we use the implementation and default settings provided by the respective authors.

6.2.1. Parameter Stability

We select the most relevant parameters according to author's tracker description (we use default values for the non-selected ones). For MeanShift-based trackers (MS, CBWH and SOAMST), we analyze search area size (as a factor of target size, W_{SA}) and the maximum optimization iterations (*maxiter*). For template-based trackers, we study the search area for TM (W_{SA}) and the number of initial and translation iterations of LK for, respectively, coarse and fine location (*SigmaIter* and *TransIter*). For the probabilistic tracker (PFC), we examine N (number of particles), σ_{pos} (center prediction) and σ_{size} (size

(a)



(b)

Figure 7. Comparison of tracker evaluation measures. (a) Results for *Seq_mb [10]* (first row) and *l3_people_nH [37]* (second row) sequences that have the same TC and different SFDA. (b) Results for *AB_Easy_Man* and *mv2_002_redcar* sequences where SFDA and AWSO indicate, respectively, low and medium accuracy. Estimations and ground-truth targets are, respectively, blue and green squares.

prediction). For detection-based trackers (IVT and TLD), we inspect the number of samples taken (*numsample*), the template size (*tmplsize*), the number of previous frames buffered (*batchsize*), the modeling complexity (*numtrees*) and the number of detections evaluated in each iteration (*maxbbox*). We use seven representative sequences of the dataset for stability analysis.

Table 4 presents the results of the proposed stability measures and we analyze them considering $\sigma_{max} = 0.05$ (i.e., tolerating a maximum 5% of deviation). For MeanShift-based trackers (MS, CBWH and SOAMST), *maxiter* is stable and W_{SA} shows a noticeable decreasing trend (high σ_p and negative η_p).

Table 4. Stability analysis for the selected parameters of the evaluated trackers (mean values over all selected sequences). Results are presented using AWSO measure (Eq. 2).

Tracker	Parameter	#test values	Stability measures			Optim. value
			σ_p	η_p	γ_p	
MS	W_{SA}	8	.189	-.362	.381	.25
	maxIter	6	.002	-.002	1	5
CBWH	W_{SA}	8	.204	-.513	.476	.1
	maxIter	6	.003	-.001	1	5
SOAMST	W_{SA}	8	.167	-.224	.333	.25
	maxIter	6	.001	.000	1	5
TM	W_{SA}	8	.171	.114	.667	.5
LK	*SigmaIter*	6	.012	-.028	.958	1
	TransIter	6	.014	.012	.933	5
PFC	*N*	5	.011	-.015	1	300
	σ_{pos}	5	.063	.060	.625	250
	σ_{size}	5	.024	-.021	.800	.5
IVT	*numsample*	5	.057	-.086	.500	100
	tmplsize	5	.283	-.021	.167	32
	batchsize	5	.050	.076	.583	10
	maxIter	5	.000	.000	1	5
TLD	*maxbbox*	5	.167	.055	.417	.75
	num_trees	5	.157	.031	.208	5
	tmplsize	5	.168	.029	.250	32
ST	N	5	.015	-.021	.880	500
	σ_{pos}	5	.073	.065	.825	5
CT	M	5	.051	.065	.583	90
STC	lambda	5	.050	.005	.350	0.05
PCOM	*numsample*	5	.008	.076	.726	200

Low values of W_{SA} are preferred for better performance. Identical behavior has W_{SA} in TM. Both LK parameters (*SigmaIter* and *TransIter*) demonstrated high invariance to tracker results. Thus, low values are selected to reduce complexity. For PFC, σ_{pos} requires more tuning effort as it has high σ_p. However, it has a clear stable range as $\gamma_p = .625$ and no relevant increasing/decreasing trend $\eta_p = .060$. The most invariant parameter is the number of particles (*N*) as all the test values provide similar performance. For IVT, a noteworthy reliance on *tmplsize* is observed where best performance is clearly obtained for few values ($\gamma_p = .167$) with an almost peaked pattern (no slope as $\eta_p = -.021$ and high

variance as $\sigma_p = .283$) independently of target size. The rest of parameters have acceptable performance ($\sigma_p \sim \sigma_{max}$), being constant almost half of test values ($\gamma_p \sim .500$). TLD parameters are all unstable indicating the difficult optimum tuning of TLD tracker. Comments for *tmplsize* and *numtrees* are similar to those of *tmplsize* for IVT. For *maxbbox*, a higher range of stability is observed without predefined trend. Hence, intermediate values (of the test ones) are preferred for *maxbbox*. For the rest of the trackers, the optimal values of the evaluated parameters are shown in Table 4.

6.2.2. Robustness to Target Initialization

We measure the effect of inaccurate initialization for three target types: cars, faces and people. For each target, we operate as indicated in Sec. 5.2.2 using sequences with medium complexity (*mv2_006_redtruck* and *mv2_020_silver* for cars, *seq_jd* and *seq_sb* for faces, *l3_people_illumlocal_m* and *PETS09_S2L2_view001_1* for people). The results are depicted in Fig. 8:

Cars (First Row of Fig. 8). For high overlap (90%), results are very similar to ground-truth initialization (100% case) indicating that the relevant target data is still included. Among the trackers, LK has low performance with a decreasing trend. PFC shows high robustness to size changes (wh) due to its ability to change the estimated target scale (from inaccurate to the true one). For IVT and TLD, their update schemes for target models are rapidly degraded if non-accurate target samples are included (low overlaps) and background data contains features similar to those of the target. CBWH, CT, ACA and PCOM are the best approaches being capable to deal non-accurate initialization better than the other trackers since they employ sparse representations and adapt to color changes.

Faces (Second Row of Fig. 8). Face targets had a lower general complexity which is shown in the results. In general, face targets allow easy annotation using a bounding box format. Hence, the values are higher than the *Cars* case in all the categories. In fact, some trackers (e.g., CBWH, MS, PCOM) get better results for slight size changes (90% case). This can be explained as the ground-truth annotation was not completely accurate having errors at the borders. Besides, PFC and SOAMST also show high robustness to size and position changes. Unlike for *Cars*, sequential update of target model of IVT

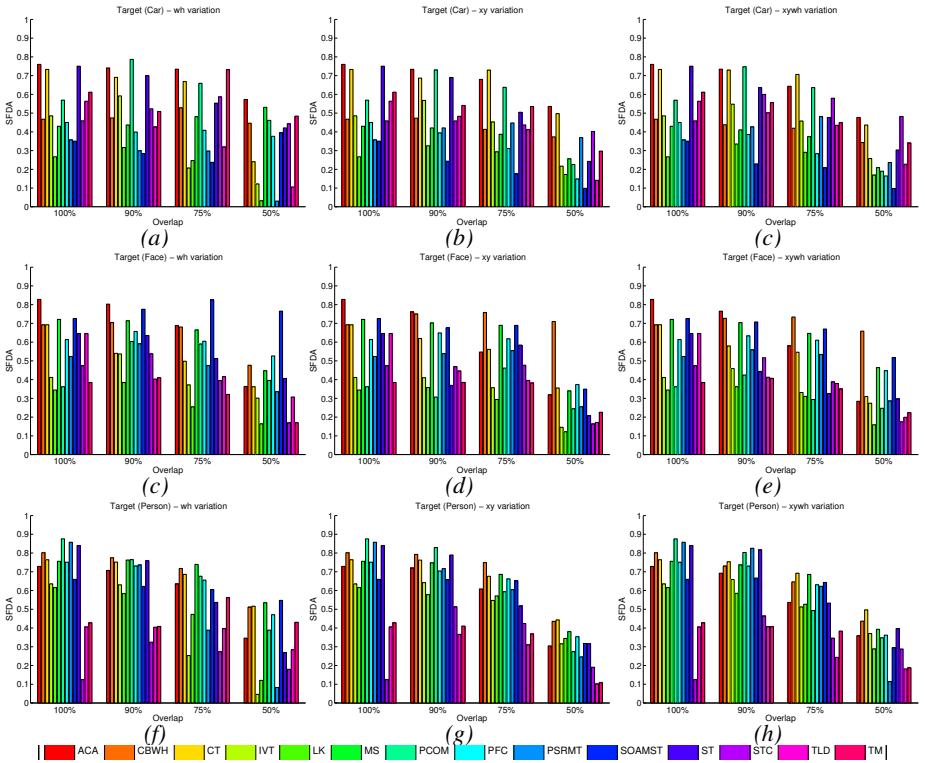


Figure 8. Target initialization performance of the selected trackers for Car (first row), Face (second row) and Person (third row) targets using SFDA [25]. First, second and third columns correspond to changes in, respectively, size (wh), position (xy) and both (xywh). For each change, three spatial overlaps with ground-truth data are considered: 90%, 75% and 50%. 100% case is the ground-truth initialization.

does not have a significant impact in performance. However, TLD still presents degradation due to the intensive use of background data for such update. Trackers with low scores (LK, STC and TM) improve their results as a reduction of the initialization size allowed to reduce the amount of background information in the computed target model. Finally, CT shows its robustness in the 90-75% cases.

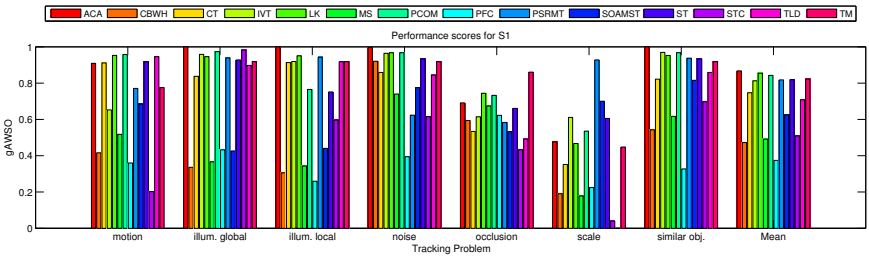


Figure 9. Performance evaluation of selected trackers for each problem of the S1 situation (synthetic sequences).

People (Third Row of Fig. 8). It can be noticed that as the overlap is reduced, the performance is degraded at a higher rate than the previous cases. Moreover, SOAMST, ST and PFC demonstrate high robustness since the performance drop is not severe for size and position target changes. CBWH and PCOM also get similar results (for 90% and 75% cases) to ground-truth initialization indicating possible errors in the borders of the ground-truth annotations. Similarly to the Faces case, the accuracy decrease is more difficult to observe in low performance trackers such as TM and LK. In fact, TM shows an improvement when highly reducing the initialization size. STC shows that the spatial context of people targets can dramatically change depending on the initialization. IVT, TLD and the rest of the trackers have similar conclusions to previous results. Among the trackers, CBWH, PCOM and PSRMT obtain the best results for size and position variations.

6.2.3. Global Accuracy

After analyzing parameter variation and target initialization, we present the results of the selected trackers for the modeled situations in the dataset. We use the same settings for all sequences.

S1: Synthetic Sequences (Fig. 9). Best results are provided by many trackers (TM, ACA, PCOM, ST and LK as the background of all sequences has an uniform color different from the target one. LK has high performance in most of the tracking problems except for scale changes as it is not able to adapt to drastic size changes. CBWH also presents good performance for the noise and com-

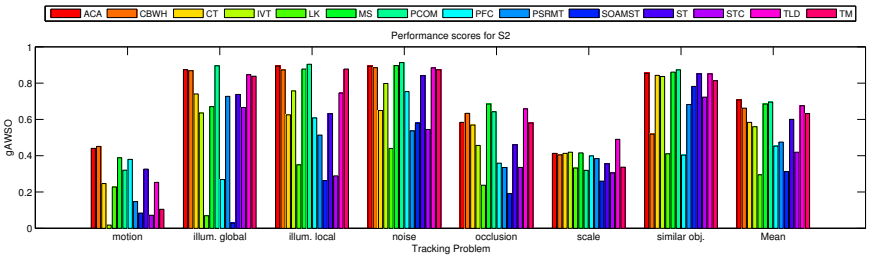


Figure 10. Performance scores of selected trackers for each problem of the S2 situation (lab sequences).

plex movement problems. However, illumination changes dramatically affect CBWH as it does not adapt to target appearance changes. MS obtains comparable performance to that of CBWH. SOAMST gets high results showing its good adaptation to variable target size. Surprisingly, PFC does not perform as expected as it focuses on tracking patches of the target (with features similar to the entire target). Thus, although PFC successfully tracks the target center, the estimated size is not correct and low performance is obtained. IVT is able to adapt the estimation to changes in illumination, noise and scale. However, target model update is corrupted in presence of occlusions and sudden motion. TLD is only affected by occlusions being successful in the rest of the problems. Scale changes are perfectly handled by PSRMT followed by SOAMST. ACA shows the power of adaptability of color attributes under simple backgrounds. Abrupt motion is faced by ACA, CT, LK, PCOM, ST and TLD. Finally, it can be observed that all trackers exhibited difficulties for occlusions.

S2: Lab Sequences (Fig. 10). Six trackers have the best performance (see Mean bars) in most of the problems: ACA, CBWH, PCOM, MS, TLD and TM. A high robustness to noise is observed in most of the trackers whereas they struggle in presence of abrupt motion, scale changes and occlusions. For TM, its non-adaptivity to scale changes is observed by its low results as compared to other approaches. LK obtains low performance as most of the problems made to lost the target in the beginning of the sequence without finding it again, therefore, delivering poor results. In particular, its performance decrease for occlusions is relevant compared with the other trackers. CBWH successfully performs in most the cases except for scale changes and similar objects

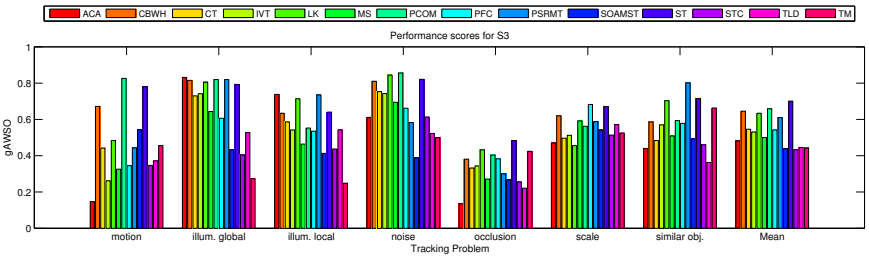


Figure 11. Performance scores of selected trackers for each problem of the S3 situation (Simple real sequences).

because of, respectively, it tracks constant size targets and it does not compute an accurate target model in presence of similar objects. MS slightly outperforms CBWH showing that discriminating the features in the target neighborhood does not always improve results (specially, if the sequence contains complex backgrounds). PFC gets medium results in all the problems except for scale changes demonstrating its adaptation to size changes. However, overall results are worst than other trackers. SOAMST is similar to PFC showing a significant performance decrease for complex movement and global illumination changes. IVT and TLD have limitations to track under sudden target motions. Both exhibit performance among the best trackers for the other problems. For similar objects, discriminative trackers (PSRMT, ST, STC and PCOM) obtain good results as they consider the nearby background which is slightly different for nearby objects. In summary, ACA provides a good compromise for each tracking problem, being always among the top trackers.

S3: Simple Real Sequences (Fig. 11). Robustness to noise and global illumination changes is achieved by CBWH, LK, PCOM, ACA, PSRMT, ST and IVT. It should be noted that all trackers failed for occlusions. CBWH obtains good results closely followed by LK. Unlike for S1 and S2, PFC results are comparable to the best results as real targets always undergo small changes in size and appearance that PFC is capable to deal with. For this reason, TM results drop as compared to previous situations. The presence of objects similar to the target is frequent in real data and, therefore, TM is easily distracted. However, LK is able to adapt the template to target changes considering the neighbor of the target. SOAMST adaptation is heavily affected by target-like

objects in the background as it only looks for similar features disregarding the target size. It is not able to correctly track scale changes with real data and other similar approaches without size adaptability (CBWH and MS) get better results. MS has average performance with a great robustness for global illumination changes. IVT has similar conclusions as for S2: it depends on complex motion and occlusions, affecting the accuracy of the target model update which leads to drifting. TLD shares IVT drawbacks whilst having strong performance dependency for similar objects close to the target. Sparse target modeling of ST is adequate when facing real data problems. Unlike the previous situations, ACA shows that color adaptability is present additional challenges compared to controlled environments (S2 and S3). PCOM is again among the best trackers as the modeling of noise is appropriate for real data. STC shows that contextual target information is difficult to extract as many clutter often exists around the target. Again, sparse models (ST and PSRMT) present robustness against occlusions and similar objects.

S4: Complex Real Sequences (Fig. 12). S4 data have the highest complexity as sequences mix various tracking problems. Thus, we analyze the target types instead of each problem. Results for Cars present best performance as these targets allow an easy annotation and modeling. Face targets have similar characteristics but they might move quickly (as camera distance is usually closer than that for Cars). Hence, the model update scheme is affected by wrong target estimations, explaining the performance drop of ACA, CT, STC, IVT, LK and SOAMST. For People targets, a decrease in performance is clearly observed showing their difficulty to model and track. Among the trackers, CBWH is the best closely followed by ACA as color cues are adapted and CT/PFC as removing background data from the Person model improves accuracy. In second order, MS and ST present slightly lower performance being limited to track People targets. Finally, IVT, SOAMST, LK, TLD and TM reduce their accuracy when dealing with complex real data (as compared with the other situations). As expected the performance for S4 is the lowest compared to the other situations.

6.2.4. Computational Complexity

To analyze tracker complexity, we measure the execution time of the selected trackers using the implementations of the respective authors (all in MATLAB).

For the mean execution time (in milliseconds per pixel), the quickest track-

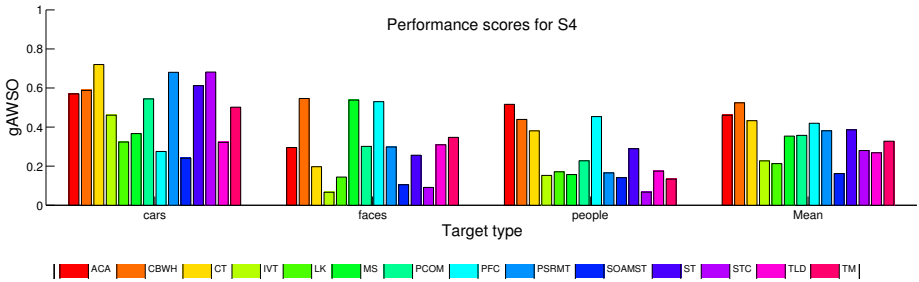


Figure 12. Performance scores of selected trackers for each target type of the S4 situation (Complex real sequences).

ers are ACA, STC, CT, TM, CBWH and MS (with respectively 0.013, 0.009, 0.015, 0.011, 0.025 and 0.033) due to their simple computations. Then, a second category comprises trackers with medium complexity such as LK (0.309) and SOAMST (0.362). Finally, advanced trackers are the slowest ones: TLD (0.578), IVT (0.789), PFC (1.155), PCOM (0.49), PSRMT (0.57) and ST (0.96). Although these results depend on implementations that may not be optimal, they allow a rough speed-based categorization.

Fig. 13 depicts the execution time versus the target area that can be understood as a measure of complexity scalability. As expected, most of the trackers require more time for increasing target sizes. On the contrary, IVT and TLD show a different trend. Both trackers use a predefined number of fixed-size patches extracted from the target, allowing them to be almost size-independent. This advantage could be useful when dealing with high quality data. However, there is an additional cost for small targets having higher execution time than the other trackers.

6.2.5. Discussion

Here we discuss the major findings after analyzing the selected trackers with the proposed methodology.

For parameter stability, the search area is a sensible parameter in many trackers. In real settings, it should be close to target area to avoid including similar objects to the target in the analysis. However, robustness against size changes and sudden motion requires higher search areas. As a result, the tuning of this parameter exhibits a trade-off between adaptability (to size and motion)

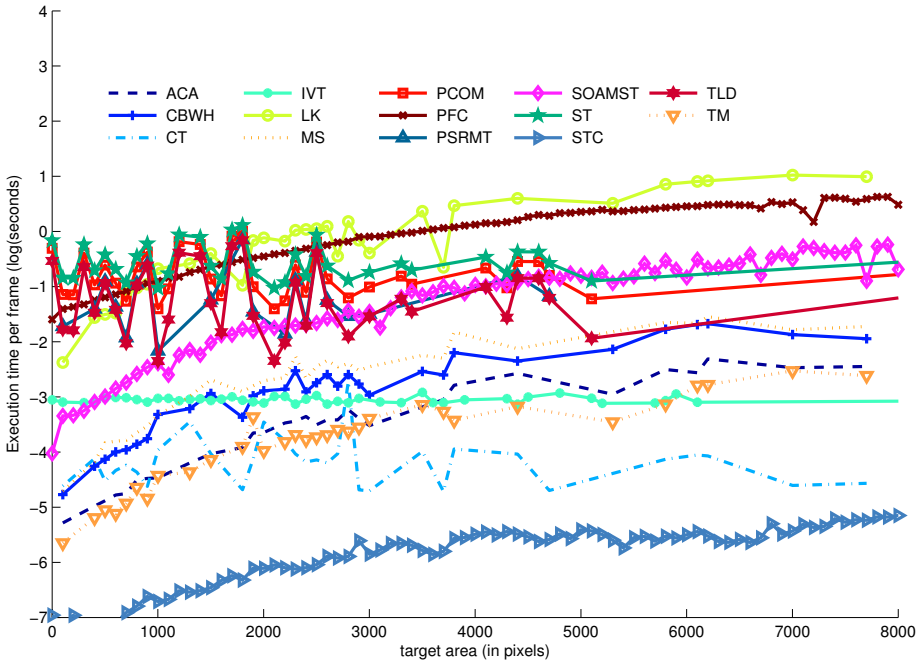


Figure 13. Execution time of each tracker (in logarithmic scale) versus the area of the target being tracked (in pixels).

and drifting (of target model). For the probabilistic tracker (PFC) and many trackers by detection (ST, PCOM), the most relevant parameter regards the predicted target position (σ_{pos}) instead of the number of particles, which depends on the expected target motion. Patch-based trackers (IVT and TLD) are very dependant on the size of such template and it should be fixed for all target types. Parameters in charge of model update schemes (*batchsize* of IVT and *maxbbox* of TLD) are also not stable showing that automatic update is still an open issue.

Concerning results of initialization three findings are noteworthy: first, a slight reduction of ground-truth size is preferred to avoid annotation errors and improve performance; second, non-accurate target initialization frequently leads to errors for automatic updating of target models; third, all trackers have a trend for size-position changes, showing that the higher the overlap, the better results (as expected).

Some conclusions can be extracted from the analysis of the tracking prob-

lems. CBWH, ACA and PCOM show best results in most of the experiments as discarding background data and modeling noisy inputs are good strategies to improve tracking (MS). As CBWH tracks fixed-size targets, it demonstrates that size adaptation is not fully solved in real scenarios (see results of SOAMST, LK, IVT and TLD). Context information is difficult to use for managing the update of the target model (STC). Multi-hypothesis trackers such as ST globally improve the performance but increase the computational cost. Robustness to noise is achieved by all the trackers and illumination changes are partially handled by the evaluated trackers (PCOM, PSRMT, CBWH, LK, TLD, IVT). For occlusions and similar objects, selected trackers obtain low performance even in presence of short-term occlusions. Finally, a noticeable performance drop is observed in sequences mixing problems (situation S4) which represents complex real data. Unlike the trend exhibited by many trackers, PFC has better results for real data as it handles data complexity more effectively.

Conclusion

In this paper, we have presented a methodology for performance evaluation of single-object visual tracking based on ground-truth data. It proposes a standard procedure for comparing trackers on sequences that represent the most relevant problems. In particular, we consider four situations ranging from controlled (synthetic sequences) to uncontrolled (real complex sequences) conditions. For each one, a set of sequences is generated for each problem with different degrees of complexity. This dataset can be extended by including video sequences from large-scale evaluations [45]. This methodology evaluates tracker performance in terms of its parameter stability, robustness to initialization, global accuracy and computational complexity. For estimating accuracy, a novel measure is proposed that compensates the errors made by the annotators (mainly in the target borders) based on the widely used spatial overlap measure. Finally, experiments are performed to demonstrate the utility of the proposed methodology. We compare the proposed accuracy measure against the representative state-of-the-art ones demonstrating its utility for high, medium and low error cases. Then, we apply the proposed methodology to evaluate relevant state-of-the-art trackers against different tracking problems.

As future work, we will focus on extending the proposed approach to evaluate multi-target tracking.

Acknowledgment

This work has been partially supported by the Spanish Government (TEC2014-53176-R HAVideo).

References

[1] (Last accessed, 05 Apr 2013). Institut fur Algorithmen und Kognitive Systeme: Cars Dataset. http://i21www.ira.uka.de/image-sequences/.

[2] (Last accessed, 05 Apr 2013). Mit traffic data set. http://www.ee.cuhk.edu.hk/ xgwang/MITtraffic.html.

[3] AVSS2007 (Last accessed, 05 Apr 2013). I-LIDS dataset for avss 2007. http://www.avss2007.org/.

[4] Bailer, C., Pagani, A., and Stricker, D. (2014). A superior tracking approach: Building a strong tracker through fusion. In *European Conf. on Computer Vision*, page (In press).

[5] Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *Int. Journal of Computer Vision*, 56(3):221–255.

[6] Bashir, F. and Porikli, F. (2006). Performance evaluation of object detection and tracking systems. In *Proc. IEEE Int. Workshop Perform. Eval. Track. Surveill.*, pages 7–14, New York (USA).

[7] Baumann, A., Boltz, M., Ebling, J., Koenig, M., Loos, H. S., Merkel, M., Niem, W., Warzelham, J. K., and Yu, J. (2008). A review and comparison of measures for automatic video surveillance systems. *EURASIP J Image Video Process*, 2008:1–30.

[8] Benfold, B. and Reid, I. (2011). Stable multi-target tracking in real-time surveillance video. In *IEEE Int. Conf. on Comput. Vision and Pattern Recog.*, pages 3457–3464.

[9] Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP J Image Video Process*, 2008:1–10.

[10] Birchfield, S. Elliptical Head Tracking Using Intensity Gradients and Color Histograms.
http://www.ces.clemson.edu/ stb/research/headtracker/.

[11] Black, J., Ellis, T., and Rosin, P. (2003). A nov.el method for video tracking performance evaluation. In *Proc. IEEE Int. Workshop Perform. Eval. Track. Surveill.*, pages 125–132, Nice (France).

[12] Brunelli, R. (2009). *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley Publishing.

[13] Carvalho, P., Cardoso, J. S., and Corte-Real, L. (2012). Filling the gap in quality assessment of video object tracking. *Image Vision Comput.*, 30(9):630 – 640.

[14] CAVIAR (Last accessed, 05 Apr 2013). Context Aware Vision using Image-based Active Recognition.
http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.

[15] Chu, D. and Smeulders, A. (2010). Thirteen hard cases in visual tracking. In *Proc. IEEE Adv. Video-Based Signal Surveill.*, pages 103–110, Boston (USA).

[16] Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *IEEE Trans Pattern Anal. Mach. Intell.*, 25(5):564–577.

[17] Danelljan, M., Khan, F. S., Felsberg, M., and van de Weijer, J. (2014). Adaptive color attributes for real-time visual tracking. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, page (In press).

[18] Doermann, D. and Mihalcik, D. (2000). Tools and techniques for video performance evaluation. In *Proc. Int. Conf. Pattern Recog.*, pages 167–170.

[19] Edward, K., Matthew, P., and Michael, B. (2009). An information theoretic approach for tracker performance evaluation. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 1523 –1529.

[20] Gao, Y., Ji, R., Zhang, L., and Hauptmann, A. (2014). Symbiotic tracker ensemble toward a unified tracking framework. *IEEE Trans. on Circuits and Systems for Video Technology*, 24(7):1122–1131.

[21] Hong, S., Kwak, S., and Han, B. (2013). Orderless tracking through model-averaged posterior estimation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2296–2303.

[22] ICPR2012 (Last accessed, 05 Apr 2013). People tracking in wide baseline camera networks. http://www.wide-baseline-camera-network-contest.org/.

[23] Kalal, Z., Mikolajczyk, K., and Matas, J. (2012). Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1409–1422.

[24] Karasulu, B. and Korukoglu, S. (2011). A software for performance evaluation and comparison of people detection and tracking methods in video processing. *Multimedia Tools Appl.*, 55(3):677–723.

[25] Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Bowers, R., Boonstra, M., Korzhova, V., and Zhang, J. (2009). Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):319–336.

[26] List, T., Bins, J., Vazquez, J., and Fisher, R. (2005). Performance evaluating the evaluator. In *Proc. IEEE Int. Workshop Perform. Eval. Track. Surveill.*, pages 129–136.

[27] Maggio, E. and Cavallaro, A. (2011). *Video tracking: theory and practice*. Wiley.

[28] Nawaz, T. and Cavallaro, A. (2013). A protocol for evaluating video trackers under real-world conditions. *IEEE Trans. on Image Process.*, 22(4):1354–1361.

[29] Nghiem, A., Bremond, F., Thonnat, M., and Valentin, V. (2007). Etiseo, performance evaluation for video surveillance systems. In *Proc. IEEE Adv. Video-Based Signal Surveill.*, pages 476–481, London (UK).

[30] Ning, J., Zhang, L., Zhang, D., and Wu, C. (2012a). Robust mean shift tracking with corrected background-weighted histogram. *IET Computer Vision*, 6(1):62–69.

[31] Ning, J., Zhang, L., Zhang, D., and Wu, C. (2012b). Scale and orientation adaptive mean shift tracking. *IET-Computer Vision*, 6(1):52–61.

[32] Nummiaro, K., Koller-Meier, E., and Van Gool, E. (2003). An adaptive colour-based particle filter. *Image and Vision Computing*, 2(1):99–110.

[33] Oron, S., Bar-Hillel, A., Levi, D., and Avidan, S. (2014). Locally orderless tracking. *Int. Journal of Computer Vision*, pages 1–16.

[34] Pang, Y. and Ling, H. (2013). Finding the best from the second bests-inhibiting subjective bias in evaluation of visual tracking algorithms. In *Proc. of IEEE Int. Conf. on Computer Vision*, pages 1–8, Sidney (Australia).

[35] PETS Datasets (Last accessed, 05 Apr 2013). IEEE Int. Workshop Perform. Eval. Track. Surveill. (2001-2007). http://www.cvg.rdg.ac.uk/datasets/index.html.

[36] PETS2000 (Last accessed, 05 Apr 2013). IEEE Int. Workshop Perform. Eval. Track. Surveill. (2000). ftp://ftp.pets.rdg.ac.uk/pub/PETS2000.

[37] PETS2010 (Last accessed, 05 Apr 2013). IEEE Int. Workshop Perform. Eval. Track. Surveill. (2010). http://pets2010.net/.

[38] Popoola, J. and Amer, A. (2008). Performance evaluation for tracking algorithms using object labels. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 733 –736.

[39] Rasid, L. N. and Suandi, S. A. (2010). Versatile object tracking standard database for security surveillance. In *Proc. Int. Conf. Inform. Science*, pages 782–785.

[40] Ross, D. A., Lim, J., Lin, R.-S., and Yang, M.-H. (2008). Incremental learning for robust visual tracking. *Int. J. Comput. Vision*, 77(1-3):125–141.

[41] SanMiguel, J., Cavallaro, A., and Martinez, J. (2012). Adaptive online performance evaluation of video trackers. *IEEE Trans. Image Process.*, 21(5):2812 –2823.

[42] Schlogl, T., Beleznai, C., Winter, M., and Bischof, H. (2004). Performance evaluation metrics for motion detection and tracking. In *Proc. IEEE Int. Conf. Pattern Recogn.*, volume 4, pages 519 – 522 Vol.4.

[43] Sebastian, P., Comley, R., and Voon, Y. (Dec. 2011). Performance evaluation metrics for video tracking. *IETE Tech. Review*, 28(6):493–502.

[44] Sipser, M. (2006). *Introduction to the Theory of Computation*, volume 27. Thomson Course Technology Boston, MA.

[45] Smeulders, A., Chu, D., Cucchiara, R., Calderara, S., Dehghan, A., and Shah, M. (2014). Visual Tracking: An Experimental Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468.

[46] SPEVI (Last accessed, 05 Apr 2013). Surveillance Performance EValuation Initiative. http://www.eecs.qmul.ac.uk/ andrea/spevi.html.

[47] TRECVID-SED (Last accessed, 05 Apr 2013). TRECVID 2009 Surveillance Event Detection dataset. http://trecvid.nist.gov/trecvid.data.html.

[48] Vezzani, R. and Cucchiara, R. (2010). Video surveillance online repository (visor): an integrated framework. *Multimedia Tools App.*, 50(2):359–380.

[49] Wang, D. and Lu, H. (2014). Visual tracking via probability continuous outlier model. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, page (In press).

[50] Wang, D., Lu, H., and Yang, M.-H. (2013). Online object tracking with sparse prototypes. *IEEE Transactions on Image Processing*, 22(1):314–325.

[51] Wang, Q., Chen, F., Xu, W., and Yang, M. H. (2011). An experimental comparison of online object-tracking algorithms. In *Proceedings of the SPIE*, pages 81381A–81381A.

[52] Wu, H., Sankaranarayanan, A., and Chellappa, R. (2010). Online empirical evaluation of tracking algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1443–1458.

[53] Wu, Y., Lim, J., and Yang, M. H. (2013). Online object tracking: A benchmark. In *Proc. of IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 1–8, Portland (Oregon, USA).

[54] Yang, F., Lu, H., and Yang, M.-H. (2014). Robust superpixel tracking. *IEEE Trans on Image Processing*, 23(4):1639–1651.

[55] Yang, H., Shao, L., Zheng, F., Wang, L., and Song, Z. (2011). Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18):3823 – 3831.

[56] Yi, K. M., Jeong, H., Heo, B., Chang, H. J., and Choi, J. Y. (2013). Initialization-insensitive visual tracking through voting with salient local features. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2912–2919.

[57] Yin, F., Makris, D., Velastin, S., and Orwell, J. (2010). Quantitative evaluation of different aspects of motion trackers under various challenges. *Annals of the BMVA*, 5:1–11.

[58] Zhang, K., Zhang, L., and Yang, M. (2014a). Fast compressive tracking.

[59] Zhang, K., Zhang, L., Yang, M.-H., and Zhang, D. (2014b). Fast tracking via spatio-temporal context learning. In *European Conf. on Computer Vision*, page (In press).

[60] Zhang, L. and van der Maaten, L. (2014). Preserving structure in model-free tracking. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 36(4):756–769.

[61] Zhang, Z. and Wong, K. H. (2014). Pyramid-based visual tracking using sparsity represented mean transform. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, page (In press).

INDEX

A

adaptability, 131, 133, 134
adaptation(s), 60, 62, 65, 84, 110, 131, 132, 136
algorithm, viii, 2, 3, 4, 10, 11, 21, 24, 27, 35, 36, 37, 48, 49, 50, 53, 55, 57, 58, 59, 60, 61, 62, 63, 64, 65, 68, 71, 72, 73, 74, 75, 77, 78, 79, 80, 81, 85, 86, 87, 88, 92, 93, 94, 96, 98, 99, 100, 101, 104, 105
annotation, 125, 128, 133, 135
assessment, 93, 138

B

background information, 129
background subtraction, viii, 25, 26, 37, 39, 56, 57, 59, 62, 63, 64, 65, 66, 67, 72, 74, 75, 77, 78, 80, 85, 87, 88, 90, 91, 94, 99, 100, 102, 103, 104, 105, 106
benefits, 10, 109
blood pressure, 9
body shape, 35, 36, 43
Bureau of Justice Statistics, 4

C

challenges, 108, 112, 113, 133, 142
China, 52
Coast Guard, 50
color, ix, 56, 58, 60, 63, 64, 65, 67, 76, 80, 81, 85, 86, 87, 88, 90, 91, 95, 96, 100, 101, 102, 110, 115, 116, 117, 128, 130, 131, 133, 138
combined tracker, 59, 91, 95, 99
community, 7, 108
compensation, 65
complexity, ix, 13, 21, 22, 24, 79, 92, 107, 108, 109, 112, 113, 114, 115, 116, 117, 123, 126, 128, 133, 134, 136
compression, 7, 21, 23, 24, 67
computation, ix, 56, 78, 81, 82, 83, 85, 87, 88, 90, 99, 100, 124
computer, vii, viii, 2, 7, 9, 12, 13, 105
computer vision field, vii, 2, 12
computing, 52, 56, 78, 80, 81, 85, 90, 91, 95, 105
configuration, ix, 6, 109, 112, 113, 114
conflict, ix, 56, 57, 75, 76, 77, 78, 80, 86, 87, 88, 89, 90, 91, 95, 99
consumption, 79
content analysis, 56, 104, 105
contour, 64, 65
correlation, 82, 124
cost, 3, 5, 6, 7, 10, 11, 13, 78, 99, 110, 134, 136

D

damages, 50
danger, 3
data processing, 106
data set, 137
database, 140
defects, 63
deformation, 3, 15
degradation, 129
depth, 111
detection, vii, viii, 1, 2, 3, 4, 5, 6, 7, 10, 24, 27, 36, 37, 39, 42, 48, 49, 52, 53, 55, 56, 57, 58, 59, 61, 62, 63, 65, 67, 71, 72, 73, 74, 75, 80, 91, 101, 102, 103, 104, 105, 106, 110, 112, 126, 135, 139, 141
detection system, 6, 7
deviation, 60, 108, 116, 120
digital cameras, 7, 11
dimensionality, 110
direct measure, 75
discontinuity, 3, 13
dispersion, 80, 86
distribution, 60, 68, 78, 79, 80, 83
distribution function, 83
DOI, 106

E

elders, 10
electromagnetic, 6
elongation, 23
emergency, 4, 9, 10
emergency response, 9
energy, 56, 79
energy consumption, 79
environmental conditions, 6
environmental effects, 5
environment(s), 8, 12, 24, 53, 58, 77, 104, 133
equipment, 8
erosion, 63
evidence, vii, 1, 4, 11
execution, 24, 133, 134
extraction, 25, 26, 64, 87
extracts, 56

F

faint detection, vii, 2, 3, 4, 24, 27, 49
fainting, 3, 11, 24, 36, 49
false alarms, 5, 6, 42
false negative, 66, 92
false positive, 61, 66, 92
feature selection, 12
features extraction, 26
FFT, 110
filters, viii, 55, 57, 58, 68, 75, 78, 99, 100, 101, 102, 104, 105
force, 6, 67
France, 138
fusion, 137

G

GPS, 9
graph, 39, 42, 44, 45, 46, 47, 77, 93
grouping, 101
guidelines, vii, viii, 55, 72, 100

H

health, vii, viii, 1, 2, 8, 9, 10, 50
health care, vii, viii, 1, 2, 8, 10
health care surveillance system, vii, 1, 2, 8, 10
health information, 9
health-care professional, vii, 1, 8
height, 23, 50, 64
helium, 1
hemisphere, 13
histogram, ix, 56, 57, 58, 76, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 95, 96, 100, 103, 139
history, 53
HLS, 81
home break-ins, vii, 1
home intrusion, vii, 1

house, 50, 102
human, vii, viii, ix, 1, 2, 11, 25, 35, 36, 42, 50, 53, 56, 57, 102, 107, 119
human body, 35, 36
human faint detection, vii, 1
Hunter, 53
hypothesis, 78, 79, 86, 125, 136

I

identification, 100, 104, 106
illumination, ix, 5, 36, 50, 107, 108, 114, 115, 116, 117, 118, 131, 132, 136
image analysis, 102
image processing, 3, 10, 11, 13, 15, 24, 25, 49, 53
image(s), vii, viii, ix, 2, 3, 7, 8, 10, 11, 12, 13, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 35, 37, 38, 39, 43, 48, 49, 52, 53, 56, 57, 58, 59, 60, 62, 63, 64, 67, 68, 76, 81, 82, 83, 84, 87, 88, 90, 92, 96, 102, 103, 105, 106, 114, 115, 137
inattentiveness, vii, 1, 8
independence, 9
individuals, 39, 43, 48, 49
initial state, 72
injuries, 3, 8
injury, vii, 1, 8, 10
integration, 10
interface, 101
interference, 4, 6, 7
issues, viii, 2, 52
iteration, 126

K

Kalman filters, viii, 55, 57, 58, 68, 99, 101, 104, 105

L

laptop, 13
late notification, vii, 1, 8
learning, 5, 61, 62, 65, 110, 125, 139, 140, 142
LED, 5
lens, 12, 13, 51
light, 5, 6, 9, 24, 38, 59, 60, 61, 62, 65, 83

M

magnetic field, 6, 7
Malaysia, 1
manpower, 8, 10
mapping, viii, 2, 3, 12, 15, 17, 18, 19, 20, 21, 22, 23, 24, 49, 52
mass, 78
matrix, 16, 34, 70, 71, 72, 75, 88, 91
matter, 4, 9, 12
measurement(s), ix, 56, 58, 59, 68, 71, 72, 73, 74, 75, 76, 77, 80, 84, 86, 88, 89, 90, 91, 94, 95, 105
medical, 8, 9
memory, 24, 62, 63, 81, 85, 123
methodology, vii, ix, 56, 100, 107, 108, 109, 113, 123, 134, 136
mobile robots, 12
models, 15, 57, 60, 69, 70, 78, 79, 102, 103, 110, 113, 128, 133, 135
modifications, 78, 123
monitoring surveillance systems, viii, 2
Monte Carlo method, 78
morphology, 53
multimedia, 104, 105

N

nonfatal fall injury, vii, 1, 8
normal distribution, 68, 78, 80, 90

O

object detection, viii, 2, 55, 56, 57, 58, 59, 63, 65, 67, 71, 72, 73, 74, 75, 80, 91, 102, 103, 104, 105
object tracking, viii, 53, 55, 56, 57, 58, 59, 67, 68, 75, 78, 83, 85, 88, 92, 96, 99,

100, 101, 103, 104, 105, 108, 109, 138, 140, 141, 142
occlusion, ix, 50, 56, 57, 64, 66, 67, 75, 76, 77, 86, 89, 94, 100, 114, 116, 117, 130, 131, 132
omnidirectional, v, vii, viii, 1, 2, 3, 4, 11, 12, 13, 14, 15, 17, 18, 19, 21, 22, 23, 24, 49, 50, 51, 52, 53
OPA, 48, 49
operations, 56, 63, 94
optimization, 125
overlap, 57, 71, 76, 90, 111, 119, 122, 124, 128, 130, 135, 136

P

parallel, 6, 62, 82, 103, 105, 106
parallel implementation, 62, 105
parallel processing, 105
particle filters, viii, 55, 57, 58, 68, 75, 78, 100, 101, 102, 105
PCA, 110
physical well-being, 8
platform, 103, 104, 106
Poland, 55, 104
polar, viii, 2, 3, 15, 19, 20, 21, 22, 23, 24, 25, 49, 52, 102
police, vii, 1, 4, 7
prevention, 7, 8, 104
probability, 78, 79, 83, 141
probability density function, 79
probability distribution, 83
programming, 105
project, 101, 104, 105
propagation, 69, 102
prototypes, 141

R

radar, 79
radius, 15, 16, 17, 18, 19, 20, 21, 23
recall, 92, 93, 95, 97, 99, 111
reconstruction, 103
requirements, 7, 62, 81, 85, 112
researchers, viii, 2, 10
resolution, 13, 16, 17, 21, 23, 51, 62, 63, 94
resources, 57, 62, 85
response, 4, 7, 9
restoration, 8
risk, 8, 67, 82, 84, 86, 87, 99
robotic vision, 19
robotics, 11, 12, 52
ROC, 93, 95
root-mean-square, 93
rotating camera, 12
routines, 78

S

safety, 2, 51
security, vii, viii, 1, 2, 4, 6, 7, 104, 140
security systems, vii, 1
security threats, 104
SED, 141
senses, 6
sensing, 110
sensitivity, 6, 66, 67, 120
sensor(s), 3, 6, 7, 8, 9, 10, 11, 56, 68, 72, 74, 101
services, 8
signals, 6, 8
smoothness, 111
software, 11, 13, 94, 109, 139
solution, 3, 8, 13, 58, 65, 75, 87, 100
SPA, 105, 106
Spain, 107
specialists, vii, viii, 55
stability, ix, 99, 107, 109, 113, 114, 120, 121, 122, 126, 128, 134, 136
standard deviation, 60
state(s), ix, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 83, 84, 86, 88, 89, 90, 91, 94, 108, 136
statistics, 9, 50
storage, 7, 10
structure, 6, 20, 67, 109, 110, 142
subtraction, viii, 25, 26, 37, 39, 56, 57, 59, 62, 63, 64, 65, 66, 67, 72, 74, 75, 77, 78,

80, 85, 87, 88, 90, 91, 94, 99, 100, 102, 103, 104, 105, 106
suppression, 101
surveillance, vii, viii, 1, 2, 3, 4, 5, 7, 8, 10, 11, 12, 13, 14, 50, 53, 55, 56, 87, 92, 99, 100, 101, 102, 106, 110, 137, 139, 140, 141
surveillance system, vii, viii, 1, 2, 3, 4, 8, 10, 11, 13, 14, 50, 55, 56, 92, 99, 137, 139
symmetry, 27, 29, 31, 33
synchronization, 85
synthesis, 105

T

target, 19, 62, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 100, 102, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 122, 123, 124, 125, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137
TBS, 26, 37, 38
techniques, 3, 15, 21, 84, 104, 138
technologies, 105
technology, vii, 2, 6, 7, 10, 12
test procedure, 92
testing, ix, 56, 92, 94, 109, 117, 120, 123
theft, 4
threats, 104
time constraints, 123
tracker, vii, ix, 9, 57, 58, 59, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 79, 80, 81, 82, 83, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 98, 99, 107, 108, 110, 111, 112, 113, 114, 115, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 133, 135, 136, 137,138
tracking conflict, ix, 56, 57, 58, 59, 68, 71, 75, 77, 78, 80, 84, 85, 87
tracks, 9, 56, 58, 59, 74, 78, 89, 92, 95, 99, 104, 111, 112, 131, 132, 136
trajectory, 111, 115
transducer, 6
transformations, 125
translation, 125
treatment, 8
trespasser detection, vii, 1, 2, 5, 6, 24, 27, 48, 49

U

uniform, 16, 17, 79, 96, 99
United States (USA), vii, 1, 4, 7, 52, 137, 138, 142
updating, ix, 56, 58, 59, 71, 73, 75, 80, 84, 85, 88, 89, 91, 94, 135
urban, 104

V

valuation, vii, ix, 107, 108, 111, 113, 114, 126
variables, 69, 70, 71, 72, 73, 76, 78, 79, 83, 90, 94
variations, vii, ix, 61, 65, 73, 107, 114, 115, 130
vector, 69, 71, 72, 79, 80, 83, 90
vegetation, 6
vehicles, 12, 59, 70, 74, 75, 76, 94, 100, 104, 105, 106
velocity, 69, 72, 73, 75, 76, 84, 88, 90, 95
vibration, 5, 6, 7
victims, 4
videos, 63, 112
vision, vii, viii, 2, 10, 12, 13, 19, 101, 102
visual attention, 19
visual field, 11
visual surveillance, vii, 2, 3
visualization, 12

W

web service, 105
well-being, 8
wide area coverage, viii, 2
witnesses, vii, 1, 4