Texts in Computer Science

Wei Qi Yan

Introduction to Intelligent Surveillance

Surveillance Data Capture, Transmission, and Analytics

Third Edition

Springer

Wei Qi Yan

Introduction to Intelligent Surveillance

Surveillance Data Capture, Transmission, and Analytics

Third Edition

Springer

Wei Qi Yan
Department of Computer Science
Auckland University of Technology
Auckland, New Zealand

ISSN 1868-0941 ISSN 1868-095X (electronic)
Texts in Computer Science
ISBN 978-3-030-10712-3 ISBN 978-3-030-10713-0 (eBook)
https://doi.org/10.1007/978-3-030-10713-0

Library of Congress Control Number: 2018966855

Preface

Digital surveillance is a leading-edge security problem and has been unprecedentedly applied to monitor and protect our lives almost anywhere. It is ubiquitous and practical and could greatly lessen security staff's human labor. With the aid of intelligence, surveillance systems have the capability to automatically complete tasks such as object detection, recognition, tracking as well as event life cycle including event detection, recognition, search, retrieval, mining, reasoning, etc. With the development of artificial intelligence (AI), the systems have taken in from the progression of supercomputing, cloud computing, big data, deep learning, etc.

This book starts from camera calibration, along with surveillance data capturing, scrambling and descrambling, secure transmission with secure network environment; afterward, the surveillance at object level has been introduced such as detection, recognition, and tracking. Biometrics is presented as an important part of this book. A picture is more than a thousand of words, an event is more than thousands of pictures. An event is the basic unit of knowledge which bridges the gap between physical world and semantic objects; the life cycle of an event includes generating new ones and operations on events, etc. The knowledge based on events could be used for discovery and exploration, and this will be employed for surveillance alarm making; at the end of this book, fundamentals of supercomputing (FPGA, GPU, and parallel computing), cloud computing, mobile computing, deep learning, etc., will be emphasized.

This book is based on our research and teaching experience, we have used the content of this book for postgraduate teaching in the higher education. The stuff of the whole course including assignments and examinations has been verified for a plurality of times and could serve the readers of this book well.

This book was written for research students and engineers as well as scientists who are interested in intelligent surveillance.

Auckland, New Zealand   Wei Qi Yan
October 2018

Acknowledgements

The first edition of this book has been published in March 2016. In the past two years, we have endowed in all aspects to make the book full and perfect. Following the feedback from readers and audiences responses, the author has further omitted mistakes and typos, detailed the mathematical descriptions and algorithms, as well as updated each chapter with the latest contents and references.

The second edition of this book was published in June 2017 and emphasized on red-hot technology such as deep learning, mobile and cloud computing, and big data in intelligent surveillance. The author's endeavor was about how surveillance research and teaching could take in nutrition from the progress of other fields and what the audience and readers could pay their attention to when they read this book.

The third edition of this book is emphasized on human behavior analysis, privacy preservation, and the details of deep learning and artificial intelligence (AI). We integrate the latest development in AI and machine learning into this book for meeting the trends of today's research. The book shows how machine intelligence could assist security people in surveillance with regard to the fundamental aspects: observation, learning, presentation, and reasoning or references.

Thanks to our peer colleagues and students whose materials were referenced and who have given invaluable comments on this book. Special thanks to my supervised students: Dr. J. Weir, Dr. D. Kieran, Mr. Y. Jiao, Mr. G. Wang, Mr. W. Cui, Mr. D. Gu, Mr. J. Wang, Dr. Y. Zhang, Mr. L. Zhou, Mr. J. Lu, Mr. J. Shen, Mr. D. Shen, Mr. K. Zheng, Ms Y. Ren, Mr. R. Li, Mr. P. Li, Mr. Z. Liu, Ms. Y. Shen, Ms. H. Wang, Mr. C. Xin, Ms. Q. Zhang, Ms X. Zhang, Dr. Q. Gu, our colleagues: Dr. M. Nguyen, Prof. R. Klette, et al.

Contents

About the Author

Dr. Wei Qi Yan is Associate Professor with the Auckland University of Technology (AUT); his expertise is in digital security, surveillance, privacy, and forensics; he is leading the Computing Cybersecurity (CCS) Research at AUT. He is Editor-in-Chief (EiC) of the International Journal of Digital Crime and Forensics (IJDCF); he was an exchange computer scientist between the Royal Society of New Zealand (RSNZ) and the Chinese Academy of Sciences (CAS), China. He is a guest (adjunct) professor with Ph.D. supervision of the Chinese Academy of Sciences, China; he was Visiting Professor of the University of Auckland, New Zealand, and the National University of Singapore, Singapore.

1 Introduction

1.1 Introduction to Surveillance

The slumped costs coupled with rapid miniaturization of video cameras have enabled its widespread use on highways, airports, railway stations, and on-road vehicles. Intelligent surveillance has become an essential and practical mean for security which combines computer science and engineering with multidisciplinary studies including data repository, computer vision [14, 17], digital image processing, computer graphics as well as computational intelligence. Different from traditional monitoring, scenes related to security concerns are possible to be monitored automatically and remotely with assistance of intelligent surveillance equipped.

Security in surveillance are closely related to public safety. The threats from attacks have been stressed significantly by governments, and asset security is always a crucial concern for all organizations. The demand of remote surveillance system for safety and security is mainly in the areas such as transport applications, maritime environments, public places, and military applications.

Basic equipment needed in intelligent surveillance consists of calibrated sensors which are installed and linked under the control of a monitoring center. The advantages of monitoring a broad scope of district effectively facilitate security staff's work as well as lessen the required number of guardians for monitoring. It is well known that an effective surveillance system should greatly reduce security staff's human labor.

The pipeline of a surveillance system includes moving object detection, recognition, tracking, event analyzing, database archiving, and alarm making as shown in Fig. 1.1.

Moving object detection in visual surveillance is the key step, especially for foreground and background separation. Background subtraction is subjective to foreground changes and has better performance to each moving object, especially in object tracking [16]. The conventional technique of video frame difference includes

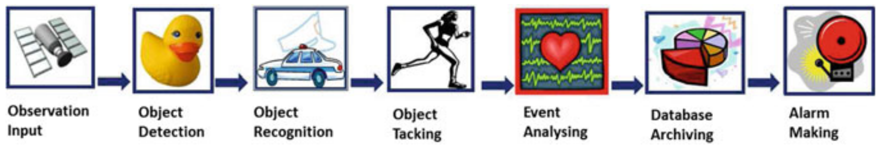


Fig. 1.1 Pipeline of a surveillance system

the subtraction of two consecutive frames simply followed by thresholding and analyzing gradient or histogram; key points as a typical feature of motion also have been applied to foreground and background separation. Nowadays, motion analysis and optical flow by using information theory and deep learning have been employed to video dynamic analysis [42] and could get surprising results [9].

Object tracking techniques are split into two main categories: 2D and 3D models. The model-based approaches explicitly take use of a priori geometrical information of moving objects which usually include pedestrians, vehicles, or both in surveillance applications. A number of approaches have been applied to classify newly detected objects and furthermore to be used for object tracking.

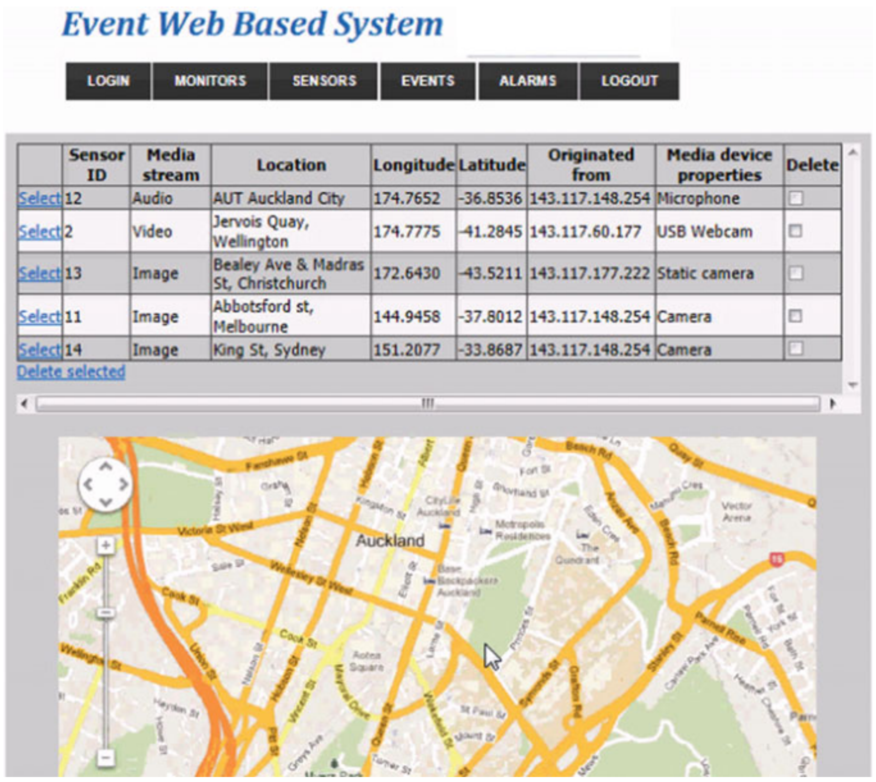
At the stage of object tracking, filtering (Kalman filetering, Bayesian filtering, particle filtering, etc.) is used to predict every position of the target object. Another tracking approach uses connected components to segment the changes into different objects without prior knowledge. The approach exhibits a good performance when the object has a very low resolution.

A plenty of scenes having high security interest, such as airports, train stations, shopping malls, and street intersections, usually are very crowded. It is impossible to track objects over long distance without failures because of frequent occlusions among objects in such scenes. Although a great deal of existing surveillance systems work well in sparse scenes, there are still a slew of challenges unsolved in the real applications.

Intelligent surveillance is beneficial from recognition and understanding of actions, activities, and behaviors of the tracked objects. This stage corresponds to classification of the time-varying features that are acquired from the preceding stages. Therefore, the tracking comprises of matching a picture sequence to a well-prepared dataset having labels that needs to be learned via data training process.

One of the key stages in intelligent surveillance is search and retrieval that are the crucial aspect, especially for alarm making. Relevant research work has been conducted in how to store and retrieve all the obtained surveillance events in an efficient manner, especially over the cloud.

Alarm making is the ultimate goal of intelligent surveillance; the difficulty in alarm making is to reduce false alarms which easily make security staff annoyed and potentially miss the positive alarms. We thus insist working for alarm making approaches, typically simple and complex alarming. Alarm making could be spatially-based or temporally-based, but it should follow the process of decision making basically. Hence, decision trees and random forests could help in decision making.



	Sensor ID	Media stream	Location	Longitude	Latitude	Originated from	Media device properties	Delete
Select	12	Audio	AUT Auckland City	174.7652	-36.8536	143.117.148.254	Microphone	☐
Select	2	Video	Jervois Quay, Wellington	174.7775	-41.2845	143.117.60.177	USB Webcam	☐
Select	13	Image	Bealey Ave & Madras St, Christchurch	172.6430	-43.5211	143.117.177.222	Static camera	☐
Select	11	Image	Abbotsford st, Melbourne	144.9458	-37.8012	143.117.148.254	Camera	☐
Select	14	Image	King St, Sydney	151.2077	-33.8687	143.117.148.254	Camera	☐

Delete selected

Fig. 1.2 Event-based surveillance system

An event-based surveillance system shown in Fig. 1.2 was developed which has the features: login, monitors, sensors, events, alarms, and logout.

1.1.1 Surveillance History

Surveillance is deemed to have experienced three generations. The first generation (1G) is analog closed-circuit television (CCTV). In this stage, distribution and storage of surveillance information are taking advantage of analog technologies, digital-to-analog (D/A) conversion has taken the pivotal role. The second generation (2G) is aided by utilizing computer vision technology [14]; typically event detection [5], behavior analysis, machine learning and deep learning, scene understanding, and semantic interpretations have been adopted in surveillance, semantic analysis was emphasized on this generation shown in Fig. 1.3.

Since 1G systems were initially developed in the middle of the 1960s which are based on analog signals, the CCTV systems are a local television system in which TV scenes are transmitted over a very relatively short distance. Conventional CCTV

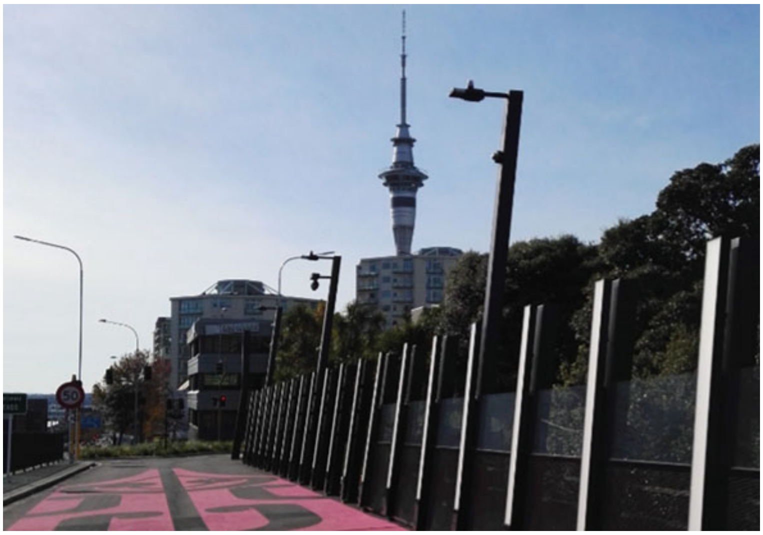


Fig. 1.3 Cameras mounted everywhere at Nelson Street Cycleway

cameras generally used a digital charge-coupled device (CCD) to capture images. The captured images were converted into an analog composite video signal, which is connected to CCTV monitors and recording equipment with magnetic media, generally via coaxial cables. At this stage, image distribution and storage are in analog; D/A conversions have been adopted for digital image and video processing.

Surveillance systems at this stage were manually based and utilized to recognize, track, and identify any objects, different kinds of security infractions and violations are based on human observations by staring at monitor screens. 1G intelligent surveillance systems have the advantageous characteristics of visual performance. However, D/A conversion has a delay which highly generates image degradations and the analog signals were sensitive to noises and easily disturbed by strong electromagnetic sources. Furthermore, security staff is easily exhausted and distracted to multiple-target tasks after several hours of gazing at the cathode ray tube (CRT) monitors. Therefore, 1G surveillance systems are not sufficient for automated processing. The typical challenges in 1G systems include digital and analog signal conversation, digital video recording and storage, artifact removal, and video compression.

The technological improvement led to develop the 2G surveillance systems, namely semi-automatic surveillance systems. Compared to the 1G systems, 2G systems have the merits of increasing the efficiency by using computer vision-based visual content analysis. The research problem of this generation usually lies in robust object detection and object tracking. Current research focus of 2G surveillance systems orientates to the objectives of live broadcasting in real-time and robust computer vision algorithms, machine learning of scene variability and patterns of human behaviors, bridging the gap between the physical world and natural language interpretations. The most prevalent technologies at present are serving for the 2G surveillance systems.

The 3G surveillance systems concentrate on automated multimodal system, intelligent distribution systems (integration and communication), multisensor platforms, and data fusion; probabilistic reasoning has been adopted in this phase. 3G systems are working on wide area based with more accurate information due to the deployment of distributed multisensors. On the other hand, existing problems of 3G systems include the lack of information integration and communications. Current research problems of 3G surveillance systems encompass distributed and centralized systems, data fusion, probabilistic reasoning, and multicamera surveillance techniques [7,11,31].

In 3G systems, the difficulties we encounter are related to scene understanding such as indoor, outdoor, crowds, scene geometry, and scene activity. The relevant technologies include data fusion, ontology, and syntax and semantics. Among them, the secure communications between various modules apparently are very critical. Therefore, we need to understand its communication protocols, metadata, real-time systems, etc.

Our goals in intelligent surveillance are how to save security staff's time and labor, how to make right alarms so as to reduce false ones and ensure an alarm is positive, right and effective. The reduction of human labor in intelligent surveillance depends on the maximum area of scanning region and sensitivity of the sensors whether the system can effectively and promptly extract and analyze the data captured by cameras. False alarming is a prominent issue in surveillance which is highly related to robustness and reliability of a surveillance system. False alarming not only wastes security staff's time in processing those meaningless information, but also potentially causes a real suspect to skip through the monitored area without any alerts. Therefore, reducing false alarms is the paramount goal of all surveillance systems; meanwhile, we should be aware that false alarming is hard to be prevented as the techniques of object detection, because tracking and recognition at present are still far away from the practical needs (Fig. 1.4).

Fig. 1.4 Hyperwall for the purpose of display

1.1.2 Open Problems in Intelligent Surveillance

Corresponded to difficulties in intelligent surveillance, open problems in intelligent surveillance are grouped into the categories such as hardware, software, objects, and events and ethics. At hardware layer, typical problems are sensor deployment and management, data capturing, compression, and secure transmission. At object layer, the issues include data semantic analysis and scene understanding, object analysis such as fire and smoking. At event layer, the difficulties may have event database creation and exploration, event search and retrieval, mining and reasoning, event exploration and presentation using Web/mobile [16]. At ethics layer, we have to face the problems like privacy preservation and protection in social networks, supercomputing (FPGA/GPU), and cloud [39] and mobile computing.

Digital event is a key issue in intelligent surveillance since we need a good story which bridges the gap between cyberspace and our real world. However, the definitions of an event across various research fields are very diverse and only reflect content of designated applications. For example, in textual topic detection and extraction, an event is something that happened somewhere at a certain time; in pattern recognition, an event is defined as a pattern matched with a class of patterns; in signal processing, an event is usually triggered by a state change.

In surveillance, an event at least consists of two aspects: spatial and temporal. It could be either instantaneous or spanning over a period of time. Moreover, it is atomic that an event in surveillance is in accordance with how elementary events are defined within a surveillance system. It is noteworthy that an event should be with a particular object that is correlated with others to constitute a whole scene which is meaningful and worthwhile to be studied.

Data fusion is an integration process of multiple data and knowledge representing the same real-world object into a consistent, accurate, and useful representation. Theoretically, data fusion in surveillance systems denotes techniques and tools which are used for combining sensor data, or data derived from sensory array into a common representational format. In data fusion, the goal of intelligent surveillance is to improve, filter, and refine the quality of surveillance information. A multisensor data fusion system is a major component in the fields dealing with pattern detection, recognition, etc. [7]. The aggregation of several sensors to achieve better results is always expected.

Multisensor data fusion improves the performance of surveillance systems in four ways, namely representation, certainty, accuracy, and completeness. The characteristics of *representation* denote that output information obtained during or after the process of data fusion has richer semantic meaning than that of individual input data. *Certainty* refers to the probability of data which is obtained by applying fusion. Similarly, *accuracy* is explained as the standard deviation of the data after the fusion process is smaller than that of input data. Lastly, *completeness* means new information is added into the understanding of certain environments after fusion. There are also four types of data fusion, namely fusion across sensors, fusion across attributes, fusion across domains, and fusion across time.

To implement the scalability of a large surveillance system, surveillance ontology is necessary to be considered. Ontology is related to the nature of existence as well as the basic categories of objects and their relations. Traditional ontology is listed as a part of the major branch of philosophy known as metaphysics. Ontology deals with questions concerning what entities exist and how such entities can be grouped within a hierarchy and subdivided cluster according to similarities and differences. In surveillance, each camera or sensor is treated as an agent and the ontology is adopted to standardize and merge these types of information under specific environments.

The implementation of semantic translation in surveillance is closely related to ontology. Because various sources are used to detect events with different types of information [5], a formal event model with the ability to define all its elements with clear semantics is required not only to represent events themselves but also to make inferences for subsequent alarm making.

Secure communications between various modules in surveillance are also very thorny. Therefore, we need study the communication protocols, metadata, real-time system, etc. These include distribution of processing tasks as well as creation of metadata standards or protocols to cope with current limitations in bandwidth capacities.

Metadata is also called data of data which means additional information of a given set of data. Structural metadata is about design and specification of data structures and is more properly called "data about the containers of data." On the other hand, descriptive metadata is about individual instances of application data and the data content. For instance, every pixel in a digital image is treated as the data of the image itself, while parameters of the camera, resolution of this picture and its created date and time are stored in the file header (e.g., EXIF) which are thought as the metadata of this image.

Big data usually includes datasets beyond the ability of commonly used software tools to capture, curate, manage, and process within a tolerable elapsed time, i.e., increasing *volume* (amount of data), *velocity* (speed of data in and out), and *variety* (range of data types and sources). Big data uses inductive statistics and nonlinear system identification (regressions, nonlinear relationships, and causal effects) from large sets of data with low information density to reveal relationships, dependencies and perform predictions of outcomes and behaviors. Sampling (statistics) enables the selection of right data points within the dataset. While big data continues to grow exponentially, visual surveillance has become the biggest data source since all of these surveillance cameras capture a huge amount of video and images while feeding them into cyberspace daily [13].

In recent years, with the abusive usages of surveillance data generated as a result of the frequent occurrence of security issues, concerns such as human right, ethics, privacy [6], and authentication are needed to attract attentions from publics. The abusive usage of surveillance data is severely harmful to the whole society as relevant private information might be leaked through surveillance systems, once surveillance data are intercepted, the entire public would experience trust crisis. Therefore, social, ethical, organizational, and technical issues are urgent for us to provide right solutions such as social networks typically to protect individual's privacy [6].

Our computer system has confronted with challenges in computing. The bottleneck lies in CPU speed for computations that has not taken over great step in so many years. Parallel computing technologies including multicore and multithread programming have been applied to accelerate in real applications. Coming up with the hardware such as graphics processing unit (GPU) and field-programmable gate array (FPGA), the accumulations could be sped up and expedited.

In supercomputing, networking could join computing forces from all aspects together. Mobile, cloud, and social networking have played pivotal roles in modern intelligent surveillance and will push surveillance events as notifications into those feasible and convenient devices in easy ways [39].

In this book, we will organize our content to explain these issues in different chapters:

- Sensor deployment, calibration, and management
- Media capturing, compression, and secure transmission
- Media semantic analysis and understanding
- Object tracking and event detection
- Decision making and alarm making (e.g., fire, smoking, etc.)
- Event database and exploration
- Big data surveillance
- Discrete event computing
- Event presentation
- Ethics issues/privacy in surveillance
- Supercomputing (e.g., GPU, FPGA, etc.)
- Cloud/mobile computing
- Artificial neural networks and deep learning

1.2 Introduction to Sensor Deployment and Calibration

1.2.1 Sensors

Surveillance sensors are deployed everywhere working under all weather conditions, all of them even have been connected as a network, and some of them even have been connected to the Internet [15] shown in Figs. 1.5 and 1.6. For all sensors, we would like to group the connecting models of surveillance sensors into below categories:

- *Visual sensors*: the sensors include analog camera, network camera, IP camera, Web camera, wireless camera, and infrared camera. A recent new camera Dynamic Vision Sensor (DVS) is also called silicon retina which successfully simulated Human Vision System by using the neural networks [8]. The indispensable cameras could capture not only images but also video signals.

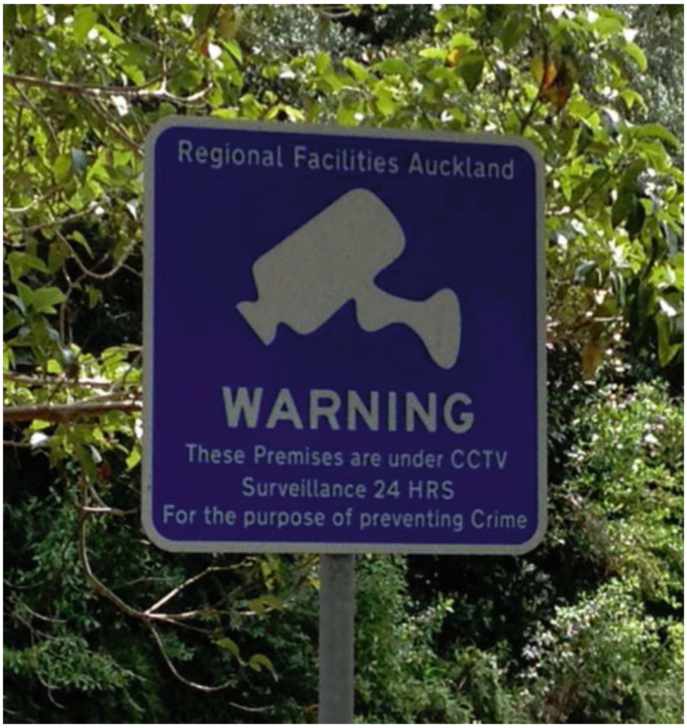

Regional Facilities Auckland
WARNING
These Premises are under CCTV
Surveillance 24 HRS
For the purpose of preventing Crime

Fig. 1.5 Sign of surveillance for the purpose of preventing crimes

- *Audio sensors*: the sensors are related to acoustic, sound, and vibration sensors for audio, sound, and voice recording; the recorded audio data also has metadata.
- *Automotive sensors*: these sensors are used for transportation, electric current, electric potential, magnetic, radio for signals. The sensors could reflect the real-time situation changes while working without stopping.
- *Environmental sensors*: the scene sensors could record the data reflecting moisture, humidity, saturation, and temperature for environmental monitoring;
- *Flow sensors*: the fluid velocity-related meters are specially designed for flow detection, such as wind and water flow. The flow changes could be revealed in the velocity.
- *Navigation sensors*: the sensors have absolute and relative positioning functionalities which have the navigation instruments embedded inside such as mobile phones.
- *Locating sensors*: the typical sensor is Global Positioning System (GPS); other sensors are for measuring altitude, angle, displacement, distance, speed, and acceleration.
- *Others*: Other sensors are specially designed for optical, force, density, pressure, thermal measurement, etc.

Fig. 1.6 Satellite system for earth monitoring

For sensor deployment, communications, cooperations, control, and power management are the fundamental issues in intelligent surveillance. Sensors are typically connected together using the topological ways such as centralized, distributed, hybrid, and multitier. Like computer networks, sensors have been linked to a central server or mutual or multitiers at various levels. The fact is that hybrid and multitier connections of miscellaneous sensors are taking effects in real surveillance systems.

A centralized network shown in Fig. 1.7 is the one where all sensors are connected to the central server for the purpose of data transmission in which the server stores both communications and user account information. Most public instant messaging platforms adopt the framework of a centralized network.

A distributed network shown in Fig. 1.8 is spread over heterogeneous networks. Distributed networks have not the central server which is emphasized on centralized network. Instead, all the server nodes are being connected in a distributed status. This provides such a direct and single data communicating network.

Apart from the shared communications within a network, a distributed network often shares the data transmission and processing. A hybrid network shown in Fig. 1.9 utilizes multiple-communication standards or protocols simultaneously; that means, a network is made up of equipment from multiple vendors. It also is understood that the network mixes more than one networking technologies together. The concept of multitier sensor network has arisen as a correspondence to the concept of single-tier sensor network.

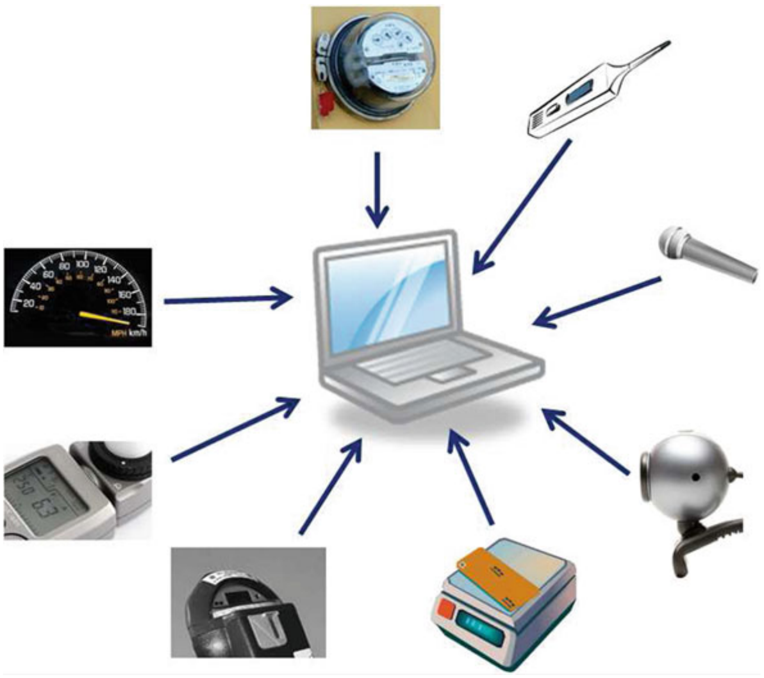


Fig. 1.7 Centralized connection architecture of surveillance sensors

Single-tier network is one that reaches every other network on the Internet without purchasing IP transit or paying settlements. In single-tier networks, a set of application-oriented requirements and tasks are accomplished by an appropriate sensor and embedded platform which has the capability to charge the entire network.

A multitier sensor network shown in Fig. 1.10 is a hierarchical network of sensors which are connected in the shape like a tree. With the inheritance of several levels of reliability and expenditure based on the type of sensor adopted for application tasks, multitier networks flexibly organize the construction of whole sensor network based on different goals and requirements. Centralized, distributed, and hybrid as well as multitier sensor networks are shown in the figures of this section.

The deployment of sensors has a great influence on performance and a vast cost of surveillance systems. Redundant sensors increase processing time and costs of installation. On the other hand, lack of sensors may cause blind regions which reduce the reliability of the monitoring system. Thus, it is true to simulate and deploy sensors beforehand so that the configuration covers the entire region with the minimum number of sensors and costs [15].

An example showing more advanced technique which involves intelligent object detection and camera self-calibration [4,20,21,41] is illustrated in Fig. 1.11. The cooperation between these cameras is implemented by using camera communications which are event-driven based; these cameras are able to transfer mutual

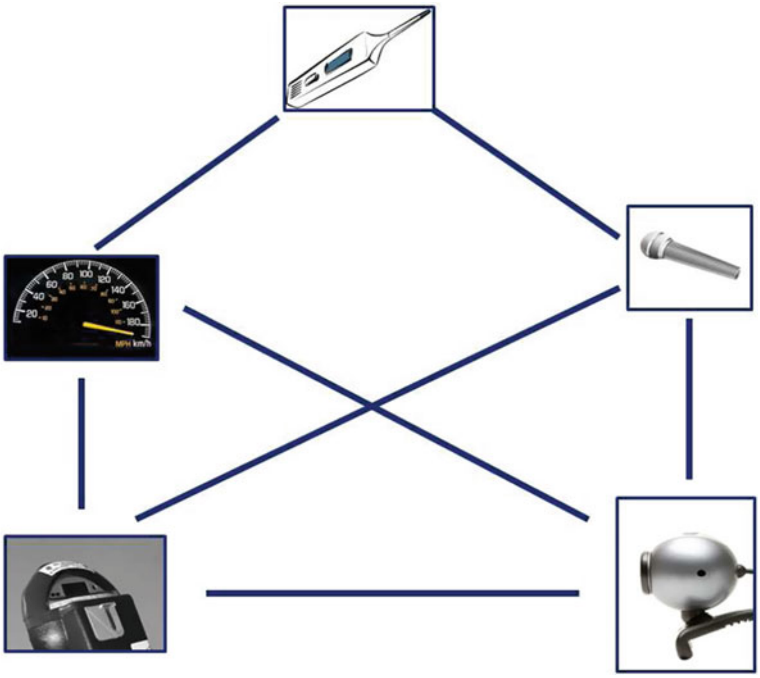


Fig. 1.8 Distributed connection architecture of surveillance sensors

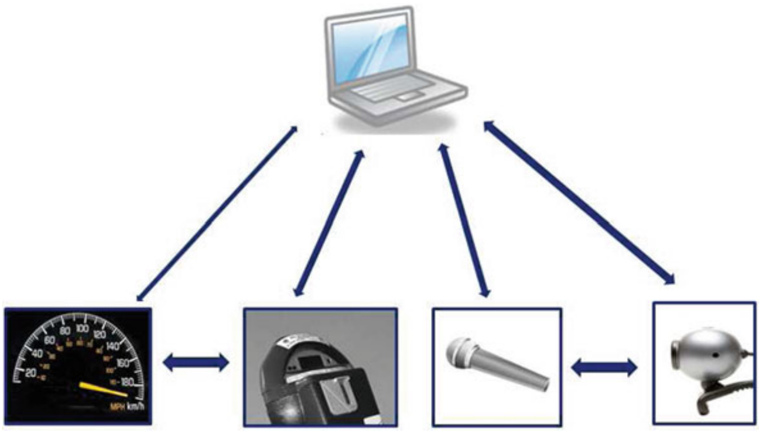


Fig. 1.9 Hybrid connection architecture of surveillance sensors

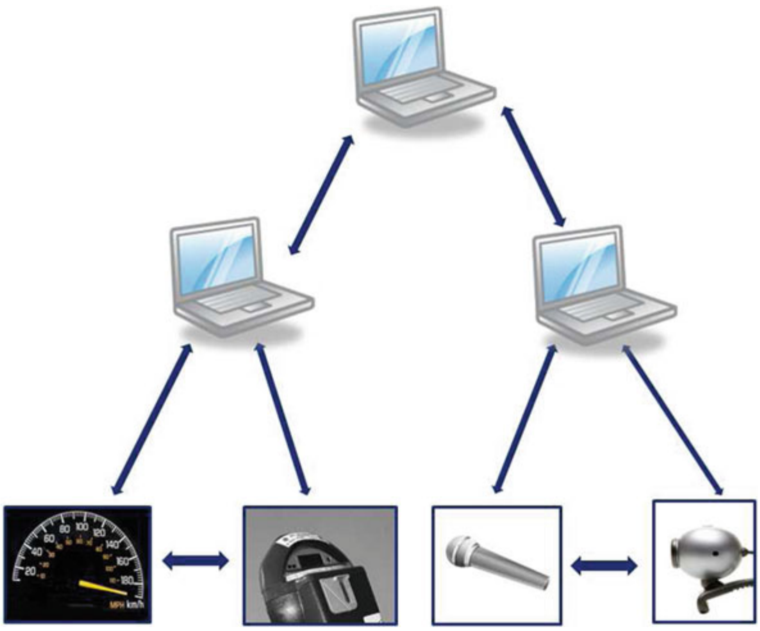


Fig. 1.10 Multitier connection architecture of surveillance sensors

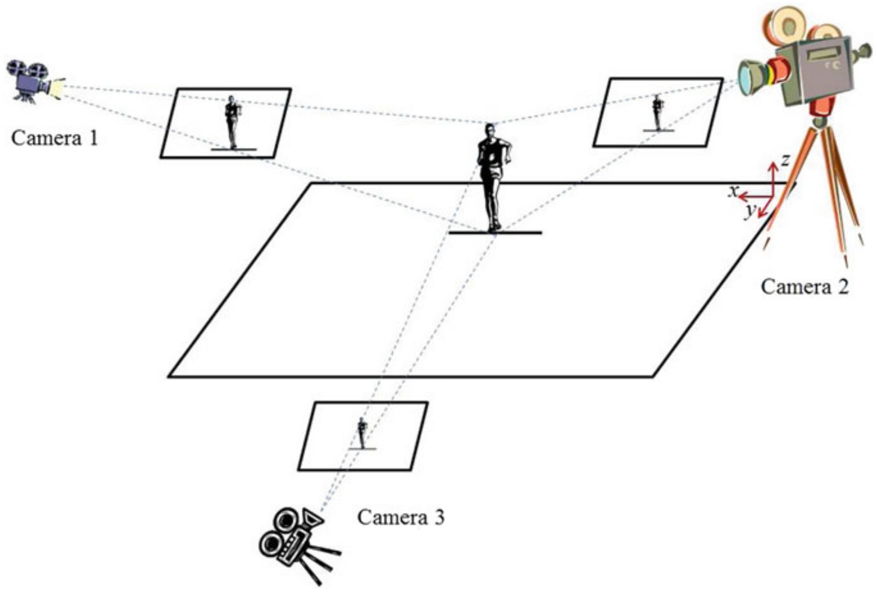


Fig. 1.11 Multicamera calibrations

messages and decide automatically what measurement should be taken to deal with the ongoing events. Automatic camera calibration and management methods are under development which addresses the camera network calibration. The objective is to remove, or at least reduce the amount of manual input required.

1.2.2 Surveillance Cameras

Surveillance cameras can be integrated into the devices like smartphones and cars which could be connected to central servers in wired or wireless way. There is a broadwide spectrum of camera lenses for surveillance cameras like wide-angle or fish-eye lenses. Therefore, a vast amount of geometric distortions from various cameras may be generated, the principle of a camera is shown in Fig. 1.12. The distortion removal is preprocessed in visual surveillance and is also one of the primary tasks in digital image processing.

There are two basic purposes for deploying surveillance cameras, namely investigation and deterrence. The data recorded by surveillance cameras will most often be used to review a crime commitment or unexpected accident or incident, so that a channel is provided for security staff to understand and search back what really happened there. However, the cameras themselves also have a deterrent value because people who know they are being watched usually performed on their best behaviors. In order to implement the maximum investigative and deterrent values of the cameras, it appears necessary to carefully choose where we place these cameras and which direction we point them in. Generally, the four locations where security cameras are often installed include entrances and exits, shopping customers' transaction points, targets, and secluded areas.

We provide a camera system installed on a school bus as the example shown in Fig. 1.13. In this bus, two cameras point in the front door and back door. Inside the

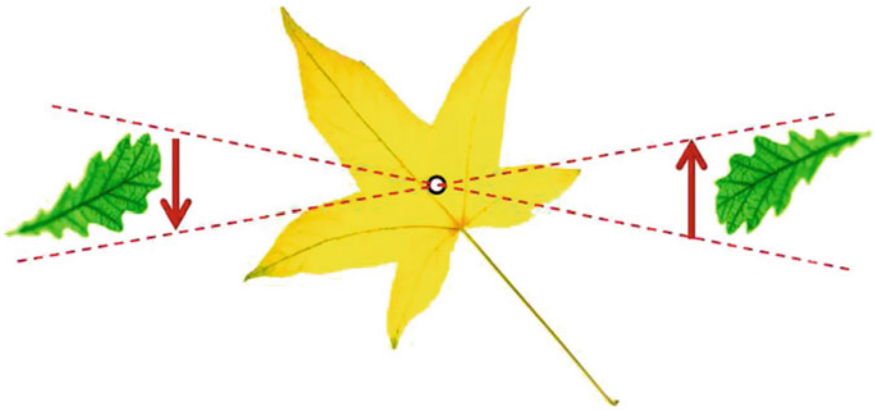


Fig. 1.12 Nature of a pinhole camera

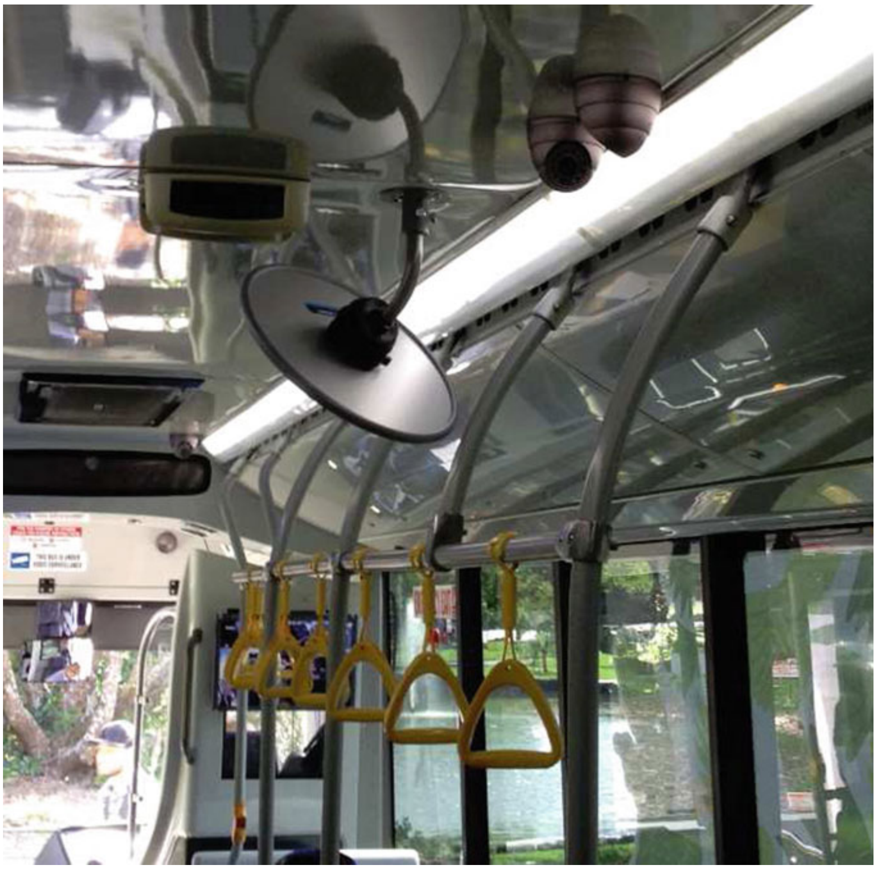


Fig. 1.13 Bus surveillance system with the deployment of multiple cameras

bus, multiple cameras are lodged and oriented to passengers; a screen is fixed to show the pictures sourced from all camera. Outside the bus, multiple cameras are monitoring the surroundings including in front, back, and lanes beside, providing road monitoring services as shown in Fig. 1.13.

Determining a digital camera's field of view (FoV) is complicated; the exact view can be various depending on the focal length of each camera. This means that a standard camera placed at a far corner of a 15×15-square-meter room will still capture the most area of the scene. It is discovered that the smaller the lens and the shorter the focal length, the wider the field of view (FoV). Cameras with longer focal length often perform better than those are not with so larger lens to capture images at a long distance. Many of today's cameras come with the ability to automatically adjust the lens in order to meet the needs of its users. This "varifocal " ability is a great feature when unseeing which focal length is required. Digital camera lens with fixed focal length are less expensive and generate less distortions.

The view of a single camera is finite and limited by scene structures. In order to monitor a wide region, such as tracking a vehicle through the road network of a city or analyzing the global activities happening in a large train station, videos taken from multiple cameras simultaneously have to be adopted. Wide-area surveillance control technique is based on large-scale data analysis and management for covering a broad territory in intelligent visual surveillance. It is a multidisciplinary field related to computer vision, pattern recognition, signal processing, communication, multimedia technology, etc. Wide-area surveillance control technique is related to multisensor control and cooperative camera technique [7].

There are often large changes of viewpoints, illumination conditions, and camera settings between views of the field. Matching the appearance of objects across camera views is difficult. The topology of an extensive camera network could be complicated, and scene deployment limits the FOV of cameras. Some camera views are disjointed and may cover multiple ground planes. These bring great challenges for camera calibration, inference, and links in topology.

In order to monitor a wide area with a small number of cameras and to acquire high-resolution images from optimal view, a number of surveillance systems employ both static cameras and active cameras, whose panning, tilting and zooming (PTZ) parameters are automatically and dynamically controlled [19]. Calibration, motion detection, object tracking, and behavior analysis with hybrid cameras face new challenges compared with only using static cameras.

1.2.3 Camera Calibration

Camera calibration originally refers to correct image distortions which are the phenomenon that lines of the image are curved due to the surface irregularities of camera lens when generating the images from sensors [40]. Geometric distortion is separated into two categories: internal distortion and external distortion. Internal distortion [29] results from the geometry of the lens (as radial distortion [38], projection distortion, and stepwise distortion); external distortion is owning to the shape distortion of the object; an example is shown in Fig. 1.14.

Image distortion detection is an essential problem in digital image processing. A vast majority of techniques have been applied to image distortion detection and cor-

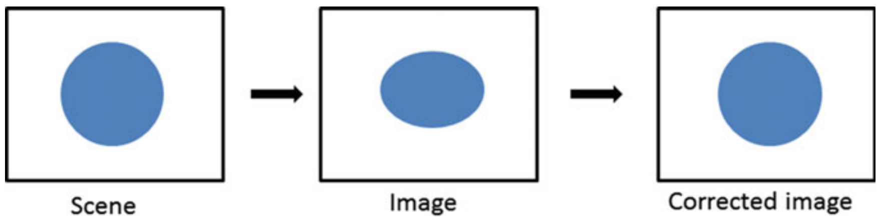


Fig. 1.14 Example of image correction using camera calibration

rection. The scale-invariant feature transform (SIFT) has been employed for detection; meanwhile, the theory of Kolmogorov complexity has been applied to normalized information distance [17]. According to general affine transformation, we have

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{R} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} + \mathbf{t} \tag{1.1}$$

where,

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{1.2}$$

$$\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{1.3}$$

Furthermore, we denote the homogeneous matrix $[\mathbf{R} \mid \mathbf{t}]$ as,

$$[\mathbf{R} \mid \mathbf{t}] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{1.4}$$

Mathematically, a perspective projection for simulating a camera projection takes use of Eq. (1.5) below:

$$\begin{cases} x' = \frac{x}{z} \\ y' = \frac{y}{z} \end{cases} \tag{1.5}$$

Generally,

$$s \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1.6}$$

Therefore, we simplify the equation as

$$s \cdot \mathbf{m} = A \cdot [\mathbf{R} \mid \mathbf{t}] \cdot \mathbf{M} \tag{1.7}$$

where $\mathbf{m} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \mathbf{M} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix},$

(X, Y, Z) is the coordinate of a 3D point in the world coordinate system, (u, v) are the coordinates of the projection point in the corresponding pixel, A is called a camera matrix or a matrix of intrinsic parameters, (c_x, c_y) is a principal point that is usually at the image center, (f_x, f_y) are the focal lengths, and the joint rotation–translation matrix $[\mathbf{R} \mid \mathbf{t}]$ is called a matrix of extrinsic parameters [3].

Along with camera modeling, the algorithm of image correction or camera calibration is taken part into the below steps [40]:

- *Corner extraction*: corners are the salient points in an image. A corner usually refers to the intersection between two or more edges. Computationally, corner points are associated with the Hessian matrix having the maximum eigenvalues. [17]
- *Point ordering*: detected corners will be sorted and ordered. This step is to prepare for the matching of two similar images.
- *Point correspondences*: the ordered corners need to search their matching points between two similar regions manually or automatically. Usually, a finite number of corners will be selected for matching purpose at the initial stage.
- *Bundle adjustments*: once a number of corners have been assigned to the corresponding matches, the bundle adjustment automatically corrects the remaining points to the right positions so as to align the calibration purpose.

For a single camera, the calibration is based on limited points in 3D space with coordinates corresponding to the pixels of an image. From these limited 3D points [27], all pixels of the image corresponding to all the points in the 3D space will be adjusted; it is called bundle adjustments.

For multicamera self-calibrations [10,22,25], we need set a scene first in use of a group of cameras to capture images of the fixed scene. From the captured images and the constraints between cameras and objects in this scene, we seek the corresponding coordinates of objects and cameras in the 3D Cartesian system. The method of least squares, a standard approach in regression analysis, is applied to find the approximate solution of overdetermined systems. If any cameras or objects in the scene have been slightly changed, the multicamera self-calibration system has the ability to find the extrinsic and intrinsic parameters using the method of least squares [1,23,33].

Camera calibration is a fundamental problem in computer vision which is indispensable in visual surveillance. There has been a huge number of literatures on calibrating cameras with respect to a 3D world coordinate system. Both the intrinsic parameters (such as focal length, principal point, skew coefficients, and distortion coefficients) and extrinsic parameters (such as the position of the camera center and the camera's orientation in world coordinates) of cameras are needed to be estimated. The process to find the intrinsic and extrinsic parameters varies in time-consuming, especially when the number of cameras is very substantial.

Estimating the intrinsic parameters is a prerequisite to a wide variety of machine vision tasks related to motion and stereo analysis [35]. Intrinsic parameters of an object contain aspect ratio, shift, rotation, zoom [2], skew, focal length, location, and so on. Without any information regarding the intrinsic parameters, a sequence of images can yield 3D structure up to an arbitrary projectivity of 3D space. In order to upgrade this projective reconstruction based on a Euclidean distance [34], the camera intrinsic calibration parameters are crucial.

Camera calibration is essential for multiple camera systems. Fusing and transferring information between views depends on the relationships among the cameras within the settings. Camera calibration manually is a time-consuming task. Whenever a camera is displaced, removed or added, it needs to be re-calibrated. In a synchronized network having multiple cameras, this becomes a significant hurdle to successfully deploy surveillance systems automatically. The system aims to detect and track changes and update the calibration information timely. If the views of two cameras have substantial overlapping, a homography between them can be computed during calibration time. A slew of approaches automatically select and match static features from 2D images to compute an assumed homography between two camera views and align multiple camera views to a single rectified plane. The chosen features are typically corner points, such as Harris corners and scale-invariant feature transform (SIFT) points. The matching needs to be robust to the variations of viewpoints:

- *Projective geometry*: shown in Fig. 1.11, the subject projective geometry is a branch of geometry dealing with the properties and invariants of geometric figures under projection. Projective geometry describes an object from the way as it appears to be. Projective geometry studies geometric subjects including lengths, angles and parallelism which become distorted when we look at specified objects. Further, projective geometry is also a mathematical model for how images of the 3D world are formed.
- *Kruppa equations*: Kruppa equation [18,36] maps the estimated projection matrices to Euclidean ones. By tracking a set of points among images of a rigid net, captured by a moving camera with constant intrinsic calibration parameters, the latter can be estimated by determining the image of an absolute conic which is a special one lying at the plane in infinity having the property that its image projection depends on the intrinsic parameters only [12,26]. This fact is expressed mathematically by using Kruppa equations.
  The matrix $K = \begin{bmatrix} k_1 & k_2 & k_3 \\ K_2 & k_4 & k_5 \\ k_3 & k_5 & 1 \end{bmatrix}$ is known as Kruppa coefficient matrix. By using the method of Cholesky factorization, one can obtain $k = DUV^*$. D is a lower triangular matrix with real and positive diagonal entries, U is a diagonal matrix and V^* denotes the conjugate transpose of V.
- *Motion constraints*: its purpose is to generate a specific instance of a motion through a low-dimensional task [30]. The main idea is to create a low-dimensional motion constraint from a sparse set of captured motion. Related subjects are translation, rotation, shaky, pan-tilt, radial lens distortion, critical motions sequence, and so on. Relevant motion constraint equation is therefore derived and is used to calculate optical flow with additional constraints [32].

- *Scene constraints*: typical scene constraints include vanishing points, orthogonal directions, rectified planes, and stereo-vision system [17,37].
- *Calibration techniques*: apart from the issues in self-calibration, techniques [24, 28] such as bundle adjustment, nonlinear minimization and online error estimations are also needed to be aware of.

1.3 Questions

Question 1. Why should we study the intelligent surveillance? What are advantages of a digital surveillance system? What's a CCTV system?
Question 2. What are the intrinsic and extrinsic parameters of a camera?
Question 3. When should we conduct camera calibration? What is multicamera self-calibration?
Question 4. What are the relationships between field of view (FoV) and focal length of a camera?
Question 5. What is the pipeline of surveillance systems?

References

1. Agapito D, Hayman E, Reid I (1998) Self-calibration of a rotating camera with varying intrinsic parameters. In: BMVC, pp 105–114
2. Agapito D, Hartley R, Hayman E (1999) Linear calibration of a rotating and zooming camera. In: IEEE CVPR, pp 15–21
3. Agapito D, Hayman E, Reid L (2001) Self-calibration of rotating and zooming camera. IJCV 45(2):107–127
4. Bougnoux S (1998) From projective to Euclidean space under any practical situation, a criticism of self-calibration. In: IEEE ICCV, pp 790–796
5. Calvel C, Ehrette T, Richard G (2005) Event detection for audio-based surveillance system. In: IEEE ICME, pp 1306–1309
6. Cavallaro A (2007) Privacy in video surveillance. IEEE Signal Process 24(2):168–169
7. Collins RT, Lipton AJ, Fujiyoshi H, Kanade T (2001) Algorithms for cooperative multi-sensor surveillance. Proc IEEE 89(10):1456–1475
8. Dhoble K, Nuntalid N, Indivery G, Kasabov N (2012) Online spatio-temporal pattern recognition with evolving spiking neural networks utilising address event representation, rank order, and temporal spike learning. In: IEEE world congress on computational intelligence, pp 554–560
9. Edwards C (2018) Deep learning hunts for signals among the noise: neural networks can deliver surprising, and sometimes unwanted, results. Commun ACM 61(6):13–14
10. Fraser C (1997) Digital camera self-calibration. J Photogramm Remote Sens 52(4):149–159
11. Hameete P, Leysen S, van der Laan T, Lefter I, Rothkrantz L (2012) Intelligent multi-camera video surveillance. Int J Inf Technol Secur 4(4):51–62

12. Hartley R, Hayman E, Agapito L, Reid I (1999) Camera calibration and the search for infinity. In: IEEE ICCV, pp 510–517
13. Huang T (2014) Surveillance video: the biggest big data. Comput Now 7(2)
14. Jain R, Kasturi R, Schunck B (1995) Machine vision. McGraw-Hill series in computer science. McGraw-Hill, New York
15. Kandhalu A, Rowe A, Rajkumar R (2009) DSPcam: a camera sensor system for surveillance networks. In: ACM/IEEE ICDSC, pp 1–7
16. Kieran D, Yan W (2010) A framework for an event driven video surveillance system. In: IEEE AVSS, pp 97–102
17. Klette R (2014) Concise computer vision. Springer, London
18. Lei C, Wu F, Hu Z, Tsui HT (2002) A new approach to solving Kruppa equations for camera self-calibration. In: ICPR, pp 308–311
19. Liebowitz D, Zisserman A (1999) Combining scene and auto-calibration constraints. In: ICCV, pp 293–300
20. Luong QT, Faugeras Q (1997) Self-calibration of a moving camera from point correspondences and fundamental matrices. IJCV 22(3):261–289
21. Lv F, Zhao T, Nevatia R (2002) Self-calibration of a camera from video of a walking human. In: IEEE ICPR, pp 562–567
22. Malis E, Cipolla R (2000) Self-calibration of zooming cameras observing an unknown planar structure. In: IEEE ICPR, pp 85–88
23. Malis E, Cipolla R (2000) Multi-view constraints between collineations: application to self-calibration from unknown planar structures. In: ECCV, pp 610–624
24. Maybank S, Faugeras OD (1992) A theory of self-calibration of a moving camera. IJCV 8(2):123–151
25. Mendelsohn J, Daniilidis K (1999) Constrained self-calibration. In: IEEE CVPR, pp 581–587
26. Mendonca PRS, Cipolla R (1999) A simple technique for self-calibration. In: IEEE CVPR, pp 500–505
27. Pollefeys M (1999) Self-calibration and metric 3D reconstruction from uncalibrated image sequences. PhD thesis, K. U. Leuven
28. Pollefeys M, Gool V (1999) Stratified self-calibration with the modulus constraint. IEEE PAMI 21(8):707–724
29. Pollefeys M, Khoch R, Gool V (1998) Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In: IEEE ICCV, pp 90–95
30. Rousso B, Shilat E (1998) Varying focal length self-calibration and pose estimation from two images. In: IEEE ICPR, pp 469–475
31. Saini M, Atrey P, Mehrota S, Kankanhalli M (2014) W3-privacy: understanding what, when and where inference channels in multi-camera surveillance video. Springer Multimed Tools Appl 68(1):135–158
32. Seo Y, Heyden A (2000) Auto-calibration from the orthogonality constraints. In: IEEE ICPR, pp 1067–1072
33. Seo Y, Hong K (1999) About self-calibration of a rotating and zooming camera: theory and practice. In: IEEE ICCV, pp 166–172
34. Sturm P (1997) Critical motion sequences for monocular self-calibration and uncalibrated Euclidean reconstruction. In: IEEE CVPR, pp 1100–1105
35. Sturm P (1999) Critical motion sequences for the self-calibration of cameras and stereo systems with variable focal length. In: BMVC, pp 63–72
36. Sturm P (2000) A case against Kruppa's equations for camera self-calibration. IEEE PAMI 22(10):1199–1204
37. Sturm P, Maybank S (1999) On plane based camera calibration: a general algorithm, singularities, applications. In: IEEE CVPR, pp 432–437

38. Tordoff B, Murray W (2000) Violating rotating camera geometry: the effect of radial distortion on self-calibration. In: IEEE ICPR, pp 1423–1427
39. Xu Z, Mei L, Liu Y, Hu C, Chen L (2016) Semantic enhanced cloud environment for surveillance data management using video structural description. Computing 98(1–2):35–54
40. Zhang Z (2000) A flexible new technique for camera calibration. IEEE PAMI 22(11):1330–1334
41. Zhang Z, Schenk V (1997) Self-maintaining camera calibration over time. In: IEEE CVPR, pp 231–236
42. Zheng K, Nand P, Yan W (2018) Video dynamics detection using deep neural networks. IEEE Trans Emerg Top Comput Intell 2(4):224–234

2 Surveillance Data Capturing and Compression

2.1 Data Capturing Using FSM

The theory of finite state machine (FSM) was originally from mathematics. In FSM, the state is uniquely determined by the current state and a single input. FSM has been employed to the applications that need to recognize or control the execution of a sequence of operations [16]. Most programmable devices contain controllers that are designed using FSM(s) [6,7,9,23,25]. FSMs could be found in almost all electronic products in home, office or a car, consumer electronics such as cellular telephones and CD players [8], even dishwashers and washing machines, etc.

The concept *state diagram* is commonly used in software engineering [13]. State diagram from unified modeling language (UML) in software engineering is a vital component used to describe FSM as a chart of the life-span of an object. *Life-span* refers when the object starts and ends; it is a special chart for content description. It has a start state and a finish state both at least. Connections between the states represent for those transitions that occur if the conditions are met. It is possible for various transitions of states to be produced between the time from the start state and the finish state. Furthermore, a state diagram also includes events, guard conditions, activities, and actions.

State is an abstraction of the attribute values and links of an object. The state reflects attributes and object links. The object defined in state could be either physical or semantic object. When the target object is a suspicious person in surveillance, state of the object could be human motion such as walking, sitting, standing, and sleeping. In a typical FSM, there exist various states during the life-span of state diagrams such as alive, asleep, and await which refer to state activated, state closed, and state ready to be processed, respectively.

Event refers to something that happens or starts at a point in time which is defined by the National Institute of Standards and Technology (NIST) US [4]. It only tells us when the transition starts and it is not used to discuss the whole process or parts

W. Q. Yan, *Introduction to Intelligent Surveillance*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-10713-0_2

of life-span. An event in a FSM is usually used to illustrate the action that happens before the transition between two states.

Activity is an operation that takes a period of time to be processed. During the processing time, an event is linked to its states.

Action is an operation performed before or after a state changes. Consequently, conditions normally trigger actions. The action occurs by entering into the new state. Each action represents a change in the process which is programmed (timer starting, counter increasing) or hardware based (actuators). There are normally two types of actions which are used to depict the different status of a state.

State diagram derives from software engineering. Each state in a state diagram has its own activity. The diagram helps us to understand a problem in software engineering, and it also helps us in design, implementation, and testing, especially for task analysis and task refactoring. In software engineering, a state diagram is associated with structural diagrams and behavioral diagrams. Structural diagrams include class diagram, object diagram, component diagram, deployment diagram, while behavioral diagrams encapsulate use case diagram, sequence diagram collaboration diagram, activity diagram, etc. [21].

A FSM is a model of behavior composed of a finite number of internal states (or simple states), the transitions between these states, the set of events that occur, and actions perform. The internal state describes status of the machine based on past events or actual inputs, thus reflecting history of what happened from the moment system starting up. Between two states, we must have activities or actions; a FSM diagram is different from the flowchart in programming.

Flowchart includes start and end points with ellipse shapes; between them, a diamond shape is used for condition selection (if $< condition >$, then $< structure >$), and a rectangle shape refers to an operation; a flowchart matches a logic order; a FSM is an abstract model of a machine with a primitive internal memory [12]; thus, a FSM must run with a memory and closely relate to the computer memory.

The FSM shown in Fig. 2.1 could be unfolded in a time series like HMM or RNN [11,15] as shown in Fig. 2.2.

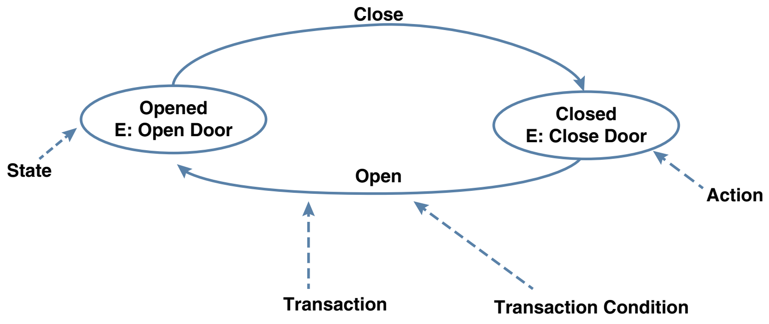


Fig. 2.1 States and transitions of the FSM of a door. Door state 1: open; door state 2: closed

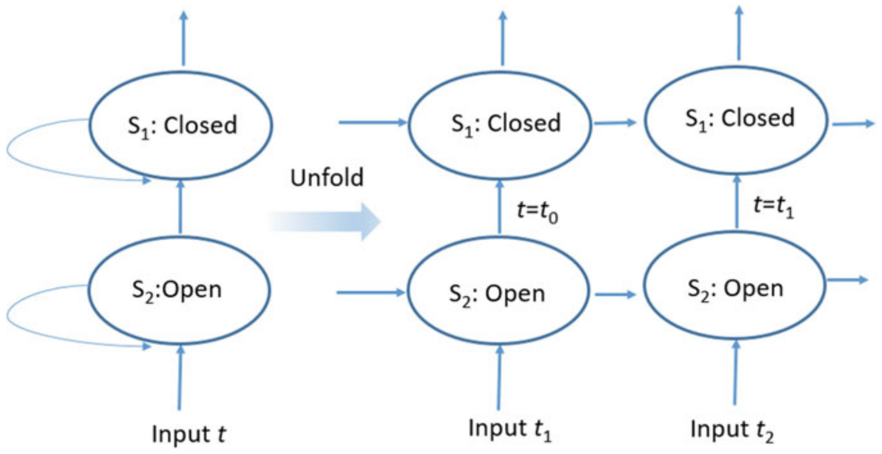


Fig. 2.2 Unfolded FSM of a door with time t_1 and t_2 as well as the states s_1 and s_2

Table 2.1 State transition table

Inputs	State s_1	State s_2	State $\cdots$
Input t_1	$\cdots$	$\cdots$	$\cdots$
Input t_2	$\cdots$	$<$ action $>$	$\cdots$
Input t_3	$\cdots$	$\cdots$	$\cdots$

A FSM definition including the full actions is presented by using Fig. 2.2. Correspondingly, we show the state transitions of this FSM in Table 2.1; this table is called as state transition table.

Given a sequence of inputs $I = \{i_1, i_2, \ldots, i_N\}$, the FSM generates a sequence of outputs $O = \{o_1, o_2, \ldots, o_M\}$ which are independent on the initial state $s_1 \in S$, $S = (s_1, s_2, \ldots, s_K)$, a transaction function $T(S, I)$ maps each current state and input to next state and an output function $O(I)$ maps each current state to an output; S is a finite and nonempty set of states.

A deterministic FSM [4] is a quintuple (I, S, s_1, T, F); I is the input alphabet (nonempty set of symbols), s_1 is an initial state, an element of S; T is the state transition function; F is the set of final states, a subset of S, $F \subset S$.

For two states s_1 and s_2 in a FSM, they are equivalent if and only if s_1 and s_2 generate the identical output sequence. For two FSMs are equivalent if and only if (i.f.f.) every input sequence $T(s_j, i)$ produces identical output sequences.

Given a FSM, we hope to find the simplest equivalent FSM with the minimum number of states using the equivalence existing in FSMs by marking each pair of states (s_1, s_2) as nonequivalent if they have different outputs. For each input i, we find and mark the state pair $(T(s_1, i), T(s_2, i))$; we compute the marks iteratively until no more marking is possible. After all the pairs of states are marked as nonequivalent, the

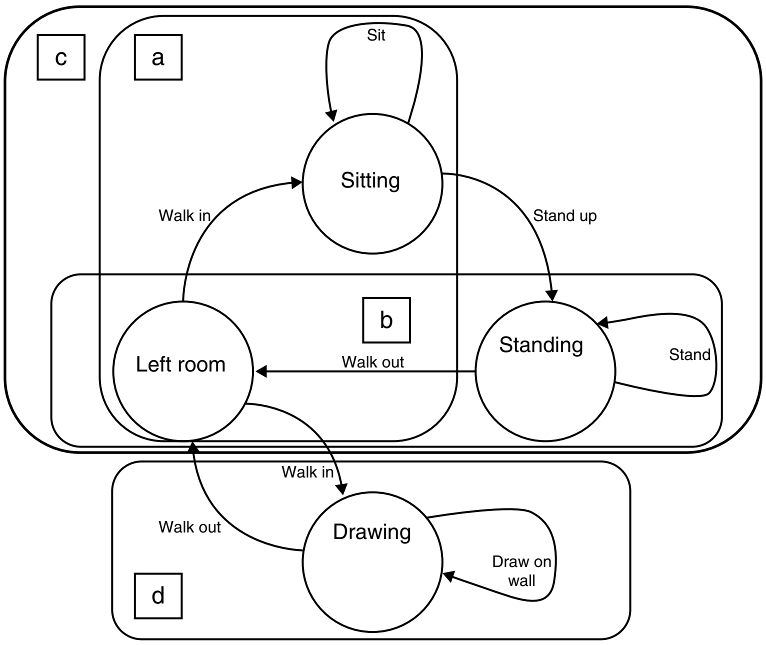


Fig. 2.3 A surveillance state diagram

unmarked state pairs are all equivalent. This procedure is called FSM minimization. Accordingly, the FSM has been simplified.

A typical example of the FSM is the states and transitions between them as shown in Fig. 2.1. The door has two states, namely open and closed. The two states could be transmitted between each other if somebody opens or closes the door. In this case, the object is the door which is to be opened or closed. The two states that are to be used in describing the status of door are opened and closed. The triggering condition which decides the states of the door is that someone opens the door or closes it. The action is the person starts opening the door or the person finishes closing the door.

For example, we have a surveillance state diagram shown in Fig. 2.3. This state diagram tells us the story what happens in this room and shows the events: a person coming in, sitting down, standing up, drawing, and walking out. After we make out the states, we could start the programming that is able to detect the events [13,22]. Specifically, the events described in the diagram include:

- Walk in and sit down event (a)
- Stand up and walk out event (b)
- Walk in, sit down, stand up, and walk out event (c)
- Draw on walls event (d).

Surveillance event is a semantic unit of sensor data. When surveillance data is being captured, we should think how to depict a semantic data in surveillance. Thus, we use the concept "event" as the semantic unit. Surveillance events bridge the gap between our physical world and the semantic cyberspace. The semantic concept refers to object description with meanings such as "tree" or "apple," "sitting," and "walking." [3]

In surveillance event computing, we need to investigate the typical "5W" questions: *who*, *when*, *where*, *what* and *why*. It just likes the fundamental elements of a piece of news from daily newspapers. In surveillance, if we need to present a story or an event, we must find out and define the content of the "5W" that are related to an event. Therefore, we need to answer the following questions in surveillance.

- Who participated in the event?
- What is the event?
- Where did the event take place?
- When did the event take place?
- Why did the event happen?

In surveillance, "who" is related to main object or role, biometrics provides assistance in solving this problem, such as face recognition, fingerprint, iris, gait, or pedestrian recognition or plat recognition of cars, buses, and trucks. "When" refers to date and time, namely time stamps; "where" is the location information from GPS, WiFi, Mac Address, or locating software; if we compose "when" and "where" together, we call it spatial–temporal relationship formally. "What" means the content description, usually relates to ontology; meanwhile, "why" refers to the reasons that are from reasoning and inference.

There is a typical example of state diagram in surveillance that controls the function of audio capturing. Within the FSM, there are only two states, namely recording and sleeping (Fig. 2.4). The events might occur in the FSM which contains the states recording and sleeping, activate an alarm from a sleeping, and inactivate the alarm during sleeping from recording.

The program starts from cleaning the buffer and flags; the four conditions are taken into consideration for audio capturing. If an audio is captured successfully, the waveform will be played back.

The pseudocode below shows the FSM implementation by using the script language of MATLAB. MATLAB is a powerful research tool that has been successfully employed to scientific simulations. MATLAB has the capability to run the scripts in recursive and parallel.

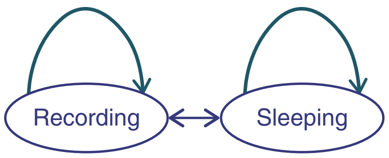


Fig. 2.4 A surveillance state diagram for audio capturing

The algorithm (1) based on MATLAB is used to record sounds. When we are talking, this program starts recording our voice; when stop, the computer starts writing our voice data into a wave file and then will play the audio waveform back. This program has the potential to help us and broadcast the audio footage through the Internet if we install an Apache HTTP server on our computer like the Skype. In this example, we capture audio clips using FSM, and the computer will save what we talked as a ".wav" file and play it. When we keep talking, the data will continue being stored into the running computer. The audio will be broadcasted when a HTTP server is set up and has been linked to the voice file with a streaming player.

```
Input  : Sound signals;
Output: Audio files;
Initialization;
Clean buffer and flags;
while < True > do
    if < no sound captured > && < the buffer is empty > then
        capture audio;
    end
    if < no sound captured > && < the buffer is not empty > then
        write the buffer out;
        clean buffer and flags;
        play the audio;
    end
    if < sound captured > && < the buffer is empty > then
        add data to the buffer;
        set flag;
        start recording;
    end
    if < sound captured > && < the buffer is not empty > then
        add data to the buffer;
        continue recording;
    end
end
```

Algorithm 1: A surveillance algorithm for audio capturing

In a general surveillance capturing system, flags, states, and transitions between these states, sleeping and recording are shown in Fig. 2.5. The different transitions will lead to distinct FSM actions.

The difficulty in FSM is how to find the threshold of audio and video signals; the typical way is to collect the data as a training dataset and seek the interval of the threshold value after several times of observations; then, an ideal threshold will be fixed by using tests. The feedback and evaluations based on the given threshold should be taken into consideration in further study.

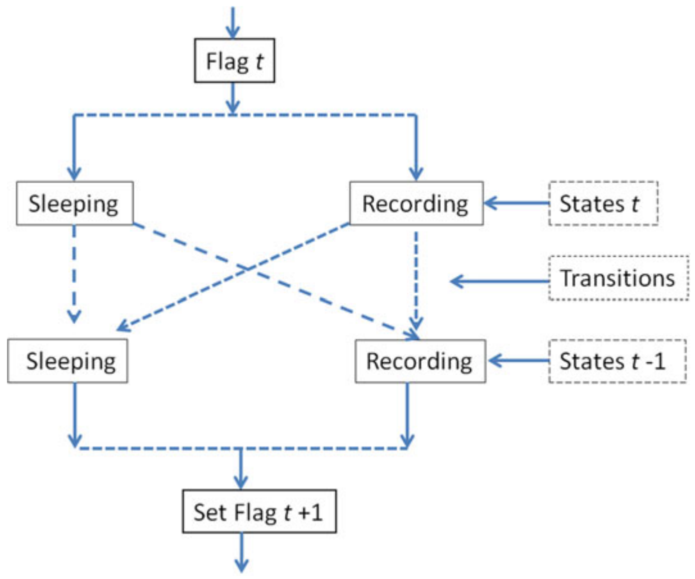


Fig. 2.5 Transitions between surveillance states

2.2 Data Capturing Using DNN

In recent years, deep neural networks (DNNs) have achieved a remarkable progression in solving many complex problems [26]. DNNs are suitable for dealing with the problems related to time series, such as speech recognition [17] and natural language processing. Video dynamics detection, as an instance, is time dependent. Apparently, video dynamics detection needs to utilize the present, previous, and next frames of a given video. If a frame change occurs, it triggers whether a video event happens or not.

Moving object detection is the basestone of object monitoring and tracking. The fast and accurate detection of moving objects has become a hot area in the field of video surveillance. According to the existing deep learning models, we now achieve high-precision and real-time video dynamics compared to the traditional methods.

Gated recurrent unit (GRU) is used to construct the recurrent neural network (RNN); the GRU cells are employed to declare a GRU unit. The output of convolutional layer is exported into a batch GRU vector; then, the feed function is used to get the output from RNN. Among them, the actual RNN model needs to be carried out according to the concrete steps. The output of RNN is treated as input of the fully connected neural network layer and the matrix multiplication by using the weight matrix to get the final output. After defining the DNN model, we also need to specify the loss function and training algorithm to train the model. We use cross entropy as the loss function of this model and the adaptive moment estimation (ADAM) algorithm as the optimization algorithm.

We observed that video dynamics detection based on deep learning is more resilient to external influence. Dynamic object segmentation, tracking, and event recognition could also be implemented based on DNN [18].

2.3 Surveillance Data Compression

After captured surveillance data [1], we need to find a space and deposit them. The surveillance data will be compressed because surveillance devices usually are operated in all day long and all year round. Various standards have been developed for the purpose of data compression [5], which are grouped into two categories:

- Still image compression. Each video frame is encoded independently as a static image.
  The most well-known standard of image compression is JPEG and JPEG 2000. JPEG stands for Joint Photographic Experts Group. Motion JPEG (M-JPEG) standard encodes each frame using the well-known JPEG algorithm.
- Motion estimation. Video frames are defined as a group of pictures (GoP) in which the first frame is encoded independently. For the other frames, only differences between the previous frames and itself are encoded. Typical standards are MPEG-2, MPEG-4 (H.263, H.264, H.265), MPEG-7, MPEG-21, etc. MPEG stands for Moving Picture Experts Group [10].
  Generally, there are three methods for the still image compression:
- *Discrete Cosine Transform* (DCT).
  DCT (DCT) is a lossy compression algorithm, and it has been successfully applied to JPEG, MPEG, H.26x, etc. DCT stands for a finite sequence of data points in terms of cosine functions $\cos(n \cdot \theta)$ oscillating at different frequencies; n is the frequency as shown in Fig. 2.6. DCT is important to a number of applications in engineering, from lossy compression of audio signals (e.g., MP3) to images (e.g., JPEG) where some high-frequency components usually are discarded.
  Figure 2.6 shows changes in the frequency of one of the trigonometric functions. Interestingly, the trigonometric functions such as $y = \sin(nx)$ and $y = \cos(nx)$, $x \in (-\infty, +\infty), n = 0, 1, 2, \ldots$, construct an orthogonal function system for digital signal processing.
  In particular, DCT is a Fourier-related transform which is very similar to discrete Fourier transform (DFT), but using only real numbers for energy compaction. The DCT is equivalent to DFT transform roughly twice of the length operating on real data with even symmetry.
  The discrete formula of 1D-DCT is denoted as,

$$y(k) = w(k) \sum_{n=1}^{N} x(n) \cos\left[\frac{\pi}{2N}(2n-1)\cdot(k-1)\right], k = 1, 2, \ldots, N \qquad (2.1)$$

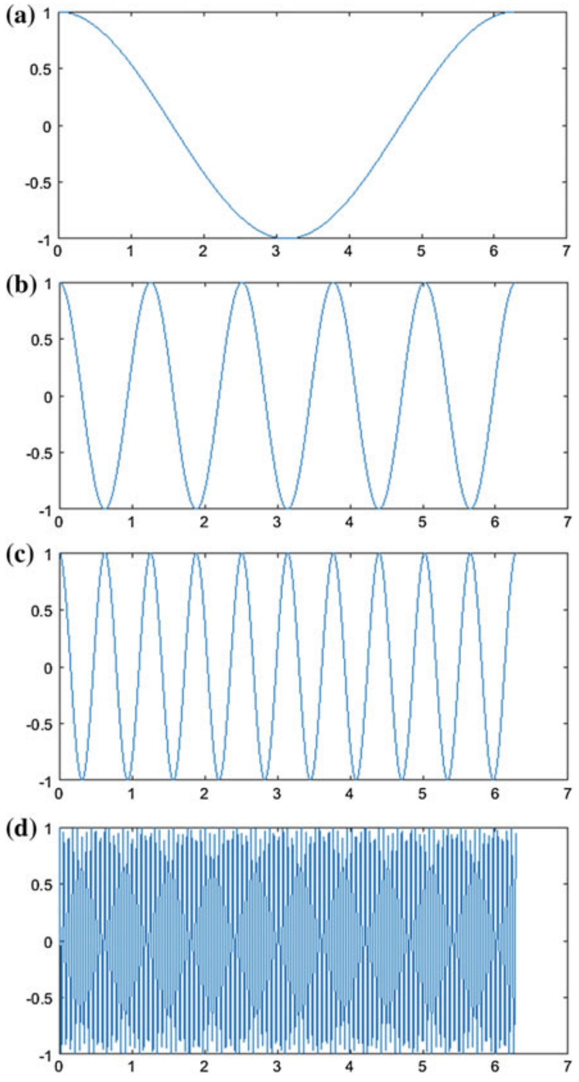


Fig. 2.6 Frequency changes of the function $y = \cos(nx)$, **a** $n = 1$ **b** $n = 5$ **c** $n = 10$ **d** $n = 100$

where N is the length of x, and x, y are the same size. The series are indexed from $n = 1$ and $k = 1$ instead of the usual $n = 0$ and $k = 0$ because the vectors run from 1 to N instead of from 0 to $N - 1$.

$$w(k) = \begin{cases} \frac{1}{\sqrt{N}}, k = 1; \\ \sqrt{\frac{2}{N}}, 2 \le k \le N; \end{cases} \tag{2.2}$$

1D-IDCT reconstructs the signals using the coefficients $x(k)$,

$$x(n) = \sum_{k=1}^{N} w(k)y(k)\cos\left[\frac{\pi(2n-1)(k-1)}{2N}\right] \tag{2.3}$$

- *Fractal Compression.*
  Fractal compression is performed by locating self-similar sections of an image, it is employed to compress images by using affine transformation which is a lossy compression method and reconstructs an approximation of the original image that could be accepted.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \alpha \cdot \cos\theta & -\sin\theta \\ \sin\theta & \beta \cdot \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{2.4}$$

  where α and β are scaling factors in x and y directions; Δx and Δy are the shift along x and y, respectively; θ is the rotation angle along the anti-clockwise direction.
  Affine transformation of an image is a combination of a rotation, scaling (or zooming), or translation (shift). Fractal image compression first segments an image into non-overlapping regions (they can be any size or shape). Then, a collection of possible regions are refined. The regions do not need overlap and cover the entire image, but must be larger than the domain regions. For each domain region, the algorithm searches for a matching region that very closely resembles the domain region when applied to an appropriate affine transformation. Afterward, a fractal image format (FIF) file is generated for the image. This file contains information on the choice of domain regions and the list of affine coefficients (i.e., the entries of the transformation matrix) of all associated affine transformations. So all the pixels in a given region are compressed into a small set of entries of the transformation matrix corresponding to a color value in integer between 0 and 255. This process is independent on resolution of the original image. The output graphics will look like the original one at any resolution, because the compressor has found an iterated function system (IFS) whose attractor replicates the original one.
- *Discrete Wavelet Transform* (DWT):
  DWT is a hierarchical representation of an image where each layer represents a frequency band. The less important layers will be discarded for lossy compression. Wavelets are functions which allow data analysis of visual signals or images according to scales or resolutions.
  The DWT represents an image as a sum of wavelet functions with different locations and scales that represent the data into a set of high-pass (detail) and low-pass (approximate) coefficients. In one-dimensional discrete wavelet transform (1D-DWT), the input data passes through a set of low-pass filters and high-pass filters. The output of high-pass and low-pass filters is usually downsampled by 2. The output from low-pass filter is an approximate coefficient and the output from the high-pass filter is a detailed coefficient. In the case of image compressing that is in two directions, both rows and columns, two-dimensional discrete wavelet trans-

form (2D-DWT) should be taken into account. The outputs are then downsampled in each direction.

$$x_l(n) = \sum_{k=0}^{K-1} x(Q \cdot n - k) \cdot g(k) \tag{2.5}$$

$$x_h(n) = \sum_{k=0}^{K-1} x(Q \cdot n - k) \cdot h(k) \tag{2.6}$$

$$x_{1,l}(m, n) = \sum_{k=0}^{K-1} x(m, Q \cdot n - k) \cdot g(k) \tag{2.7}$$

$$x_{1,ll}(m, n) = \sum_{k=0}^{K-1} x_{1,l}(Q \cdot m - k, n) \cdot g(k) \tag{2.8}$$

$$x_{1,h}(m, n) = \sum_{k=0}^{K-1} x(m, Q \cdot n - k) \cdot h(k) \tag{2.9}$$

$$x_{1,hh}(m, n) = \sum_{k=0}^{K-1} x_{1,h}(Q \cdot m - k, n) \cdot g(k) \tag{2.10}$$

where $g(k)$ represents low-pass filter while $h(k)$ means high-pass filter, Q stands for downsampling filter.

2.4 Digital Image Encoding and Decoding

Generally, the steps for encoding and decoding of digital images in JPEG include color space conversion, subsampling, DCT and IDCT transforms, quantization, RLE coding, etc.

The presentation of colors in an image is converted from RGB to YCbCr first as shown in Eq. (2.11).

$$\begin{cases} R = Y & & +1.402 \cdot (C_r - 128) \\ G = Y & -0.34414 \cdot (C_b - 128) & -0.71414 \cdot (C_r - 128) \\ B = Y & +1.772 \cdot (C_b - 128) & \end{cases} \tag{2.11}$$

However, RGB color space is not the best one. YCbCr and YUV color spaces are more effective than the RGB does. Meanwhile, YCbCr and YUV have been encoded well with adjustable proportions.

As well known, RGB space is nonlinear in perception when it is used to present color images. However, our human visual system (HVS) is based on lighting stimulation; therefore, luminance will play a pivotal role; thus, we hope to separate the luminance component (Y) out from the color information [14].

The resolution of chroma data is reduced usually by downsampling. Downsampling refers to reduce the size of an image block in half; consequently, it will greatly reduce the file size. Hence, the image is split into blocks comprising of 8×8 pixels. Each of the Y, Cb, and Cr components undergoes a discrete cosine transform (DCT). 2D-DCT is based on the decomposition using the sine or cosine function-based orthogonal systems.

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \cos\left(\frac{\pi(2m+1)p}{2M}\right) \cos\left(\frac{\pi(2n+1)q}{2N}\right) \tag{2.12}$$

where $0 \le p \le M - 1$, $0 \le q \le N - 1$, M and N are the row and column size, respectively. If we apply the DCT to image data in real numbers, the result is also real. The DCT tends to concentrate energy and select the important elements at the upper left corner of 8×8 blocks for digital image compression.

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, p = 0; \\ \sqrt{\frac{2}{M}}, 1 \le p \le M - 1; \end{cases}$$

$$\alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, q = 0; \\ \sqrt{\frac{2}{N}}, 1 \le q \le N - 1; \end{cases}$$

The 2D-IDCT is,

$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N} \tag{2.13}$$

where $0 \le m \le M - 1$ and $0 \le n \le N - 1$.

Subsequently, amplitudes of the frequency components are quantized so as to round the value of the DCT coefficients; the high-frequency components are discarded so as to achieve the purpose of image compression; however, the visual quality will not be changed too much, because our human visual system (HVS) could not percept any minor changes.

The resulting data for all 8×8 blocks is further compressed with a lossless algorithm such as a variant of Huffman encoding [5]. The compressed file will greatly reduce file size. Huffman coding is the process of finding and using a method for construction of minimum-redundancy codes, and it is a technique in entropy encoding, including lossless data compression. The algorithm output can be viewed as a variable-length code table for encoding a source symbol (such as a character in a file). Huffman's algorithm derives this table based on the estimated probability or frequency of occurrence (weight) for each possible value of the source symbol. As entropy encoding, symbols or alphabetical letters are generally represented using fewer bits than original symbols. The problem which Huffman coding attempts to solve is described as below.

Given an alphabetical set $A = \{a_1, a_2, \ldots, a_N\}$ (N is the length of set A) and a set $W = \{w_1, w_2, \ldots, w_N\}$ which represents the weights of all the elements in A, how to encode every element of A in binary code with regard to its weight in the whole set. The routine for Huffman coding is shown in Algorithm 2.

```
Input  : The coding symbols and their probabilities
Output: The coding tree
Initialization;
Create a leaf node for each symbol;
Add each symbol to the priority queue;
while < The queue of symbols is not empty > do
    (1) Remove the two nodes of highest priority (lowest probability) from the queue;
    (2) Create a new internal node with these two nodes as children and with probability
    equal to the sum of the two nodes' probabilities;
    (3) Add the new node to the queue;
end
The remaining node is the root node and the tree is complete;
```

Algorithm 2: Huffman coding algorithm

In summary, DCT transfers the energy of a 8×8 blocks to the upper left corner. This energy helps us encode the image in a very compact way. The decoding process reverses these steps.

In our real world, we have not only integer dimensions, such as 1, 2, and 3 dimensions, but also fractal dimensions such as the constants: $e = 2.718\cdots$, $\pi = 3, 1415926\ldots$, etc. Fractal theory was applied to digital image compression. The features of fractal theory include (Fig. 2.7),

- *Recursion*: Functions call themselves in programming nest with a condition, e.g.,
```
function F(float a){
if(a>1){
a: =a/2;
F(a);
}
}
```
- *Self-similarity*: The fractal image is quite similar to itself in details but with different size (zooming) or rotation angles; an example is shown in Fig. 2.7.

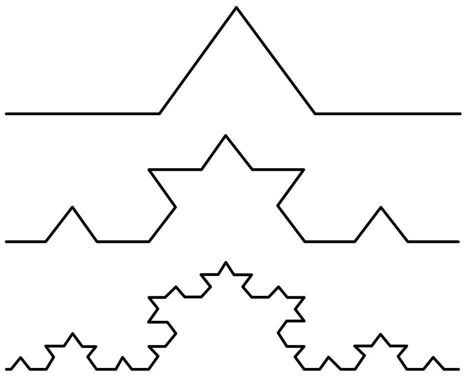


Fig. 2.7 Similarity of fractal curves

- *Iterated function system (IFS)*: Fixed-point iteration in mathematics typically is,

$$x_{n+1} = \alpha \cdot f(x_n) + \beta, n = 1, 2, \ldots \tag{2.14}$$

  where $x^* = \lim_{n \to \infty} x_n$, x^* is the fixed point, α and β are constants, f is the iterated function ($|\alpha| < 1$).
- *Photocopy machine algorithm*: The algorithm is used for the purpose of duplication based on the mechanism of iteration and IFS system. An example is shown in Fig. 2.8 for this photocopy machine algorithm using affine transformations.

In JPEG 2000, fractal theory has been employed for image compression. The similar blocks using affine transformation have been selected as codebook; afterward, the affine coefficients are stored; eventually, digital images are encoded using this codebook with the coefficients of affine transformations.

Wavelet transform is a kind of comb filtering. The visual signals usually are decomposed into various frequencies by using wavelet orthogonal functions. Those important information is included in low-frequency portion, and less important information is contained in high-frequency part. After finite times of filtering like a comb, an image will be progressively decomposed. The layered visual signals are employed to reconstruct the original image according to the requirements.

Usually, an image is progressively displayed while the further low-frequency details are added into the basic information for the purpose of displays. For image compression, images are segmented and stored in hierarchical structure; when the details are added into reconstructing process progressively, the final image will be reconstructed while unnecessary high-frequency details are discarded. The image compression is subject to the acceptance of image quality of human visual system. An example of wavelet-based image decomposition is shown in Fig. 2.9, and the reconstructed image is shown in Fig. 2.10.

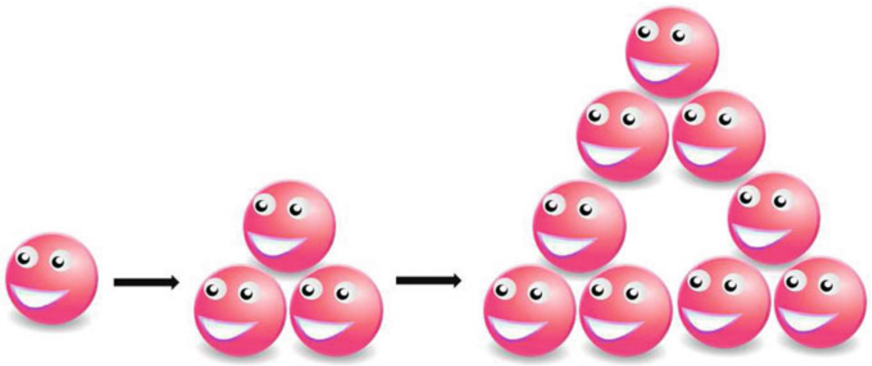


Fig. 2.8 Smiley faces are used as an example of the photocopy machine algorithm

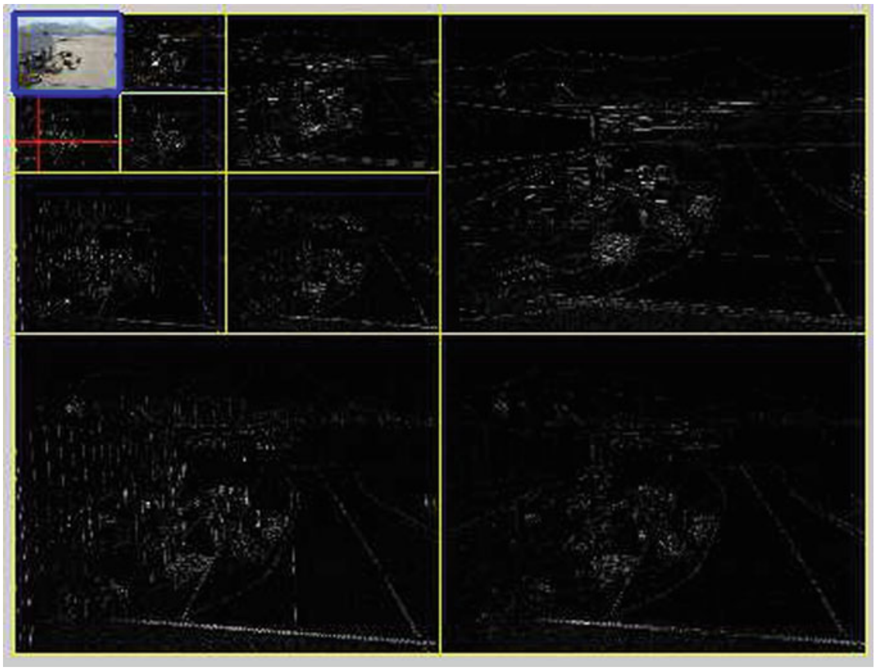


Fig. 2.9 Decomposing an image using Haar wavelet at level three

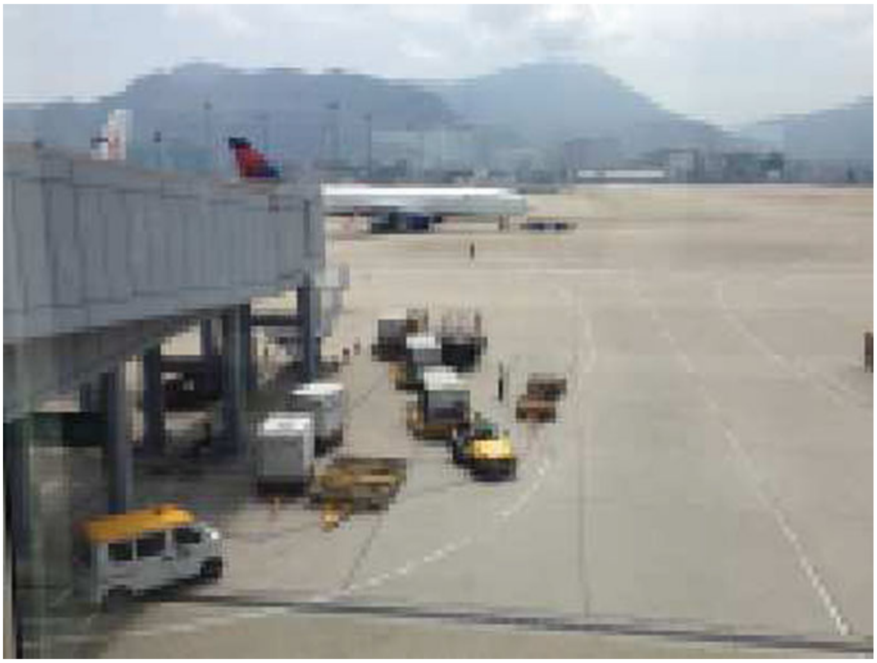


Fig. 2.10 Reconstructed image using wavelet components partially

2.5 Digital Video Encoding and Decoding

In famous YouTube Web site, most of the videos are compressed using MPEG-4 standard, such as MP4 and FLV. MPEG stands for Moving Picture Experts Group; the brief history of the MPEG video compressions is listed below [24].

- MPEG-1 (1992). This video compression scheme was designed for VCD players, and the data stream is up to 1.5 Mbit/s. The level 3 of MPEG-1 has been developed as the well-known MP3 format for audio compression.
- MPEG-2 (1994). This video compression standard was for digital broadcast television, and the bit rates are between 1.5 and 15 Mbit/s.
- MPEG-4 (1999, 23 Parts). This widely adopted video compression is for video adaptive coding called advanced video coding (AVC) which was designed for object-oriented composite files; the block size is variant from 2×2, and 4×4 to 8×8, even more larger, etc.
- MPEG-7 (2001, 10 Parts). The compression standard is for event-based multimedia content description which uses XML to store metadata in order to tag particular events. The standard includes information on content manipulation, filtering, and personalization.

- MPEG-21 (2002, 9 Parts): This standard is for sharing digital rights/permissions/ restrictions of digital content. It will be used for the purpose of copyright protection of digital media [2].

MPEG-1 has been offered for VCD player in the industry of recreation and entertainment; however, the resolution is very low; meanwhile, MPEG-2 was designed for DVD player; afterward, MPEG-4 is a very popular video standard that is widely used in today's videophone, the Internet and mobile multimedia communications such as H.26x, even today's high dimension (HD) and 4K TV. MPEG-7 is designed for content-based event encoding that will be used for semantic description, especially for further content-based search, retrieval, mining, and reasoning; the further video standard MPEG-21 is being designed for multimedia copyright protection [2]. In MPEG, we have two very important concepts: motion vectors (block shifting in 2D) and group of pictures (GOP); they are explained as,

- *I-Frame* refers to intra-coded picture/frame.
- *B-Frame* refers to bidirectionally predicted picture/frame.
- *P-Frame* refers to inter-coded picture/forwarded prediction frame.
- *Motion vector* refers to the corresponding position of a macroblock in another picture.

As shown in Fig. 2.11, we start from I-frames; after interpolating between frames, the P-frames are obtained. Using motion vectors and group of pictures (GOP)), we extrapolate the B-frame. After all decompressed frames are sorted in the display buffer in right order, a MPEG video only could be played. In the MPEG family, all videos have these I-frame, B-frame, and P-frame in order to be played correctly.

H.26x standard was designed for the purpose of multimedia communications such as videophone conversation; now, this MPEG compression family has been developed for the usages of the Internet, mobile phones, and HD TV display.

- H.261 (1990) was designed for dual communication over ISDN lines and supports data rate 64 Kbit/s. The scheme is based on DCT and uses intraframe (I-Frame coding) and interframe (P-Frame coding) compressions, utilized primarily in older

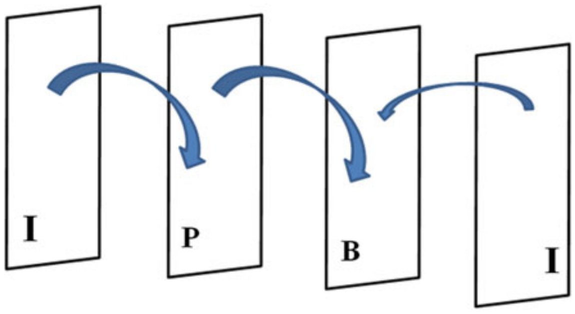


Fig. 2.11 Relationships between I-, B-, P-frames of MPEG videos

videoconferencing and video telephony products. H.261 was the first real digital video compression standard. Essentially, all designs of subsequent standard video codec are based on it. H.261 includes YC_bC_r color representation, the 4:2:0 sampling format, and 8-bit sample precision support progressive scan video.
- H.262 (1994) was designed by using ITU-T Video Coding Experts Group (VCEG) and are maintained jointly with the ISO/IEC Moving Picture Experts Group (MPEG); it is identical in content to the video part of the ISO/IEC MPEG-2 standard.
- H.263 (1996) is used primarily for videoconferencing, video telephony, and Internet video. It represents a significant step forward in standardized compression capability for progressive scan video, especially for the videos at low bit rates. H.263 was developed as an evolutionary improvement based on the obtained experience from H.261, the previous ITU-T standard for video compression, as well as the MPEG-1 and MPEG-2 standards. H.263 is a video compression standard originally designed as a low bit rate compressed format for video conferences which is a required video codec for IP multimedia subsystem (IMS), multimedia messaging service (MMS), and transparent end-to-end packet-switched streaming service (PSS).
- H.264 (2003)
  The ITU-T H.264 standard and the ISO/IEC MPEG-4 Part 10 or advanced video coding (AVC) standard are jointly maintained so that they have identical technical content [19,20].
  H.264 is also known as MPEG-4 which was developed for the use in high-definition systems such as HDTV and HD DVD as well as low-resolution portable devices. H.264 offers better quality at lower file size than both MPEG-2 and MPEG-4 ASP (DivX or XviD).
- H.265 (2013)
  H.265 is the high efficiency video coding (HEVC), scales from 320×240 pixels to 7680×4320, which is used for high dimension (HD) and 4K display technology. H.265 requires substantially more computational power to decode than H.264. Repeated quality comparison tests have demonstrated that H.265 reduces file size roughly 39–44% at the same quality compared to H.264.

Our media players need the software codec so as to play a video. The codec is a software embedded in operating systems for video playing; our players usually use these codecs for video playing. The often used codecs are listed as below:

- **Dirac** (BBC, 2004) is a prototype algorithm for encoding and decoding raw videos and video transmission over the Internet. The aim is to decode standard digital PAL TV definition in real time.
- **Indeo** (Inter, 1992) Distinguished by being one of the first codecs, it allows full-speed video playback without using hardware acceleration.
- **Indeo 5** decoders exist for Microsoft Windows, Mac OS Classic, Unix, etc.

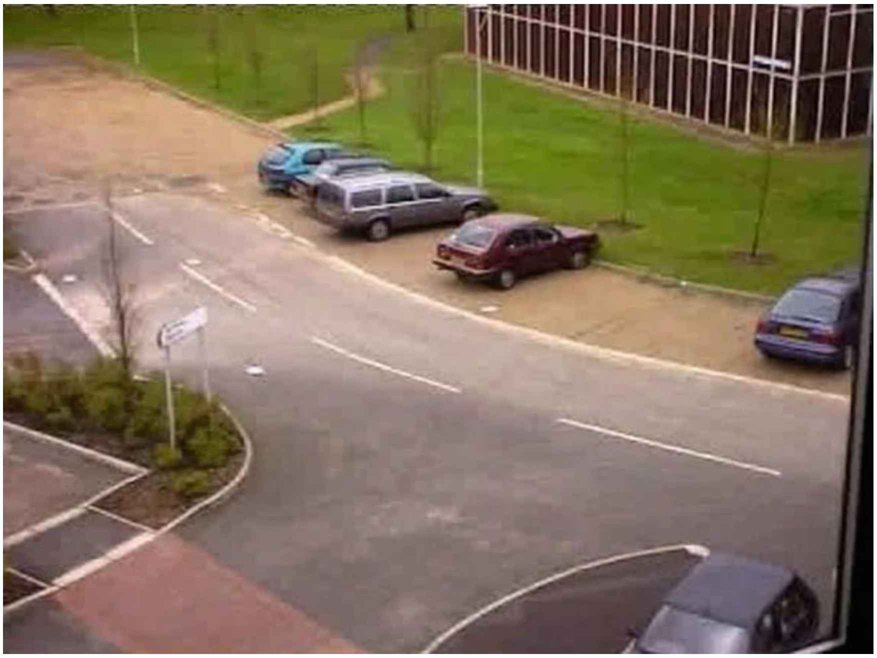


Fig. 2.12 A frame of PETS video frame for compression test

- **Motion JPEG (M-JPEG)** Each video frame or interlaced field of a digital video sequence is separately compressed as a JPEG image (DCT based). The relationship between two adjacent video frames has been broken.
- **M-JPEG** is used by IP-based video cameras via HTTP streams.
- **RealVideo** (RealNetworks, 1997) is a streaming media format and the network design based.
- **WMV** (Microsoft) was designed for Internet streaming applications as a competitor to RealVideo.
- **DivX** (Digital Video Express) uses lossy MPEG-4 Part 2 compression or MPEG-4 ASP.

An exemplar video from the famous video dataset of performance evaluation of tracking and surveillance (PETS) is shown in Fig. 2.12; the results for comparing video compression using different codecs are shown in Table 2.2; we applied various compression approaches to the same video and obtained the results after the compressions.

Table 2.2 An exemplar video frame from the PETS dataset

Video format	File size (MB)
MP4	6.7
AVI	5.461
H.264	1.523
ASF	1.447
FLV	1.682
WMV	1.447

2.6 Questions

Question 1. What is the finite state machine (FSM)? How to perform FSM simplification? What's the relationship between FSM and HMM?
Question 2. Please explain the below concepts in FSM.
(1) Action
(2) Activity
(3) Event
(4) Object
(5) State
Question 3. Draw state diagram of an automated sound recording system using FSM and explain it.
Question 4. Draw state transmission table of an automated sound recording system using FSM. What are the relationships between FSM and HMM?
Question 5. Draw flowchart of an automated sound recording system using FSM.
Question 6. Write pseudocodes (input, output, and procedures) of an automated sound recording system using FSM.
Question 7. Please explain the steps of JPEG image compression.
Question 8. Please explain the below concepts in video compression.
(1) MPEG-1
(2) MPEG-2
(3) MPEG-21
(4) MPEG-4
(5) MPEG-7
Question 9. What are the differences between H.265 and H.264?
Question 10. Please explain the below concepts in a MPEG video.
(1) I-Frame
(2) B-Frame
(3) P-Frame
(4) Motion vector
(5) GoP
(6) Quantization

Question 11. Please explain the concept fractal image compression.
Question 12. How to determine a threshold based on our observations?

References

1. Akazawa Y, Okada Y, Niijima K (2002) Real-time video based motion captured system based on color and edge distributions. In: IEEE ICME, pp 333–336
2. Anderson R (2008) Security engineering. Wiley, Indianapolis
3. Bhuyan MK (2012) FSM-based recognition of dynamic hand gestures via gesture summarization using key video object planes. Int J Comput Commun Eng 6:248–259
4. Black P (2008) Finite state machine. Dictionary of algorithms and data structures. U.S. National Institute of Standards and Technology
5. Cover T, Thomas J (2006) Elements of information theory, 2nd edn. Wiley, Hoboken
6. Devadze S, Sudnitson A (2008) Software environment for synthesis of testable FSM through decomposition. In: Microelectronics, pp 433–436
7. Djordjevic L, Stankovic R, Stojcev K (2005) Concurrent error detection in FSMs using transition checking technique. In: International conference on telecommunications in modern satellite, cable and broadcasting services, pp 61–64
8. Doi T, Itoh T, Ogawa H (1979) A long-play digital audio disk system. JAES 27(12):975–981
9. Drineas P, Makris Y (2002) Non-intrusive design of concurrently self-testable FSMs. In: Asian test symposium, pp 33–38
10. Dufaux F, Ebrahimi T (2008) H.264/AVC video scrambling for privacy protection. In: IEEE ICIP, pp 1688–1691
11. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Boston
12. Gurevich Y (2000) Sequential abstract state machines capture sequential algorithms. ACM Trans Comput Log 1(1):77–111
13. Kieran D, Yan W (2010) A framework for an event driven video surveillance system. In: IEEE AVSS, pp 97–102
14. Klette R (2014) Concise computer vision. Springer, London
15. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444
16. Noorit N, Suvonvorn N (2014) Human activity recognition from basic actions using finite state machine. In: International conference on advanced data and information engineering. Springer, pp 379–386
17. Ravanelli M, Brakel P, Omologo M, Bengio Y (2018) Light gated recurrent units for speech recognition. IEEE Trans Emerg Top Comput Intell 2(2):92–102
18. Siam M, Valipour S, Jagersand M, Ray N (2017) Convolutional gated recurrent networks for video segmentation. In: IEEE international conference on image processing (ICIP)
19. Schwarz H, Marpe D, Wiegand T (2007) Overview of the scalable video coding extension of the H.264/AVC Standard. IEEE Trans Circuits Syst Video Technol 17(9):1103–1120
20. Simone D, Naccari M, Tagliasacchi M, Dufaux F, Tubaro S, Ebrahimi T (2011) Subjective quality assessment of H.264/AVC video streaming with packet losses. EURASIP J Image Video Process 2:1–12
21. Sommerville I (2010) Software engineering, 9th edn. Pearson. ISBN-13:978-013703515
22. Trinh H (2011) Detecting human activities in retail surveillance using hierarchical finite state machine. In: IEEE ICASSP. Czech Republic
23. Venkataraman G, Reddy M, Pomeranz I (2003) GALLOP: genetic algorithm based low power FSM synthesis by simultaneous partitioning and state assignment. In: VLSI design, pp 533–538

24. Woods J (2012) Multidimentional signal, image, and video processing and coding. Elsevier Inc, Massachusetts
25. Zeng C, Saxena N, McCluskey E (1999) Finite state machine synthesis with concurrent error detection. In: International test conference
26. Zheng K, Nand P, Yan W (2018) Video dynamic capturing using deep learning. IEEE Trans Emerg Technol Comput Intell 2(1):1–11

3 Surveillance Data Secure Transmissions

3.1 Introduction to Network Communications

3.1.1 Computer Network

As well known, seven layers of a computer network provide a variety of services, and protocols [1] need various facilities to support the network so as to carry out the designed functionalities [12, 13].

- **Layer 1: Physical Layer**

In computer network, physical layer comprises of network hardware for data transmission. It is a fundamental layer underlying logical data structures of the high-level functions. Due to the diversity of available network technologies with broadly varying characteristics, this is perhaps the most complex layer in network architecture.

The physical layer defines means of transmitting bits rather than logical data packets over physical network. The bit stream may be presented by using coded words or symbols that are transmitted over a transmission medium. The physical layer provides a procedural interface to the transmission medium. The frequencies to broadcast on, the modulation scheme to use, and similar low-level parameters are specified in this layer.

Within semantics of network architecture, the physical layer translates logical communication requests from the data link layer to hardware-specific operations so as to affect transmission or reception of signals.

- **Layer 2: Data Link Layer**

Data link layer transfers data between adjacent network nodes in a wide area network (WAN) or between nodes on the same local network. The data link layer provides

W. Q. Yan, *Introduction to Intelligent Surveillance*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-10713-0_3

functional and procedural means to transfer data between network entities and might provide the means to detect and possibly correct errors that may occur in the physical layer. Examples of data link protocols are Ethernet for local networks (multinode), Point-to-Point Protocol (P2P) connections [1,13].

The data link layer is concerned with local delivery of frames between devices on the same LAN [13]. Data link frames, as these protocol data units are called, do not cross the boundaries of a local network. The data link layer is analogous to a neighborhood traffic which endeavors to arbitrate between parties contending for access to a medium without concern for their ultimate destination.

Data link protocols specify how devices detect and recover from such collisions and may provide mechanisms to reduce or prevent them. This layer takes effects through using network bridges and switches.

- **Layer 3: Network Layer**

Network layer provides functional and procedural means of transferring variable-length data sequences from a source to a destination host through one or more networks as well as maintains the quality of service functions.

The functions of this network layer include:
(1) Connection model: This layer works in connectionless communication way. For example, a datagram travels from a sender to a recipient who does not have to send any acknowledgement. Connection-oriented protocols also exist at other higher layers of the network model.
(2) Host addressing: Every host in the network must have a unique address that determines where it is. This address is normally assigned from a hierarchical system. On the Internet, the addresses are known as Internet Protocol (IP) addresses.
(3) Message forwarding: Since computer networks are partitioned into subnetworks and connected to others for wide area communications, networks use specialized hosts, called gateways or routers, to forward packets between networks. This is also of interest to mobile applications, where a user may move from one location to another. Version 4 of the Internet Protocol (IPv4) was not designed with this feature in mind though mobility extensions exist. IPv6 has a better designed solution.

Within the service layering semantics of the (OSI) Open Systems Interconnection) network architecture, the network layer responds to service requests from the transport layer and service requests to the data link layer.

- **Layer 4: Transport Layer**

In computer network, transport layer provides end-to-end or host-to-host communication services within a layered architecture of network components and protocols [1]. The transport layer provides services such as connection-oriented data stream support, reliability, flow control [5], and multiplexing.

Transport layers are implemented by using TCP/IP model which is the foundation of the Internet and the Open Systems Interconnection (OSI) model of general networking [20].

The best-known transport protocol is the transmission control protocol (TCP) [13]. It is used for connection-oriented transmissions, whereas the connectionless user datagram protocol (UDP) is used for simpler messaging transmissions. The TCP is more complex due to its design incorporating reliable transmission and data stream services. Other prominent protocols in this group are the datagram congestion control protocol (DCCP) and the stream control transmission protocol (SCTP).

Transport layer services are conveyed to an application via a programming interface to the transport layer protocols. The services may include the following features:

- *Connection-oriented communication*. It is normally easier for an application to interpret a connection as a data stream rather than to deal with the underlying connectionless models, such as the datagram model of the user datagram protocol (UDP) and that of the Internet Protocol (IP).
- *Same order delivery*. The network layer does not guarantee that packets of data will arrive in the same order that they were sent, but often this is a desirable feature. This is usually done through use of segment numbering, with which the receiver passes them to the application in order. This may cause head-of-line blocking.
- *Reliability*. Packets may be lost during transport due to network congestion and errors. By means of an error detection code, such as a checksum, the transport protocol may check that the data is not corrupted, and verify correct receipt by sending an ACK or NACK message to the sender. Automatic repeat request schemes may be used to retransmit lost or corrupted data.
- *Flow control*. The rate of data transmission between two nodes must be managed to prevent a fast sender from transmitting more data that need to be supported by using the receiving data buffer, may cause a buffer overrun.
- *Congestion avoidance*. Congestion control [5] refrains traffic entry into a telecommunication network, so as to avoid congestive collapse by controlling over subscription of any processing, such as reducing the rate of sending packets.
- *Multiplexing*. The ports provide multiple end points on a single node. Computer applications will keep listening on their own ports, which enable the use of more than one network service simultaneously. It is a part of the transport layer in TCP/IP [20].

- **Layer 5: Session Layer**

Session layer provides the mechanism for opening, closing, and managing a session between end-user applications, i.e., a semi-permanent dialog. Communication sessions consist of requests and responses that occur among applications. Session layer services are generally used in the environments that make use of remote procedure calls (RPCs).

Within service layering semantics of the OSI network architecture, the session layer responds to service requests from the presentation layer and issues service requests to the transport layer.

The session layer of the OSI model is responsible for session checking and recovery which allows information of different streams originating from different sources to be properly combined or synchronized.

An exemplar application is web conferencing, in which the streams of audio and video must be synchronous. Flow control ensures that the person displayed on screen is the current speaker. Another application is in live TV programs, where streams of audio and video need to be seamlessly merged and transitioned from one to the other so as to avoid silent airtime or excessive overlap.

- **Layer 6: Presentation Layer**

Presentation layer is responsible for delivering and formatting of information to the application layer for further processing or display. It believes the application layer regarding syntactical differences in data representation is within the end-user systems.

The presentation layer is the lowest layer at which application programmers should consider data structure and presentation, instead of simply sending data in the form of datagrams or packets between hosts. This layer deals with issues of string representation. The idea is that the application layer should be able to point at the data to be moved, and the presentation layer will deal with the rest.

Serialization of complex data structures into flat byte-strings by using mechanisms such as XML is thought as the key functionality of the presentation layer. Network encryption is typically conducted at this layer despite it can be completed at the application, session, transport, or network layers, which has its own advantages.

Network decryption is also handled at this layer. For example, when logged on to bank sites, the presentation layer will decrypt the data as it is received. Another example is the representing structure, which is normally standardized at this level, often by using XML. As well as simple pieces of data, like strings, more complicated things are standardized in this layer. Two examples are "objects" in object-oriented programming (OOP) and the exact way that streaming video is transmitted.

In many broadly used applications and protocols, no distinction is made between the presentation layer and application layer. For example, hypertext transfer protocol (HTTP) and hypertext transfer protocol secure (HTTPS), generally are regarded as an application layer protocol, have presentation layer such as the ability to identify character encoding for proper conversion, which is then carried out in the application layer.

Within the service layering semantics of the OSI network architecture, the presentation layer responds to service requests from the application layer and sends service requests to the session layer.

In the OSI model, the presentation layer ensures the information that the application layer of one system sends out is readable through the application layer. If necessary, the presentation layer might be able to translate between multiple data formats by using a known format.

- **Layer 7: Application Layer**

In the Internet, application layer is an abstraction layer reserved for communication protocols and methods designed for process-to-process communications across Internet Protocol (IP). Application layer protocols use the underlying transport layer protocols to establish process-to-process connections via ports.

In the OSI model, the definition of its application layer is narrower in scope. The OSI model defines application layer as the user interface which is responsible for displaying the information received. The OSI application layer is responsible for displaying data and images to users in a human recognizable way and interacting with the presentation layer below it.

OSI separates functionality above the transport layer is at twofold: the session layer and the presentation layer, specifying strict modular separation of functionality at these layers. It also provides protocol implementations for each layer.

The application layer is the seventh layer of the OSI model and the only one that directly interacts with end user. The application layer provides a variety of services including:

- Simple mail transfer protocol (SMTP)
- File transfer
- Web surfing
- Web chat
- email clients
- Network data sharing
- Virtual terminals
- Various file and data operations

The application layer provides fully end-user access to a large amount of shared network services for data flow of the efficient OSI model. This layer has a wealth of responsibilities including error handling and recovery, data flow over a network, and full network flow. It is also used to develop network-based applications.

More than 15 protocols are used in the application layer, including file transfer protocol (FTP), telnet, trivial file transfer protocol, and simple network management protocol [13].

3.1.2 Network Hardware

Hub is a connection device in a network. Hubs are usually used to connect segments of a LAN which contain multiple ports. When a packet arrives at one port, it is copied to other ports as well so that all segments of the LAN can see the packets.

Hubs and switches serve as a central connection for all of network equipment and handle a data type known as frames. Frames carry data. When a frame is received, it is amplified and then transmitted to the port of destination.

In a hub, a frame is passed along or broadcasted to every one of its ports. It does not matter that the frame is only destined for one port. The hub has no ways of distinguishing which port a frame should be sent to. Passing it along every port ensures that it will reach its intended destination. This places a plurality of traffic on the network and leads to poor network response time.

A network **bridge** is the hardware that connects two or more networks so that the networks can establish communications. Users within home networks or small office networks generally prefer to have a bridge when they have different types of networks, information, or files could be shared among all of the computers on the networks.

A **router** is the device that forwards data packets along networks. A router is connected to at least two networks, commonly two LANs or WANs. Routers are located at gateway, the place where two or more networks are connected.

Routers use headers and forwarding tables to determine the best path forwarding the packets which use protocols such as ICMP to communicate with each other and configure the best route between any two hosts [1].

A network **switch** [13] is a small hardware device that joins multiple computers together within one local area network (LAN). Ethernet switch devices were commonly used on home networks before routers become popular; broadband routers integrate Ethernet switches directly into the unit as one of the many functions. High-performance network switches are widely used in corporate networks and data centers.

Physically, network switches look nearly identical to network hubs. Switches, unlike hubs, are capable of inspecting data as messages are received via a method called packet switching. A switch determines the source and destination device of each packet and forwards data only to the specific device intended to conserve network bandwidth and generally improve performance compared to hubs. A typical connection between network hardware is shown in Fig. 3.1.

3.1.3 Network Threats and Attacks

A threat is potential security violation [5]. The potential threats of a network usually are from software, hardware, and virtualized system [10].

In computer networks [5], an attack is any attempt to destroy, expose, alter, disable, espionage, or gain unauthorized access or make unauthorized use of an asset. Network attack is usually defined as an intrusion on network infrastructure that

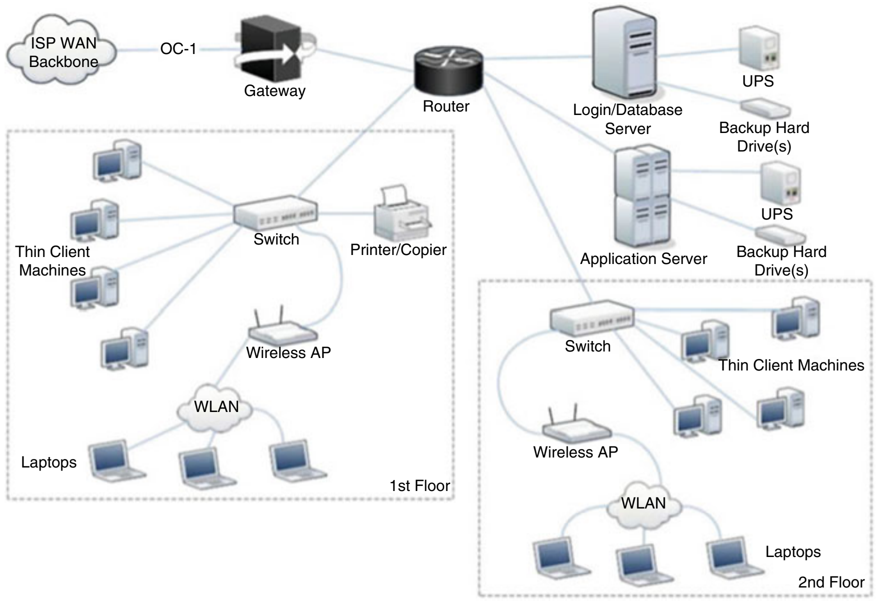


Fig. 3.1 Typical connection of a computer network

will firstly analyze the environment and collect information in order to exploit the existing open ports or vulnerabilities - this may include as well unauthorized access to resources [13]. In such cases where the purpose of attack is only to learn and get information from the system, but the system resources are not altered or disabled. Active attacks occur where the perpetrator accesses and either alters, disables, or destroys resources or data. Attacks are performed either from outside of the organization by unauthorized entity (outside attack) or from the company by using an "insider" that already has certain access to the network (inside attack). Very often the network attack itself is combined with an introduction of a malware component to attack the targeted systems.

A passive attack monitors unencrypted traffic and looks for clear-text passwords and sensitive information that are used in other types of attacks. Passive attacks are comprised of traffic analysis, monitoring unprotected communications, decrypting weakly encrypted traffic, and capturing authentication information. Passive interception of network operations enables adversaries to see upcoming actions. Passive attacks result in the disclosure of information or data files to an attacker without the consent or knowledge of the user.

In an active attack, the attacker tries to bypass or break into secured systems. This can be carried out through viruses, worms, or Trojan horses. Active attacks include attempts to circumvent or break protection features, introduce malicious code, and steal or modify information. These attacks are mounted against a network backbone, exploit information in transit, electronically penetrate an entire enclave, or attack an authorized remote user during an attempt to connect to an enclave. Active attacks result in the disclosure or dissemination of data files, DoS, or modification of data.

(DoS) Denial of Service aims at disrupting legitimate use and has two categories: hogging the resources and disrupting network. Unusually DoS will slow the network performance (opening files or accessing websites), cause unavailability of a particular website, have not ability to access any website, dramatic increase in the number of spam emails received, disconnection of a wireless or wired Internet connection, etc. [10].

A distributed attack requires the adversary introduction code, such as a Trojan horse or back-door program, a "trusted" component or software that will later be distributed to many other companies and the focus of users distribution attacks on the malicious modification of hardware or software at the factory or during distribution [13]. These attacks introduce malicious code, such as a backdoor of a product to gain unauthorized access at a specific later time.

Insider attacker
An insider attack such as a disgruntled employee, intentionally eavesdrops, steals, or damages information, uses information in a fraudulent manner or denies access to other authorized users. Not malicious attacks typically result from carelessness, lack of knowledge, or intentional circumvention of security for such reasons as performing a task.

Close-in attack: A close-in attack refers to get physically close to network components, data, and systems in order to learn more about a network. Close-in attacks consist of regular individuals attaining close physical proximity to networks, systems, or facilities for the purpose of modifying, gathering, or denying access to information. Close-in physical proximity is achieved through surreptitious entry into the network, open access, or both.

One popular form of close-in attack is a social engineering; the attacker compromises the network or system through social interaction with a person by using an email message or phone. Various tricks may be used by an individual to reveal secure information. The leaked information would most likely be used in a subsequent attack so as to gain unauthorized access to a system or network.

Phishing attack: In phishing attack, a hacker creates a fake website that looks exactly like a popular site such as the paypal. The phishing part of this attack is that the hacker then sends an email to trick users to click a link that leads to the fraud site. When a user logs on with the account information, the hacker records the username and password, then tries that information on the real site.

Hijack attack: In a hijack attack, a hacker takes over a session between a user and another individual, and disconnects the other individual from the communication channel. While the user still believes that (s)he is talking to the original party and may send private information to the hacker.

Spoof attack: In a spoof attack, the hacker modifies the source address of packets (s)he sent so that they appear to be coming from someone else. This may be to bypass the firewall rules [13].

Buffer overflow: A buffer overflow attack is happened when the attacker sends more data to an application more than what is expected. A buffer overflow attack usually results in the attacker gaining administrative access to the system in a command prompt or shell.

Exploit attack: In this type of attacks, the attacker knows a security problem within an operating system or a piece of software and leverages the knowledge by exploiting the vulnerability [13].

Password attack: An attacker tries to crack the passwords stored in a database or a password-protected file [3]. There are three major types of password attacks: a dictionary attack, a brute-force attack, and a hybrid attack. A dictionary attack uses a word list file, which is a list of potential passwords. A brute force attack is happened when the attacker tries every possible combinations of the attacks [15].

Our passwords [1] are usually associated with digital devices which have close relationship with our work and life, usually we have three types of passwords:

- *Normal Password*. Normally this type of passwords have been used in computer systems, Internet accounts, etc. A classical example is the authentication interface of Gmail (the Google email) system, see Fig. 3.2.

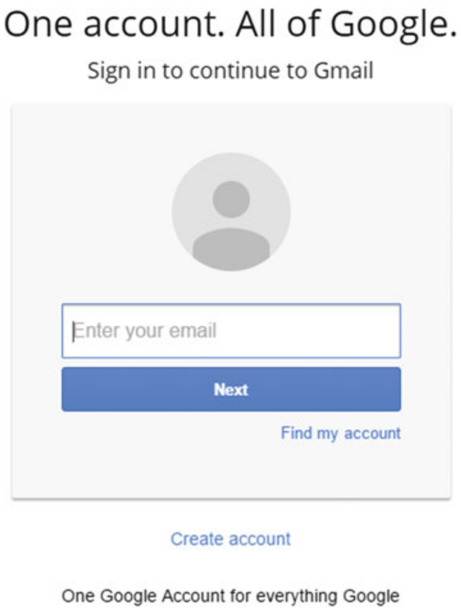

One account. All of Google.
Sign in to continue to Gmail
Enter your email
Next
Find my account
Create account
One Google Account for everything Google

Fig. 3.2 Authentication interface of the Gmail system

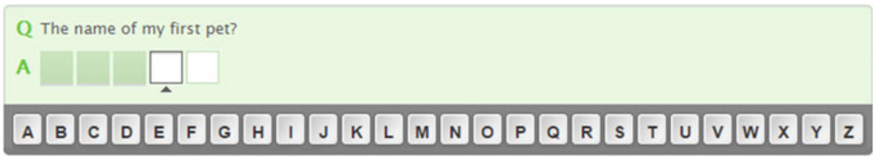
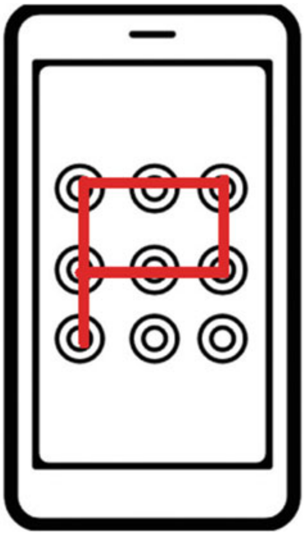

Q The name of my first pet?
A
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Fig. 3.3 An example of dynamic password based authentication

Fig. 3.4 A graphical pattern 'P' is used as the password of a smart phone



- *Dynamic Password*. Namely one pad password, it is usually updated timely, synchronization between a token and servers are required. It provides another layer of protection of the normal password systems. We may find these passwords from our bank account, ATM machines, etc. See an example from Fig. 3.3.
- *Visual Password*: Using visual information for authentication and authorization, the cyberspace is much bigger than the traditional cypher text space. Nowadays, most of smart phones are using a graphical pattern as the password to login into the mobile operating system as shown in Fig. 3.4, several systems even adopted user's fingerprint as the password.

In order to defend password guessing, we provide below solutions in this section:

- In password security, we need to change the default password, it is used to thwart exhaustive search.
- In password security, we need to set password format mixing upper- and lowercase symbol including numerical and other nonalphabetical symbols in a password.

- In password security, we should avoid obvious passwords, break the semantic and forbid to use person's name and birthday as the important passwords.

In password attacks, there are multiple ways including exhaustive brute-force attacks, dictionary-based attacks, rule-based attacks, Markov model-based attacks, and Password Guessing based on RNNs, etc.

PassGAN [9] replaces human-generated password rules with theory-grounded machine learning algorithms, it uses a Generative Adversarial Network (GAN) to autonomously learn the distribution of real passwords from actual password leaks, and to generate high-quality password guesses. PassGAN can autonomously extract a considerable number of password properties that current state-of-the art rules do not encode.

The idea of PassGAN [9] is to train a neural network so as to autonomously determine password characteristics and structures, and leverage this knowledge to generate new samples that follow the same distribution. The output of PassGAN (fake samples) becomes closer to the distribution of passwords in the original leak, and hence more likely to match real users' passwords.

In general, regarding to network security, we usually refer to:

- **CIA**, confidentiality, integrity, and availability in information security.

Confidentiality: CIA is a model designed to guide policies for information security within an organization. In the context, the confidentiality is a set of rules that limit access to the confidential information, the integrity is the assurance that the information is trustworthy and accurate, and the availability is a guarantee of ready access to the information by authorized staff.

Confidentiality prevents sensitive information from reaching the wrong persons, while making sure that the right receivers in fact get it. A good example is an account number or routing number when banking online. Data encryption is a common method of ensuring confidentiality. User IDs and passwords constitute a standard procedure. In addition, users take precautions to minimize the number of places where the information appears, the number of times which is actually transmitted to complete a required transaction.

Integrity involves maintaining the consistency, accuracy, and trustworthiness of data over its entire life cycle. The data must not be changed in transit, and the steps must be taken to ensure that the data cannot be altered by unauthorized staff members (for example, in a breach of confidentiality). If an unexpected change occurs, a backup copy must be available to restore the affected data to its correct state.

Availability is best ensured by rigorously maintaining all hardware, performing hardware repairs immediately when needed, providing a certain measure of redundancy and failover, offering adequate communications bandwidth and preventing the occurrence of bottlenecks, implementing emergency backup power systems, keeping the current with all necessary system upgrades, and guarding against malicious actions.

- **AAA**: authentication, authorization, and accounting in computer security.

Authentication, authorization, and accounting (AAA) are terms of a framework for intelligently controlling access to computer resources, enforcing policies, auditing usage [1,5], and providing the information necessary to bill for services. These combined processes are considered importantly for effective network management and security.

As the first process, **authentication** provides a way of identifying a user, typically by having the user to enter a valid username and password before an access is granted [5]. The process of authentication is based on each user having a unique set of criteria for gaining access. The AAA compares a user's authentication credentials with others stored in a database [3]. If the credentials match, the user is granted. If the credentials are at variance, authentication fails and the access is denied.

Following authentication, a user must gain authorization for committing a task. After logging into a system, for instance, the user may be issued permissions. The authorization process determines whether the user has the authority to deal with a task. Simple authorization is the process of enforcing policies: determining what types or qualities of activities, resources, or services the user is permitted [5]. Usually, authorization occurs within the context of authentication. Once a user is authenticated, (s)he may be authorized for different types of access or activity.

The final plank in the AAA framework is **accounting**, which measures the resources a user consumes during access. This includes the amount of system time or data that a user has sent and/or received during a session. Accounting is carried out by using the logs of session conversations and usage information which is used for authorization, billing, trend analysis, resource utilization, and capacity planning activities [1].

- **MARS**: monitoring, analysis, and response in network security.

Security monitoring, analysis, and response system (MARS) provide security monitoring for network devices and host applications supporting both Cisco and other vendors. Security monitoring [1] with MARS greatly reduces false positives by providing an end-to-end topological view of the network, which helps improve threat identification, mitigation responses, and compliance [13].

PSF: Protection, surveillance, and forensics in data security.

Protection refers to cryptographic level which is related to network security and data security. Surveillance concentrates on behavior and scene monitoring [13]. Network forensics is a subbranch of digital forensics relating to monitoring and analysis of computer network traffic for the purposes of information gathering, legal evidence, or intrusion detection after incidents. Unlike other areas of digital forensics, network investigations deal with volatile and dynamic information. Network traffic is transmitted and then lost, so network forensics is often a proactive investigation.

3.1.3.1 Strategies for Network Security

From management viewpoint, we hope a network is well configured and has been setup properly, the network is expected to have the capability for protecting themselves and providing responses to any potential attacks. Ideally, we protect our computer networks using the below strategies:

- *Guard the outside perimeter*: We utilize firewall and other hardware devices to separate a computer network from others [13]. This makes the problems such as computer virus from the inside that could not affect our computers; meanwhile, the outside hazards could not penetrate the protection and keep the computer inside the premises working well.
- *Control internal access*: For the most of computer networks, the problems or attacks usually are from the inside; therefore, internal access should be controlled for all the operations. The logs for monitoring networks should record any events happened in the computer network; the necessary event analysis and prediction are needed to reduce the threats from inside of the network [2,13]. Authorization and authentication are crucial at initial stage of protecting networks; accounting and auditing help us to trace back for computer security [5]. Reporting and logging on a network could help us to record the events happened in the networks [5].
- *Police the traffic*: The police could reduce network traffic and simultaneously publicly and openly guide authorized staff and users what actions are right and should be allowed inside a computer network. The strict policies reduce the crowded hazards and reasonably utilize the hardware and software resources, consequently make sure all the networking parts working in highly efficient way.
- *Secure the devices*: The server rooms and power rooms should be absolutely secured. Irrelevant staff should keep clear from these rooms with surveillance data and refrain from the devices inside important places. For the servers and power devices, the policies and regulations should be regularly explained and followed [5]. The input and output devices of computers including keyboards, mice, monitoring, printers, etc. should be forbidden to be touched and be locked in free time, any operations should be observed by more than two onsite staff members so as to reduce any security mistakes [1,13].
- *Use self-defending networks*: Computer networks should have the self-responding and defending ability so as to reduce any reproduced attacks and possible hazards, avoiding cascade failures [5]. Once any parts of the network are out of order in running, the network should separate out the portions and make sure them working well [1].
- *Policies, procedures, standards guidelines*: These regulations and guidelines always remind the staff operating under the instructions in standard and keeping alert to any risks. The regulations will take effects to new employees and the unauthorized persons keep distance from the perimeter.

- *Training and awareness campaigns*: The staff members who have the opportunities to touch the computer networks should be trained regularly and update the knowledge periodically. The campaign will help the staff to exchange experience and learn how to respond to any incidents.

3.1.3.2 Network Event Monitoring and Analysis

An event is the occurrence within a computer system that converses with other systems or users [2]. Computer and network systems contain event logs that encompass enormous amount of data. These event logs hold records of any behaviors or actions a network device performs. Events may also involve illegal activities such as malicious attacks. The plan is to assemble these events and examine their relationships with research and recording each activity into a knowledge database. This information will help to avoid further incidents or risks after the events have occurred.

The goal of network event security analytics is to provide a mechanism into gathering network data so that it may be analyzed for different purposes to improve network security and performance by viewing packets traverse across the network [4]. The ability to determine what kind of traffic enters a network and what effects have on a network can help administrators to come up with a viable solution to problems they encounter.

Network analysis tools are helpful to test the strength and vulnerabilities of a network because a network cannot be deemed secure without testing. These tools depend heavily on the OSI model that allows for communications between different network layers so that relevant information can be gathered. Investigations into this problem are helpful to identify methods of extrapolating information from the systems by implementing a listening techniques or an activity tracker within a single computer or multiple computing systems to give detailed feedback about what occurs within a system and how we can display this information in a presentable user interface that does not overwhelm the user with useless information.

Event monitoring software commonly refers to packet sniffer or logging applications because of its ability to scour network traffic and gain data about what is being transferred over a network. The ability to analyze traffic sent between two networked machines is common job for a network administrator to track down faults or suspicious activity within a network. Software integrates itself into network adapters so all traffic that passes through the adapter available for viewing. Each packet going to a computer is considered as event by a packet sniffer and can be viewed in real time.

Event logging security component such as firewall is a network tool that is purposely built to permit deny both ingoing and outgoing traffic to and from an organizations' network. Firewalls are developed through the usage of different Open System Interconnection (OSI) layers. Each layer handles different pieces of information, which allows firewalls to obtain a complete picture of information such as applications, networks, and transports. There are various types of firewalls which include packet filtering firewalls, proxy firewalls, address translation firewalls, and application layer firewalls [13]. The differences between these firewalls are that they serve different layers of the OSI model and are suitable for specific tasks within a network.

By default, firewalls follow a rule set that must be configured by a network administrator for appropriate traffic and if nothing is configured they are in a deny mode when activated. Firewalls are typically placed at network boundaries, which separate the private domain and public domain of an organization network [13]. This is done so all traffic must be inspected before entering or leaving the network mitigating the risk of malicious network attacks that can occur through bypassing security measures [1].

Firewalls log any network events into a file or a built-in event tracker for understanding the causes of why a traffic is permitted and others are not. This is helpful for troubleshooting and creating solutions to solve problems that occur [13].

Files accessed are scanned in real time to prevent any code execution of infected files. Alternatively, users can schedule or run scans of an entire computer to find inert files that may not have executed but contain malicious code. Any file access triggers the anti-virus into its scanning mode before the file is permitted for run-time execution. Such events are continually logged into the anti-virus and permit it to track files that have been verified as safe [13].

Information such as source, destination, port, encapsulated data, and header can be extracted from packets. The ability to view this data is valuable because administrators can make decisions about what is going on within their network and take actions when necessary [2].

3.2 Data Scrambling and Descrambling Technologies

3.2.1 Cryptographic *mod* Operation

In cryptography [1], we have the modulo operation *mod* which is circular. The prominent property of this modulo operation is shown below. Given $x \in Z$,

$0 \; mod \; x = 0;$
$1 \; mod \; x = 1;$
$\ldots\ldots$
$x - 1 \; mod \; x = x - 1;$
$x \; mod \; x = 0,$
$x + 1 \; mod \; x = 1;$
$\ldots$

For the 26 English letters, we scramble their alphabetic order using,

$$y = (x + s) \; mod \; 26 \tag{3.1}$$

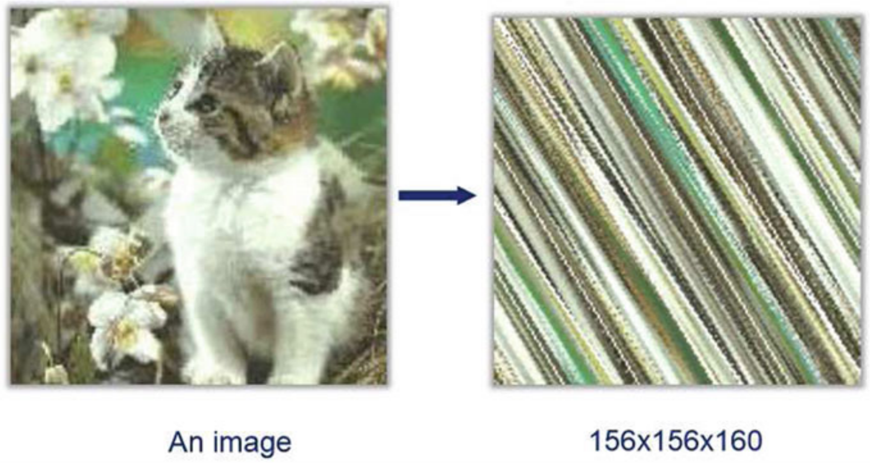


Fig. 3.5 Image scrambling using Arnold's Cat Map

where x is the position of a letter in the sequence, s is a constant for shifting, and y is letter position after scrambled. For example, the letter 'a' is located at the first position $x = 0$, after shift $s = 5$ position using the modular operation, the letter 'a' will be located at the position $y = 5$. If all the letters of a string have been shifted in this way, the string will be scrambled or encrypted.

When scrambling digital images in spatial domain, Arnold's Cat Map is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} mod\ N \tag{3.2}$$

where $x, y = 0, \ldots, N - 1$, and N is the width and height of the image. The scrambled image with resolution 156×156 is shown in Fig. 3.5.

For scrambling digital images in color space we use,

$$\begin{bmatrix} r' \\ g' \\ b' \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} mod\ c \tag{3.3}$$

where $r, g, b = 0, \ldots, c - 1$, and c is the maximum color number of the image, usually $c = 2^n$, n is the depth of a color bitmap. An example of video scrambling is shown in Fig. 3.6.

For scrambling digital audio clips, a typical equation is shown in Eq. (3.4):

$$T = \sum_{k=1}^{n} a_k T_{m,k} \tag{3.4}$$

where $T_{k,m} = \{t_{ij}\}_{n \times n}$, $t_{ij} = \begin{cases} 1, & j = (m \cdot i + k)\ mod\ n \\ 0, & \text{others} \end{cases}$.

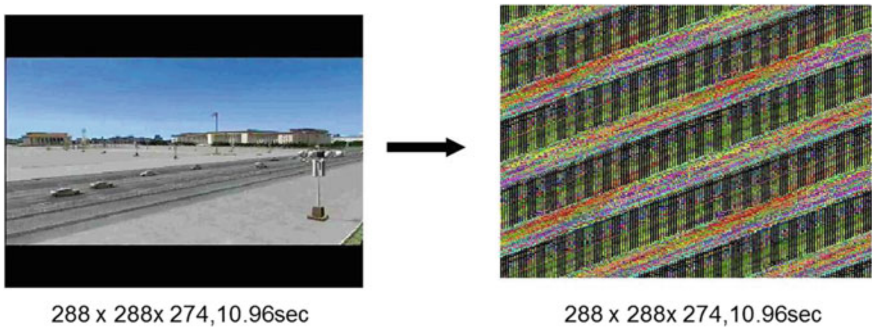


Fig. 3.6 Scrambling in spatial and color domain

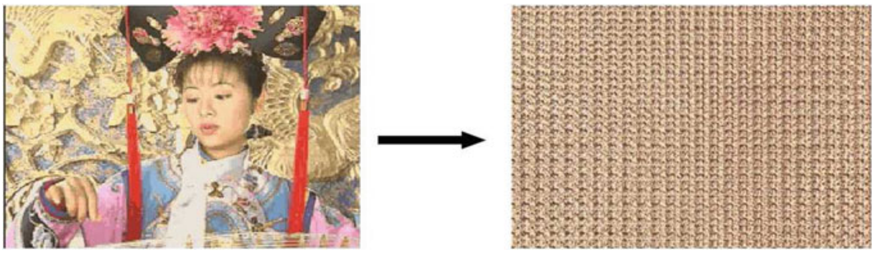


Fig. 3.7 Scrambling in spatial domain

Suppose we have a vector $\mathbf{V} = (v_1, v_2, \ldots, v_n)^\top$, an identity matrix is $I = diag(\alpha_1, \alpha_2, \ldots, \alpha_n), \alpha_i = 1, i = 1, \ldots, n,$ apparently, $\mathbf{V}' = \mathbf{V} = \mathbf{I} \cdot \mathbf{V} = \mathbf{V} \cdot \mathbf{I}$. Now if we swap the two rows i and j $(i \leq j)$ of the matrix $\mathbf{I}$, we have,

$$\mathbf{I}' = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{3.5}$$

Correspondingly, we have $\mathbf{V}' = (v_1, v_2, \ldots, v_{i-1}, v_j, v_{i+1}, \ldots, v_{j-1}, v_i, v_{j+1}, \ldots, v_n)$ after $\mathbf{V}' = \mathbf{I}' \cdot \mathbf{V}$, hence the vector $\mathbf{V}$ has been scrambled to the vector $\mathbf{V}'$. Without loss of generality, we assume all the audio clips could be segmented into the vectors $\mathbf{V}, n = |\mathbf{V}| < \infty$.

Of course, this equation $\mathbf{V}' = \mathbf{U} \cdot \mathbf{I}' \cdot \mathbf{V}$ could be generalized for scrambling digital images or videos $\mathbf{U} = (a_1, a_2, \ldots, a_n)$. If we properly set the row numbers i, j and different values for $a_k \neq 0$, $a_k \in \mathscr{R}$, $k = 1, \ldots, n$, the cypher space will be much bigger. An example of video scrambling is shown in Fig. 3.7.

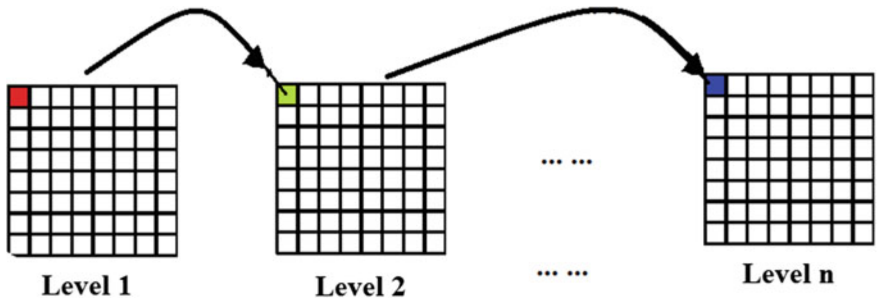


Fig. 3.8 Progressive image scrambling at multiple levels

Progressive scrambling of digital images is defined as,

$$\lim_{|\Omega|\to 0} S_{|\Omega|}(M_1, M_2) = \lim_{|\Omega|\to 0} \frac{1}{|\Omega|} \sum_{\Omega \subset M_1} \triangle_{\min}(\Omega) = \varepsilon_0 > 0 \tag{3.6}$$

$$\triangle_{\min}(\Omega) = \min(\{|\Omega - \Omega'|, |\Omega| = |\Omega'|, \forall \Omega' \subset M_2\}), \Omega \subset M_1 \tag{3.7}$$

where $S_{|\Omega|}(M_1, M_2)$ is called scrambling distance between a given media M_1 and its scrambled one M_2 under the resolution $|\Omega|$ [22].

We decompose the media in multiple resolutions,

$$D_l(m) = D_l(E)(m) = D_l([C_1, C_2, \dots, C_n](m)) \tag{3.8}$$

where $D_l(m), l = 1, 2, \dots, L$ is a decomposition of the media m at level l, C_i are the coefficients after the decomposition. The components $E = (C_1, C_2, \dots, C_n), i = 1, 2, \dots, n$ and the given random keys will be operated together using the operation $\bigotimes$. The operation was chosen usually for the sake of its simplicity of description and implementation.

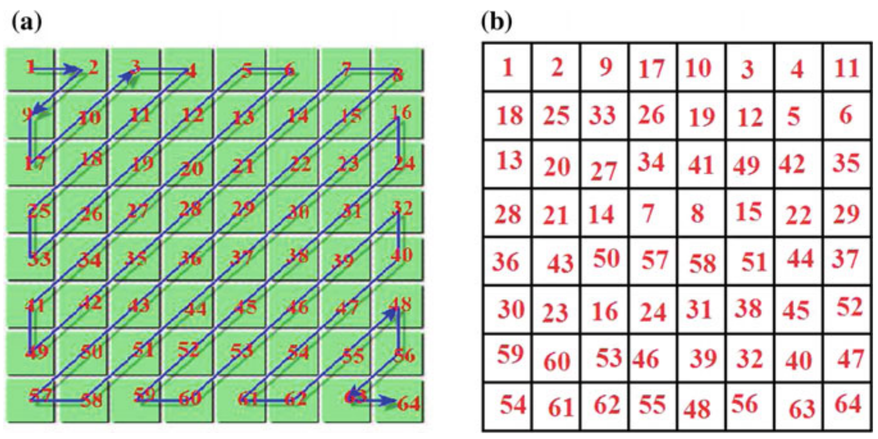
$$E'_l(m) = E_l(m) \bigotimes K_l \tag{3.9}$$

$$D'_l(m) = D_l(E'_l)(m) \tag{3.10}$$

$$E_l(m) \bigotimes K_l = [C_{1l}, C_{2l}, \dots, C_{nl}](m) \bigotimes K_l = [C_{1l} \bigotimes K_l, C_{2l} \bigotimes K_l, \dots, C_{nl} \bigotimes K_l](m) \tag{3.11}$$

Therefore, we define a tensor product $\bigotimes$, it is distributive and operative on each element iteratively shown in Fig. 3.8.

For example, the zigzag ordering of DCT components in JPEG could be defined as a tensor product and applied to 8×8 blocks of images for multiple resolutions hierarchically and iteratively in spatial or frequency domain as shown in Fig. 3.9.



1	2	9	17	10	3	4	11
18	25	33	26	19	12	5	6
13	20	27	34	41	49	42	35
28	21	14	7	8	15	22	29
36	43	50	57	58	51	44	37
30	23	16	24	31	38	45	52
59	60	53	46	39	32	40	47
54	61	62	55	48	56	63	64

Fig. 3.9 Image progressive scrambling using zigzag reordering. **a** the original image and **b** the scrambled image

Mathematically, the zigzag ordering could be implemented by using the elementary matrix. Suppose the coefficients c_i and c_j, $(i \neq j)$ will be swapped in zigzag ordering, the i-th row and j-th column of the identify matrix will be swapped simultaneously so as to implement this operation. The descrambling is exactly to use the same matrix as

$$K \cdot K = I, K = K' = K^{-1} \tag{3.12}$$

$$(C_1, C_2, \ldots, C_j, \ldots, C_i, \ldots, C_n) = (C_1, C_2, \ldots, C_i, \ldots, C_j, \ldots, C_n) \cdot K \tag{3.13}$$

$$K = \begin{bmatrix} 1\,0 & 0 & 0 & 0 & 0 & 0 & 0\,0 \\ 0\,1 & 0 & 0 & 0 & 0 & 0 & 0\,0 \\ 0\,0 & \cdots & 0 & 0 & 0 & 0 & 0\,0 \\ 0\,0 & 1 & 0 & 0 & 0 & 0 & 0\,0 \\ 0\,0 & 0 & 0 & \cdots & 0 & 0 & 0\,0 \\ 0\,0 & 0 & 0 & 0 & 0 & 1 & 0\,0 \\ 0\,0 & 0 & 0 & 0 & 0 & \cdots & 0\,0 \\ 0\,0 & 0 & 0 & 0 & 0 & 0 & 1\,0 \\ 0\,0 & 0 & 0 & 0 & 0 & 0 & 0\,1 \end{bmatrix} \tag{3.14}$$

where K is regarded as the key for progressive scrambling. For each 8×8 block, we set different types of keys. This re-ordering operation-based scrambling also is able to be applied to frequency domains such as Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT).

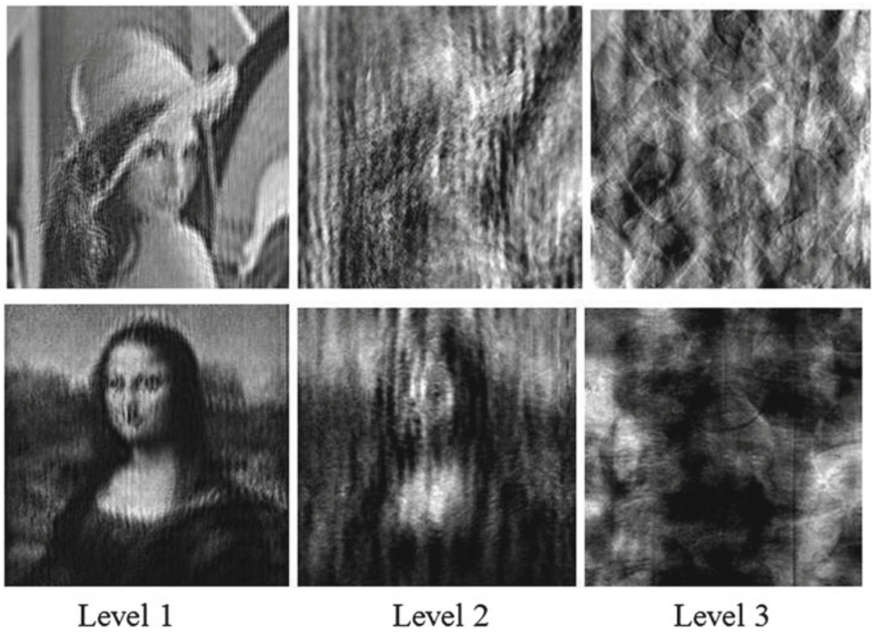


Fig. 3.10 Progressive scrambled images (512 × 512) in DCT domain for the images Lena, Mona

In spatial domain, we segment an image into 8 × 8 pixel-size blocks and scramble them using zigzag ordering. On the basis of the first round of scrambling, we regard each 8 × 8 pixel-size block as a unit and scramble 8 × 8 such blocks as the second level scrambling. After all the blocks have been scrambled in this hierarchical way, the entire image has been scrambled. The descrambling is the exactly reversal procedure of this progressive scrambling. The results are shown in Fig. 3.10.

3.2.2 Cryptographic *xor* Operation

In cryptography [1], another very crucial operator is *xor*. Since *xor* is the exclusive *or* operation in logic, we have,
$1\ xor\ 1 = 0;$
$0\ xor\ 0 = 0;$
$1\ xor\ 0 = 1;$
$0\ xor\ 1 = 1;$

Thus, the salient property of this operation is, If $a \;\; xor \;\; b = c$, then $a \;\; xor \;\; c = b$ and $c \;\; xor \;\; b = a$, where a, b, c are binary numbers $(\beta_0, \beta_1, \ldots, \beta_n)_2 = \sum_{i=0}^{n} \beta_i \cdot 2^i$, $\beta_i = 0, 1; i = 0, \ldots, n$.

Usually, encryption algorithm consists of set of K keys, set of M messages, and set of C ciphertexts (encrypted messages). Alice is the message sender A, Bob is the message receiver B, and Eva is the assumed eavesdropper. Symmetric encryption adopts the same key to encrypt and decrypt $E(k)$ from $D(k)$, and vice versa. For example, DES(Data Encryption Standard) is symmetric block-encryption algorithm, encrypts a block of data at a time using the Feistel function (expansion, key mixing, substitution, and permutation). Since it is not secure, triple-DES (3DES) has been employed and considered as more secure. The similar encryption algorithm includes AES (Advanced Encryption Standard) [15, 17, 19].

RC4, created by Rivest Cipher in 1984, is the most powerful symmetric stream cipher, but it was known to have vulnerabilities. It encrypts/decrypts a stream of bytes (i.e., wireless transmission). The key is an input to pseudorandom-bit generator which could generate an infinite key stream [15, 17, 19].

Asymmetric encryption is based on two keys, the published key is used to encrypt data while the private key is used to decrypt the data encrypted by using its published public key. Alice (A) transmits her key (n, e) to Bob so as to encrypt the plaintext M,

$$C = M^e (mod \;\; n)$$

When Bob receives encrypted message C, he uses the private key d to get the message,

$$M = C^d (mod \;\; n)$$

For example, the public key is $(n = 3233, e = 17)$,

$$C = M^{17} (mod \;\; 3233)$$

The private key is $d = 2753$,

$$M = C^{2753} (mod \;\; 3233)$$

If the message $M = 65$, then,

$$C = 65^{17} (mod \;\; 3233) = 2790$$

To decrypt,

$$M = 2790^{2753} (mod \;\; 3233) = 65$$

The security of this scheme is guaranteed by the Chinese remainder theorem (CRT),

$$d \cdot e \equiv 1(mod \ \ \phi(n))$$

where

$$\phi(n) = (p-1) \cdot (q-1)$$

and

$$\phi(n) = p \cdot q$$

Note: p and q are two very big prime numbers [15,17,19].

Formally, Chinese remainder theorem (CRT) is described as:
If $n_i \in \mathbf{Z}^+$, $n_i > 1$, $i = 1, 2, \ldots, k$ are pairwise co-prime and $a_i \in \mathbf{Z}$, $0 \leq a_i < n_i$, then there exists an integer r such that congruences $r \equiv a_i (\bmod\ n_i)$ $(i = 1, 2, \ldots, k)$.

$$r = \sum_{i=1}^{k} a_i M_i N_i \quad \bmod N \tag{3.15}$$

where $N = \prod_{i=1}^{k} n_i$, $N_i = \frac{N}{n_i}$, $N_i M_i \equiv 1(\bmod\ n_i)$.

For example, $k = 3$, $n_1 = 7$, $n_2 = 11$, $n_3 = 13$, $N = 1001$;
$N_1 = 143$, $N_2 = 91$, $N_3 = 77$; $M_1 = 5$, $M_2 = 4$, $M_3 = 12$. If $r \equiv 5(\bmod\ 7)$, $r \equiv 3(\bmod\ 11)$, $r \equiv 10(\bmod\ 13)$, namely, $a_1 = 5$, $a_2 = 3$, $a_3 = 10$, then
$r = (5 \times 143 \times 5 + 3 \times 91 \times 4 + 10 \times 77 \times 12) \bmod 1001 = 894$.

For digital images and videos, we could use *xor* operation to pixel values, pixel blocks, and video frames with the given keys so that the digital media could be scrambled. However, only using *xor* in color space for visual scrambling is not sufficient because we may perceive the object shapes, edges, or contours on the scrambled images.

Compared to those existing cryptographic or scrambling techniques based on plain text such as substitution, transposition, DES, 3DES, AES, RSA, and EEC, the image scrambling approaches have much wider encryption space instead of only taking the ASCII codes into consideration. On the other hand, the cryptographic techniques may have the problems that could not scramble a digital image bxecause the coherence existing between the row and column or blocks of a digital image could not be broken [6,16].

Hilbert space-filling curves shown in Fig. 3.11 have been used to fill up the image space with various resolutions because of its famous features such as self-similarity and multiple resolutions. Once the pixels on one of such curves are sorted in order, the digital images will be scrambled in spatial domain, this has been applied to encrypt digital TV signals. Further scrambling techniques could be combined with the ones from frequency domain and compression domain of digital images together.

The HSC is with a fractal structure generated by a recursive production rule which has the property of self-similarity and satisfies IFS system, its dimension is a fraction supported by the fixed-point theory. After several rounds of recursions

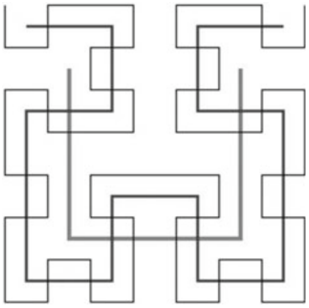


Fig. 3.11 Three rounds of hilbert curves

started from the fractal generator, the curve will fill up a given space recursively. There are multiple fractal rules to generate HSC curves. In this scrambled image, only the pixel locations have been re-ordered, the pixel color values still hold.

For each block, we recursively generate a corresponding HSC curve using the generator, the curve starts from very beginning shown in red color, its mouth points in upward. In the second iteration, for each turning point including starting and end points, we generate the same shape; however, the size and orientation will be changed. At the starting point, we rotate the generator 90° toward the left (anti-clockwise); at the end point, we turn the generator to right for 90° (clockwise), at the other two turning points, the two orientations of the generators are the same. The procedure is described as Eq. (3.16),

$$p(x, y) = HSC(p(x, y)) \tag{3.16}$$

where $HSC(\cdot)$ is the iterative function, $p(x, y)$ is the turning point on the curve, the stop condition for this recursion is the final resolution reached so as to fully fill the given space, the Hausdorff dimension of this fractal curve is 2.00.

The image scrambling operation is described as,

$$I'(x, y) = HSC(I(x, y)) \ mod \ W \tag{3.17}$$

where I is the previous image without scrambling, I' is the scrambled image, and W is image width. After generating this HSC curve, we sort the pixel order using the pixel sequence on the HSC curve shown in Eq. (3.17). Equation (3.17) first converts 2D points into 1D order; then, the 1D sequence will be used to fill up the image space line by line from top to end, consequently the image is fully scrambled which is exported as the encrypted image.

The encryption and decryption algorithms using HSC-based image scrambling in DCT domain change the pixel sequence spatially, but do not alter color values of image pixels. Therefore, they do not influence the visual effects of an image.

Figure 3.12 shows one of the results of image scrambling using the HSC-based image scrambling in DCT domain with different resolutions. We typically yield a HSC curve using recursion in fractal, then the DCT coefficients (DC and AC) within the blocks are reordered. Inverse DCT (IDCT) has been used to transfer the scrambled coefficients back to spatial domain. Since DCT and IDCT are based on orthogonal

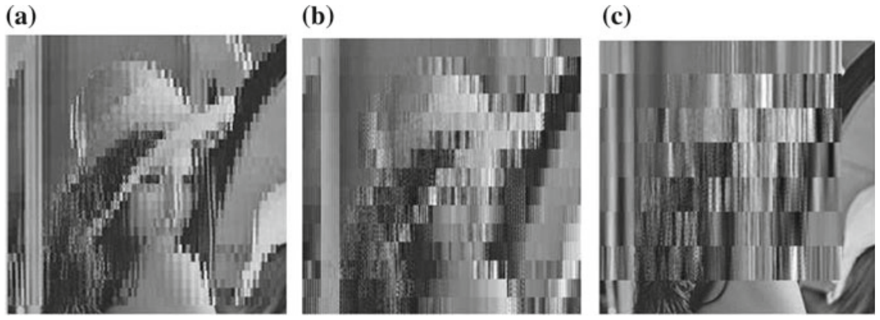


Fig. 3.12 Image encryption for Lena (512 × 512) in DCT domain using HSC scrambling for the blocks with different size **a** block size 16 × 16 **b** block size 32 × 32 **c** block size 64 × 64

decomposition of visual signals of block-based images, this decomposition ensures that after encryption and decryption, the original image is able to be reconstructed in blocks visually.

The security of this HSC encryption is ensured by using keys and the scrambling algorithm. This is the reason why the HSC generator has multiple choices, and could be rotated along the clockwise and anti-clockwise directions, the generator has four orientations. Based on different generators, the HSC curves will be completely different. Meanwhile, the image block has multiple choices with various resolutions, the scrambling based on different block sizes will lead to image encryption of different strengths. Larger the bock size, stronger the encryption. Therefore, what HSC has been selected at which resolution will be the secret key of the encryption algorithm [23].

The HSC-based image scrambling is different from traditional encryption algorithms, such as RSA, ECC, and secret sharing [15,21]. The reason is that the scrambling completely destroys the order of pixel locations in the image; therefore, the pixel operations such as edge extraction, SIFT, and others are not possible anymore (especially in DCT domain) [11].

3.2.3 Image Encryption on Frequency Domain

Image encryption techniques can be divided into two groups, the first one encrypts images on spatial domain; the other encrypts images on frequency domain [14]. Most of image encryption schemes focus on frequency domain, various methods based on Fourier Transform, discrete cosine transform (DCT) and discrete wavelet transform (DWT) have been widely applied to image encryption. Because DCT can avoid complex calculation compared to traditional DFT, DWT can obtain local properties of the input image on both spatial and frequency domains, which provides convenience for image encryption [8].

Discrete cosine transform (DCT) was widely used in image encryption. In 2010, a color image encryption algorithm was proposed based on Arnold transformation.

DCT was chosen because the pixel values of a given image are defined in real domain and the matrix still remains in the real domain after the transformation. On the other hand, the partition operation was applied before Arnold transformation so that the operation can be clearly conducted. As a result, the security of the encrypted images was improved successfully.

In 2013, a multi-image encryption algorithm was proposed based on cascade fractional Fourier transform. In this method, the input images are successively encrypted using a series of encryption keys until a final encrypted image is obtained. The algorithm not only works for the encryption of multiple images, but also is very secure. Because there are so many secret keys, the algorithm can be applied to multi-user authentication.

In 2000, a generalized image encryption algorithm based on fractional Fourier transform was proposed, and this method used a new generalized fractional Fourier transform instead of random phase mask. In this algorithm, the period of fractional Fourier transform is extended to any integer; thus, the period and transformation index are regarded as two secret keys.

3.3 Data Secure Transmission over Internet and Networks

In this section, we will introduce those HTTP servers, FTP servers broadcast those captured surveillance videos and audio around the world; meanwhile, no matter where we are, we can receive those audio and video like the famous YouTube website. Provided those cloud servers are utilized, the surveillance images, video and audio clips could be pushed to our accounts.

A typical FTP software is WinSCP shown in Fig. 3.13. If we have big surveillance data, we may upload them to a Unix, Linux, or Microsoft Windows server; traditionally, FTP software is designed to complete this work. Currently Microsoft Windows provide network services, we could upload the surveillance media to those server machines.

The typical HTTP server for uploading and playing videos based on Microsoft Windows is Apache though the operating system (OS) has its own ISP services [18]. The EasyPhP software integrated Apache and MySQL together for database management using PHP programming language, it provides great convenience for Internet users. Users could link videos to this website and utilize the possible players to play the videos like YouTube website (Fig. 3.14).

Hypertext transfer protocol secure (HTTPS) is an extension of the hypertext transfer protocol (HTTP) for secure communication over a computer network. In HTTPS, the communication protocol is encrypted using transport layer security (TLS) and secure sockets layer (SSL). HTTP is not encrypted, which can let attackers gain access to website accounts and sensitive information. HTTPS is considered secure. HTTPS must create a public key certificate for the web server. The site administrator typically creates a certificate for each user, a certificate is automatically checked by

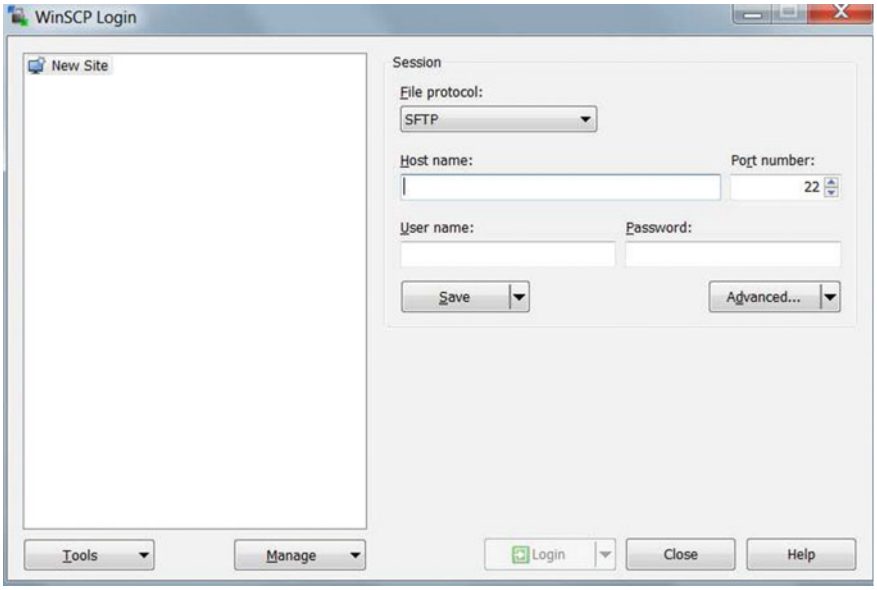


Fig. 3.13 Interface of the software WinSCP for FTP service

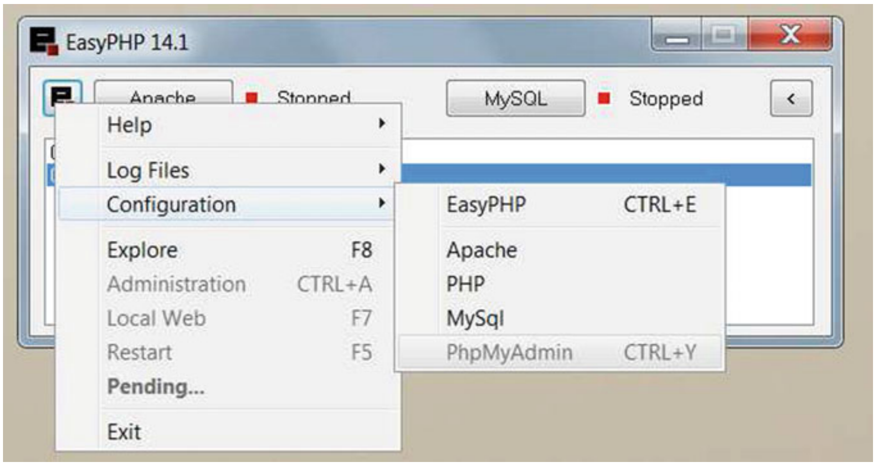
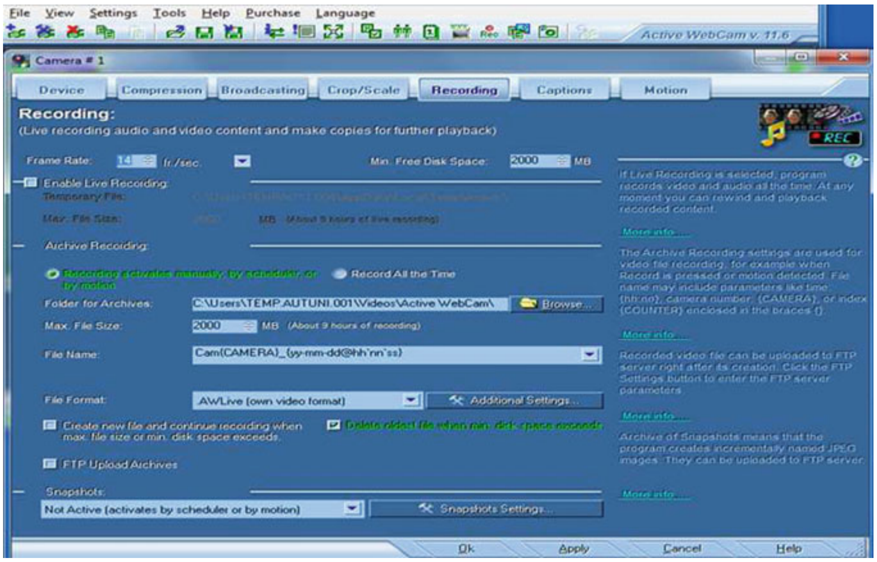


Fig. 3.14 The software EasyPhp is running as the http server



Recording:
(Live recording audio and video content and make copies for further playback)

Frame Rate: 14 fr./sec.
Min. Free Disk Space: 2000 MB

Enable Live Recording
Temporary File: [illegible]
Max. File Size: [illegible] MB (About 9 hours of live recording)

Archive Recording

[✓ Recording activates manually, by scheduler, or by motion] Record All the Time
Folder for Archives: C:\Users\TEMP.AUTUNI.001\Videos\Active WebCam\
Max. File Size: 2000 MB (About 9 hours of recording)
File Name: Cam{CAMERA}_{yy-mm-dd@hh'nn'ss}
File Format: AWLive (own video format)
Create new file and continue recording when max. file size or min. disk space exceeds.
[✓ Delete oldest file when min. disk space exceeds]
FTP Upload Archives

Snapshots
Not Active (activates by scheduler or by motion)

If Live Recording is selected, program records video and audio all the time. At any moment you can rewind and playback recorded content.

The Archive Recording settings are used for video file recording, for example when Record is pressed or motion detected. File name may include parameters like time (hh:nn), camera number (CAMERA), or index (COUNTER) enclosed in the braces {}.

Recorded video file can be uploaded to FTP server right after its creation. Click the FTP Settings button to enter the FTP server parameters.

Archive of Snapshots means that the program creates incrementally named JPEG images. They can be uploaded to FTP server.

Fig. 3.15 Interface of the software Active for video broadcasting

the server on each reconnect to verify the user's identity, potentially without even entering a password.

The media streaming software includes Active Webcam, Microsoft Express Encoder, and Adobe Media Encoder, these software could create a streaming server and transmit streaming monitoring data via Internet. At receiver sides, a web browser is sufficient enough, they are one to many broadcasting systems.

Active WebCam software shown in Fig. 3.15 broadcasts live MPEG-4 video stream and MP3 audio signals up to 30 frames per second, the software is able to,

- Monitor our home or office while we are away
- Support encrypted transmissions
- Send email and SMS when motion detected.

3.4 Questions

Question 1. What are functionalities of each layer of a computer network? Which one is directly related to our end users?

Question 2. How to set a laptop or mobile phone as a hotspot?

Question 3. How to set a multimedia streaming server?

Question 4. What are the typical attacks of a computer network? What hardware and software are used to anti-attacks? What is VPN? Can you give an example?

Question 5. Which algorithms are often employed for image and video scrambling and descrambling?

References

1. Anderson R (2008) Security engineering. Wiley, Indianapolis
2. Argano M, Gidwani T, Yan W, Issa F (2012) A comprehensive survey of event analytics. Int J Digit Crime Forensics 4(3):33–46
3. Bertino E, Sandhu R (2005) Database security-concepts, approaches, and challenges. IEEE Trans Dependable Secur Comput 2(1):2–19
4. Chen D, Tao Z, Ma G (2008) Application of wireless sensor networks for monitoring emergency events. In: IEEE conference on sensors, pp 518–521
5. Ciampa M (2009) Security + guide to network security fundamentals. Cengage Learning, Boston
6. Furht B, Muharemagic E, Socek D (2005) Multimedia encryption and watermarking. Springer, Boston
7. Gollmann D (2009) Computer security. Wiley, Hoboken
8. Gu S, Han Q (2006) The application of chaos and DWT in image scrambling. In: Machine learning and cybernetics, pp 3729–3733
9. Hitaj B et al (2018) PassGAN: a deep learning approach for password guessing. arXiv:1709.00440
10. Kizza J (2010) Computer network security. Springer, New York
11. Klette R (2014) Concise computer vision. Springer, London
12. Kurose J, Ross K (2003) Computer networking. Addison-Wesley, Boston
13. Leon-Garcia A, Widjaja I (2000) Communication networks. Mc Graw Hill, Singapore
14. Liu Z, Yang B, Yan W (2017) Image encryption based on double random phase encoding. In: IEEE IVCNZ, pp pp 1–6
15. Menezes A, Oorschot P, Vanstone S (1997) Handbook of applied cryptography. CRC Press, Boca Raton
16. Socek D, Kalva H, Magliveras S, Marques O, Culibrk D, Furht B (2007) New approaches to encryption and steganography for digital videos. Multimed Syst 13(3):191–204
17. Stallings W (2006) Cryptography and network security, 4th edn. Upper Saddle River, Pearson (International Edition)
18. Stallings W (2015) Operating systems internals and design principals. Pearson Education Limited, New Jersey
19. Stinson D (2006) Cryptography theory and practice, 3rd edn. Chapman & Hall/CRC, Talor & Francis Group, Boca Raton

20. Washburn K, Evans J (1996) TCP/IP running a successful network. Pearson Education, Harlow
21. Wee J, Apostolopoulos G (2001) Secure scalable streaming enabling transcoding without decryption. In: IEEE ICIP, pp 437–440
22. Yan W, Kankanhalli M (2008) Progressive audio scrambling in compressed domain. IEEE Trans Multimed 10(6):960–968
23. Yan W, Kankanhalli M (2015) Face search in encrypted domain. In: Pacific-rim symposium on image and video technology, pp 775–790

4 Surveillance Data Analytics

4.1 Object Computable Features

In digital image and video processing, we usually start from a single pixel of a raster display; pixel color is represented by the bits composed of binary value '0' or '1'; usually, we use red(R), green(G), and blue(B)—three channels to display one pixel; each independent channel has 8 bits or 1 byte. Sometimes, we also need the alpha channel for transparent or opaque display, which occupies another byte. Therefore, we totally use at least 4 bytes or 32 bits to display one real-color pixel in general.

- **Image Color**

Image colors are comprised of binary color, pseudocolor, grayscale color, and real color. Binary color only uses '0' or '1' to present the colors of a binary image, namely white and black. Pseudocolor using an index number presents a pixel color since the display cache could not hold too many colors simultaneously at early stage of display technology. Grayscale refers to the three channels RGB having the same intensity, namely $I = R = G = B$, $I = 0, 1, 2, \ldots, 255$, simply,

$$I \approx \lfloor \frac{R+G+B}{3} \rfloor \tag{4.1}$$

where $\lfloor \cdot \rfloor$ is the floor function. Regularly, converting a color to a grayscale image, we use the Eq. (4.2)

$$I = \lfloor \alpha \cdot R + \beta \cdot G + \gamma \cdot B \rfloor \tag{4.2}$$

where $\alpha + \beta + \gamma = 1.0$, $\alpha, \beta, \gamma \in (0, 1.0)$ for example, in terms of the CIE 1931, the linear luminance I is given by $\alpha = 0.2126$, $\beta = 0.7152$, and $\gamma = 0.0722$.

Real color means so many colors are shown on a display and our eyes could not distinguish differences between the colors from real world and ones from our display. At present, many consumer products present retina display no matter in pixel size or

W. Q. Yan, *Introduction to Intelligent Surveillance*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-10713-0_4

colors. Displaying more colors means we have to use a larger buffer to hold all colors of one image or one frame of a video, and all the pixel colors should be popped up on a screen simultaneously; that means in the display there is no time delay between the first pixel to be turned up till the last one.

- **Color Spaces: RGB, YCbCr, and YUV**

RGB is one of the fundamental color schemes in raster display; the three channels are closely adhered together linearly. Usually, we map colors from one space to another, such as the color space YCbCr, YUV, HSV, etc.; the colors in different space show their distinct properties. After the color space conversion, the color distribution and the linear relationship between these color channels are much reasonable. The color conversion between the different spaces is unique like Eq. (2.11) in image compression. For example, the conversion equations between RGB and Y′UV color spaces are shown in Eqs. (4.3) and (4.4),

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.3}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix} \tag{4.4}$$

- **Histogram**

Histogram is a statistical concept which means how many pixels have the same intensity for a bin, and the bin number is 8, 16, 64, or 256 usually. For a RGB image, we make the statistics in each channel respectively. The statistical result shows the distribution of pixel intensity of one channel in an image as a bar diagram; the concept "entropy" is derived from histogram as Eq. (4.5) that shows information capacity of this image [12].

For the same scene with a slight change, the color distributions of each image should not have too many changes; therefore, the histograms have not too many differences. The histogram is regarded as one of the very robust statistical features in image search and retrieval or object tracking [9].

$$E(I) = -\sum_{i=1}^{b} h_i \cdot \ln h_i \tag{4.5}$$

where b is the bin number of the image histogram (Fig. 4.1).

If an image or some parts of this image are too dark or bright, we are able to get the "hidden" visual information using *Histogram Equalization*. Suppose the least bin number is $b_{\min}$, the greatest bin number is $b_{\max}$, all the bin numbers between $b_{\min}$ and $b_{\max}$ hence are calculated:

$$b' = \frac{b_i - b_{\min}}{b_{\max} - b_{\min}} \cdot B; \tag{4.6}$$

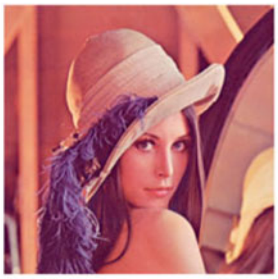
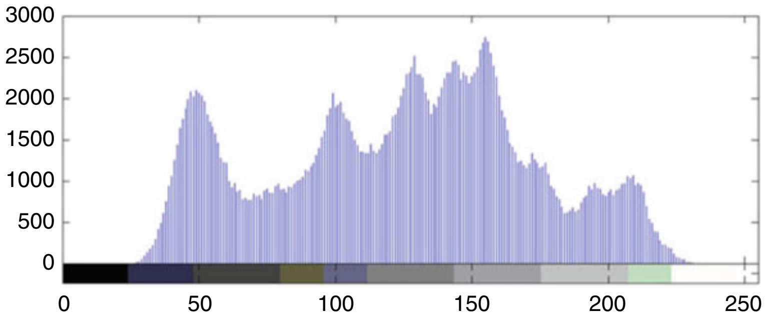



Fig. 4.1 Histogram of the grayscale image Lena (512 × 512)

$$h(b') = h(b_i); \tag{4.7}$$

Equations (4.6) and (4.6) show the pixel colors only within a specific interval which have been stretched to the full range from 0 to B.

For histograms $Q = (H_1^q, H_2^q, \ldots, H_b^q)$ and $D = (H_1^d, H_2^d, \ldots, H_b^d)$, the inner product or dot product is,

$$< Q, D >= Q \cdot D = |Q| \cdot |D| \cos(\alpha) = \sum_{i=1}^{b} (H_i^q \cdot H_i^d) \tag{4.8}$$

$$\cos(\alpha) = \frac{< Q, D >}{|Q| \cdot |D|} = \frac{Q \cdot D}{|Q| \cdot |D|} = \frac{\sum_{i=1}^{b} (H_i^q \cdot H_i^d)}{\sqrt{\sum_{i=1}^{b} (H_i^q)^2} \sqrt{\sum_{i=1}^{b} (H_i^d)^2}} \tag{4.9}$$

The histogram distance is,

$$\Delta H(Q, D) = \sum_{i=1}^{b} |H_i^q - H_i^d| \tag{4.10}$$

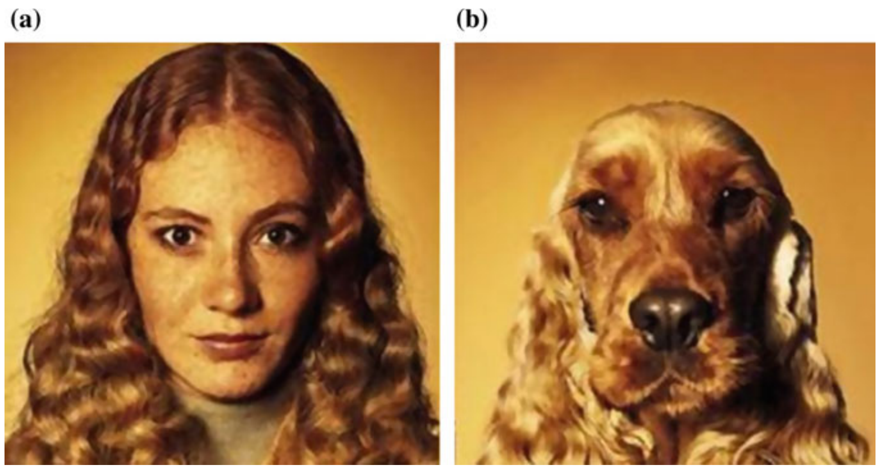


Fig. 4.2 An example of two images with the same histogram distributions

The normalized distance is,

$$\Delta\overline{H}(Q, D) = \sum_{i=1}^{b} \frac{|H_i^q - H_i^d|}{\max(H_i^q, H_i^d)} \cdot H_i^q \tag{4.11}$$

For two images having different contents but with the same histogram as shown in Fig. 4.2a, b, we need to utilize further fine-grained approaches to find the differences between them.

- **LBP**

The local binary pattern (LBP) also examines spatial relationship of pixels for texture analysis where the adjacent [71] pixels are converted to binary codes by using grayscale value of the center pixel as a threshold. Accordingly, LBP histogram was adopted as a feature descriptor of texture feature [79].

In MATLAB, LBP feature vector returns as a $1 \times n$ vector of length n representing the number of features which depends on the number of cells in an image. The function partitions an input image into nonoverlapping cells. When the cell size is increased, local details are lost.

- **Texture**

An image usually has texture from clothes, curtain, carpet, etc. Texture analysis of an image refers to characterize the regions of images by using texture content [65,91]. The texture analysis quantifies intuitive qualities described by terms such as rough, smooth, silky, or bumpy as a function of spatial variation in pixel intensities. The gray-level co-occurrence matrix (*GLCM*) considers the spatial relationship of pixels in examining texture. The GLCM functions characterize the texture of an image by calculating how often pairs of pixel with specific values in a specified spatial relationship occurred in an image. An example of GLCM-based flame detection is shown in Fig. 4.3.

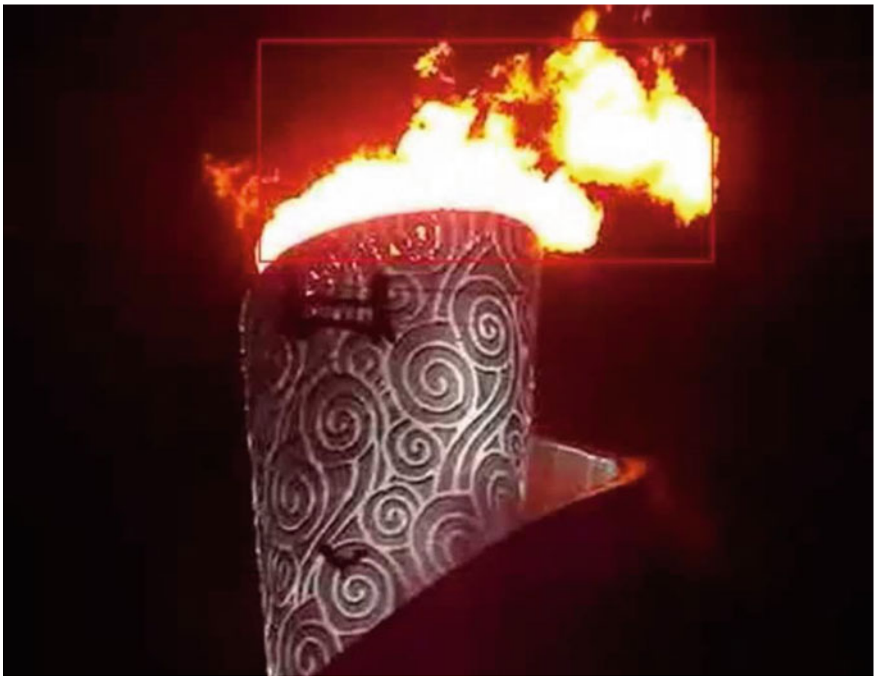


Fig. 4.3 An example of flame detection

In Fig. 4.3, we firstly use a model concerned with histogram and color to depict the candidate fire flame region. Flame region is a group of pixels (blob) with a high-intensity value. After the training process, we could segment this kind of regions by using a threshold with the assistance of color histogram. Meanwhile, we also observe that a flame region differs from that of others [41]. The most significant one is the color feature. Although it will cause a lot of errors, it is sufficient to detect all kinds of flames. We can set other constraints to define the flame regions so as to remove the wrongly detected ones. In the end, the inner part is bright which is close to white and the verge has various colors. This feature is a powerful rule that can improve the detection performance dramatically.

Secondly, we deploy Gabor texture, Tamura texture, and GLCM texture to verify the detected regions by using texture analysis. The details of the detected texture are defined by using different types of flames because the previous methods may be confused with the target which has the similar colors or other kinds of lighting conditions.

On the basis of Fig. 4.3, the method is highly flexible and robust which can be applied to almost any condition as long as the camera is fixed and stabilized. This method is reliable in all kinds of complex conditions and is fairly adaptive at detecting fire-color objects. According to the above descriptions, it is able to be applied to a multitude of environments for fire flame detection.

In MATLAB, GLCM texture feature is presented by:

- *Contrast*. A measure of intensity contrast between a pixel and its neighbor over the whole image which calculates the local variations in the gray-level co-occurrence matrix

$$c_t = \sum_i \sum_j (i-j)^2 \cdot I(i,j). \tag{4.12}$$

- *Correlation*. A measure of how a pixel is correlated to its neighbor over the whole image which shows the joint probability occurrence of the specified pixel pairs.

$$c_r = \sum_i \sum_j \frac{(i-\mu_i)(j-\mu_j)}{\sigma_i \sigma_j} \cdot I(i,j) \tag{4.13}$$

where μ_i and μ_j are means, σ_i and σ_j are variances.
- *Energy*. A sum of squared elements which provide the total of squared elements in GLCM,

$$e = \sum_i \sum_j I(i,j)^2 \tag{4.14}$$

- *Homogeneity*. A measure of the closeness which evaluates closeness of the element distributions in GLCM to its diagonal

$$h = \sum_i \sum_j \frac{I(i,j)}{1+|j-i|}. \tag{4.15}$$

Tamura texture. A texture pattern is described by using coarseness and brightness; the values are put into a vector or used as elements of a vector.

Gabor texture. A texture pattern is described by using those coefficients of Fourier transforms from different layers.

2D Fourier transform maps a scalar image I from spatial domain into a complex-valued Fourier transform $\mathscr{F}$ on frequency domain.

$$\mathscr{F}(u,v) = \frac{1}{W \cdot H} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} I(x,y) \exp\left[-i2\pi \left(\frac{x \cdot u}{W} + \frac{y \cdot v}{H}\right)\right] \tag{4.16}$$

where $u = 0, \ldots, W-1$ and $v = 0, \ldots, H-1$, $i = \sqrt{-1}$ as the imaginary unit of complex numbers, W is the image width, and H is the image height, respectively.

Inverse 2D DFT maps a Fourier transform $\mathscr{F}$ on frequency domain back into the spatial domain

$$I(x,y) = \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} \mathscr{F}(u,v) \exp\left[i2\pi \left(\frac{x \cdot u}{W} + \frac{y \cdot v}{H}\right)\right]. \tag{4.17}$$

There are three properties of the DFT [54]:

- Symmetry property: $\mathscr{F}(W-u, H-v) = \mathscr{F}(-u,-v) = \mathscr{F}(u,v)^{\star}$
- Direct current/mean: $\mathscr{F}(0,0) = \frac{1}{W \cdot H} \sum\limits_{x=0}^{W-1} \sum\limits_{y=0}^{H-1} I(x,y)$
- Parseval's theorem: $\frac{1}{|\Omega|} \sum\limits_{\Omega} |I(x,y)|^2 = \sum\limits_{\Omega} |\mathscr{F}(u,v)|^2$

where '$\star$' is the conjugate of complex numbers and Ω is a region of the image block.

Examples of Fourier transform for images using MATLAB are shown in Fig. 4.4, where the left column lists all images and the right column shows the transformed image on frequency domain, correspondingly.

Fourier transform maps an image from a real number space to a complex number one (with real and imaginary parts). We put the coefficients from different layers together to form a feature vector. The vector is one of the computable features of an object. The Fourier transform of Gabor filter's impulse response is the convolution between Fourier transform of the harmonic function $I' = I \star C(\cdot)$ and Fourier transform of Gaussian function [50],

$$C(x,y;\lambda,\theta,\psi,\sigma,\gamma) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\exp\left[i\left(2\pi\frac{x'}{\lambda}+\psi\right)\right] \tag{4.18}$$

where $x' = x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$, I is the image used for convolution operation.

In Eq. (4.18), λ represents wavelength of the sinusoidal factor, θ represents orientation of the normal to the parallel stripes of a Gabor function, ψ is phase offset, σ is sigma/standard deviation of the Gaussian envelope, γ is spatial aspect ratio and specifies ellipticity of the Gabor function.

- **Edge/Shape**

In computer vision and digital image processing, we usually need to extract the edges or shapes of objects from an image [61]. The Canny (5×5), Sobel (3×3), and Roberts (2×2) operators usually help us extract edges of an object because at the regions near edges, colors usually change very dramatically [39,54]. The operators are based on gradient or color changes of the designated regions. The magnitude usually is calculated as $G = \sqrt{G_x^2 + G_y^2}$; the phase is noted as $\theta = \arctan(G_y, G_x)$.

(1) **Canny operator**

$$G_x = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{4.19}$$

$$G_y = \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.20}$$

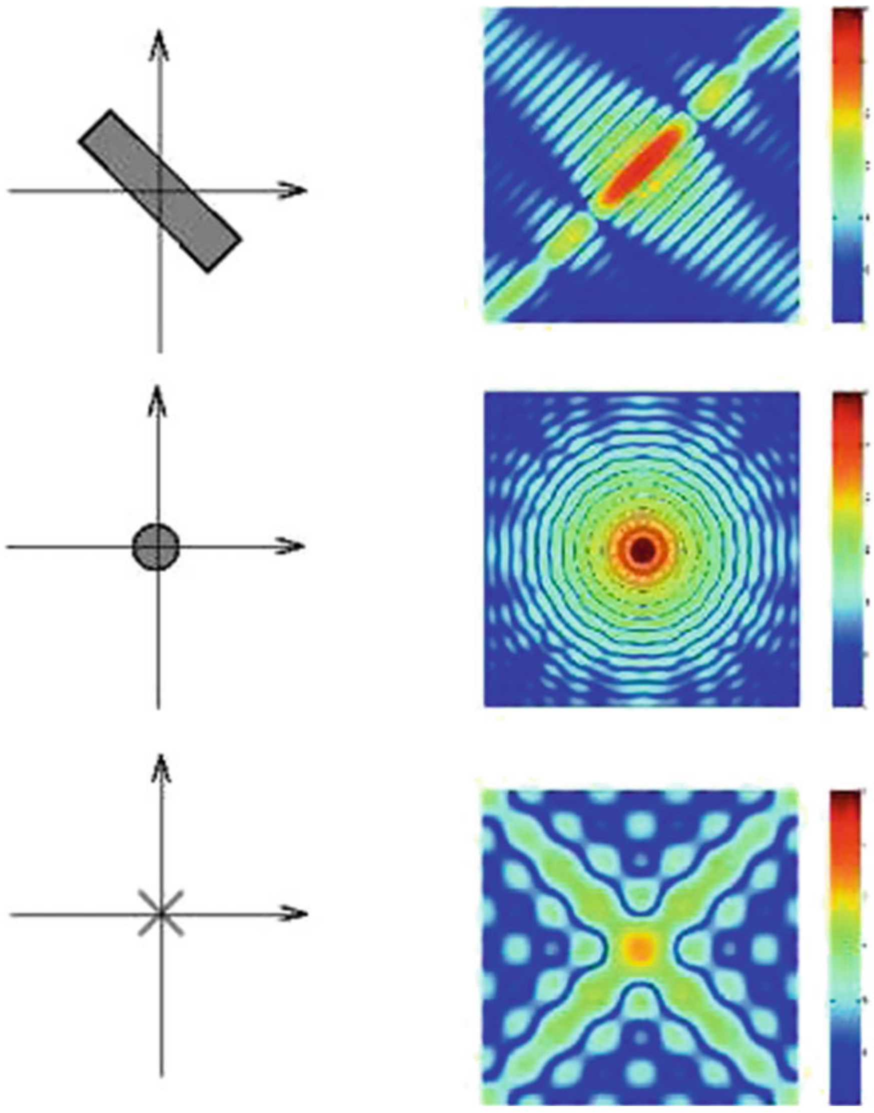


Fig. 4.4 Examples of Fourier Transform using MATLAB

An example of edge extraction using Canny operator from OpenCV is shown in Figs. 4.5 and 4.6.
(2) **Sobel operator**

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.21}$$

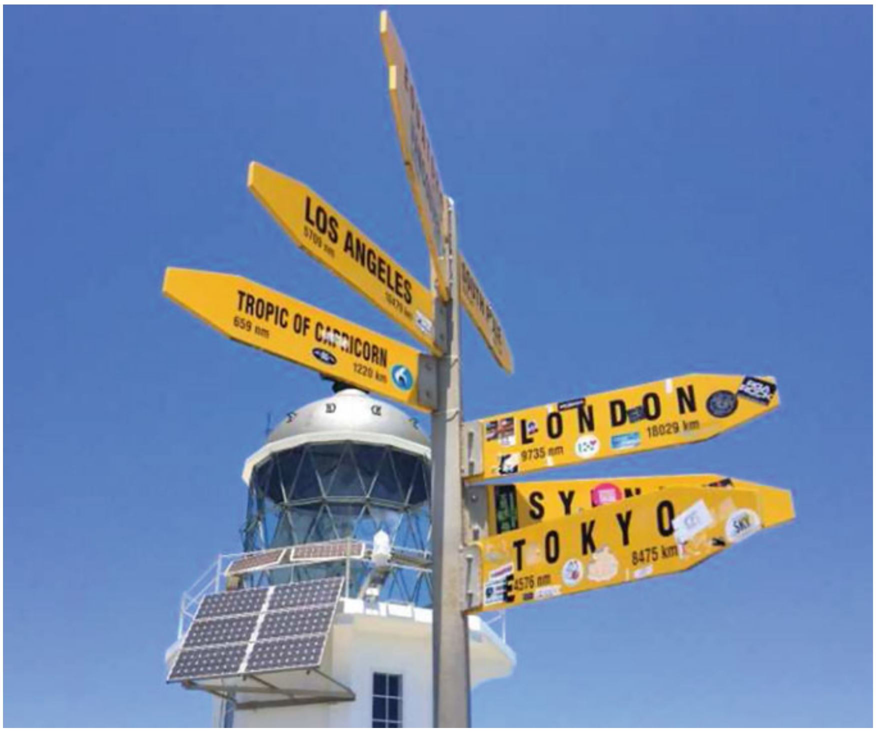


Fig. 4.5 A color image for Canny edge extraction

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.22}$$

(3) **Robert operator**

$$G_x = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \tag{4.23}$$

$$G_y = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

- **Gradient**

Gradient reflects the gradual changes of pixel colors from one region to another, it is the steepest descent direction, gradients usually include three types: horizontal, vertical, and both.

$$\nabla = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (\nabla_x, \nabla_y) \tag{4.24}$$

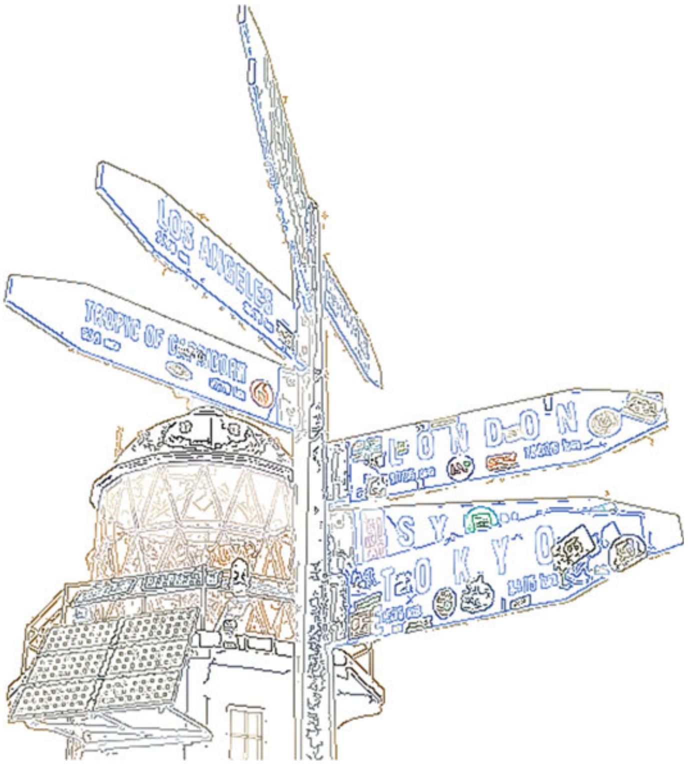


Fig. 4.6 Image after using Canny edge extraction from OpenCV

Discretely,

$$(\Delta I_x, \Delta I_y) = (I(x+1, y) - I(x, y), I(x, y+1) - I(x, y)) \tag{4.25}$$

Histogram of oriented gradients (HOG) is a feature descriptor in computer vision and digital image processing for the purpose of object detection [54]. The technique counts occurrences of gradient orientation in located portions of an image. The HOG feature shows the direction distributions with dramatic changes. A pixel usually is thought having four-connected or eight-connected directions, as shown in Fig. 4.7.

HOG is to derive a descriptor for a bounding box of an object candidate which applies intensity normalization and a smoothing filter to the given image window I meanwhile derives the gradient magnitudes and angles for each pixel. A magnitude map I_m and an angle map I_a will be generated. In order to obtain a voting vector, maps I_m and I_a are employed to calculate magnitude values into direction bins. Normalized voting values are used for generating a descriptor so as to augment all block vectors consecutively and produce the final HoG descriptor. Figure 4.8 is an example of HOG features that are plotted over the original image.

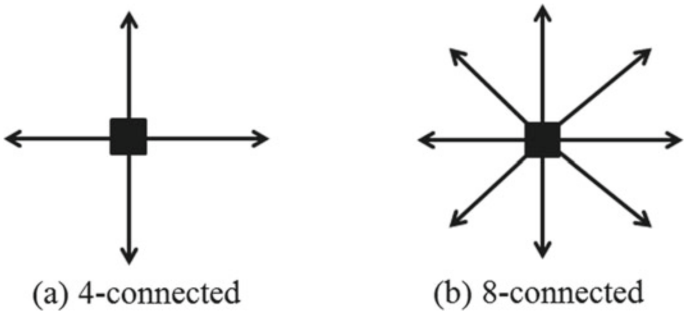


Fig. 4.7 Two typical connections

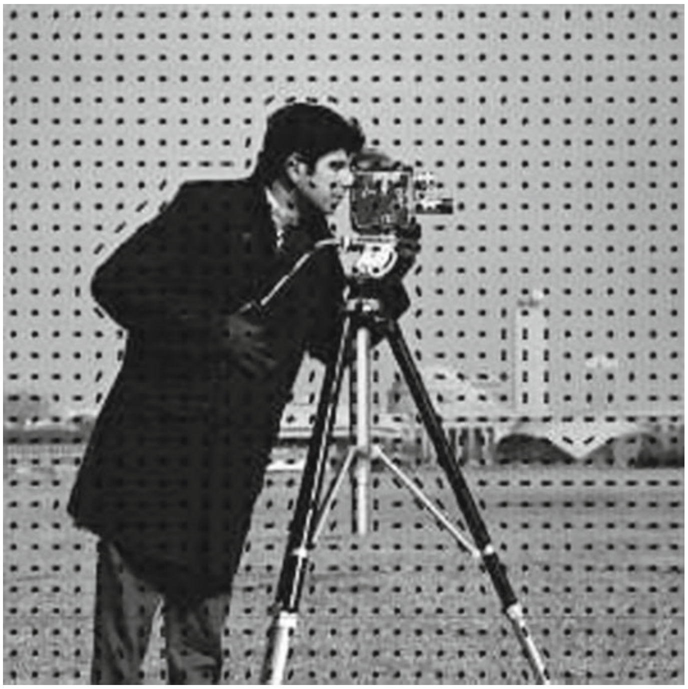


Fig. 4.8 Plotting HOG features over the MATLAB image: cameraman

- **Moments**

If we have a polygon to describe the shape of an object, we find the centroid coordinates by using average shown in Eq. (4.27). The moments include raw moments, central moments, scale-invariant moments, rotation-invariant moments, etc.; the central moments are calculated by using Eq. (4.28).

(1) **Raw moments**

$$M_{pq} = \sum_x \sum_y I(x, y) \cdot x^p \cdot y^p \tag{4.26}$$

(2) **Central moments**

$$x_c = \frac{1}{N} \sum_{i=1}^{N} x_i, \quad y_c = \frac{1}{N} \sum_{i=1}^{N} y_i \tag{4.27}$$

$$\mu_{pq} = \sum_x \sum_y I(x, y) \cdot (x - x_c)^p \cdot (y - y_c)^p \tag{4.28}$$

(3) **Scale-invariant moments**

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{\left(1+\frac{i+j}{2}\right)}} \tag{4.29}$$

(4) **Rotation-invariant moments**
The most frequently used moment is the Hu invariant moments,
$I_1 = \eta_{20} + \eta_{02}$

$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$

$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$

$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$

$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]+$
$(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$

$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]+$
$4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$

$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]-$
$(\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$

$I_8 = \eta_{11}[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] - (\eta_{20} - \eta_{02})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})$

- **Transforms**

Sine and cosine functions construct an orthogonal function system; namely, m and n are integers,

$$< \sin(mx), \cos(nx) >= \int_{-\pi}^{\pi} \sin(mx) \cos(nx) dx = 0 \tag{4.30}$$

$$< \sin(mx), \sin(nx) >= \int_{-\pi}^{\pi} \sin(mx)\sin(nx)dx = \begin{cases} 0 & m \neq n \\ \pi & m = n \end{cases} \quad (4.31)$$

$$< \cos(mx), \cos(nx) >= \int_{-\pi}^{\pi} \cos(mx)\cos(nx)dx = \begin{cases} 0 & m \neq n \\ \pi & m = n \end{cases} \quad (4.32)$$

Therefore, Fourier expansion of a continuous function $f(x)$ belongs to L^p space $(p = 2)$ or L_2, namely $\int_{-\pi}^{\pi} |f(x)|^2 dx < \infty$

$$f(x) = a_0 + \sum_{n=1}^{\infty}[a_n \sin(nx) + b_n \cos(nx)] \quad (4.33)$$

We have $n > 0$,

$$\begin{cases} a_0 = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)dx \\ a_n =< f(x), \sin(nx) >= \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)dx \\ b_n =< f(x), \cos(nx) >= \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)dx \end{cases} \quad (4.34)$$

where $< \cdot >$ is the inner product or dot product. Therefore,

$$f(x) = a_0 + \sum_{n=1}^{\infty}[< f(x), \sin(nx) > \sin(nx) + < f(x), \cos(nx) > \cos(nx)] \quad (4.35)$$

Fourier expansion for function $f(x)$ converges, it equals to $f(x)$ wherever $f(x)$ is continuous [50].

As Euler's formula,

$$e^{i\cdot x} = \cos(x) + i \cdot \sin(x) \quad (4.36)$$

Hence,

$$\cos(x) = Re\{e^{i\cdot x}\} = \frac{e^{i\cdot x} + e^{-i\cdot x}}{2}; \sin(x) = Im\{e^{i\cdot x}\} = \frac{e^{i\cdot x} - e^{-i\cdot x}}{2}; \quad (4.37)$$

furthermore,

$$\cos(n \cdot x) = 2 \cdot cos[(n-1)\cdot x] \cdot cos(x) - cos[(n-2)\cdot x] \quad (4.38)$$

When $n = 2$,

$$\cos(2 \cdot x) = 1 - 2 \cdot cos^2(x) \quad (4.39)$$

The special example is that wavelet transform uses wavelet basis functions to construct their orthogonal function systems so as to decompose digital signals or images at multiple layers with multiple resolutions [99] which outperforms than DCT [76] and Fourier transform [72].

- **Local Features**

A corner in an image is given at a pixel where two edges of different directions intersect. Corners usually lie on high-contrast regions of the image. Relative positions between corners in the original scene should not be changed after a variety of transformations.

Corners are invariant to scaling, orientation, and distortions. The best match for each pair of corners is found by identifying its nearest neighbor of corners. The nearest neighbors are defined as the corners with the minimum distance from the given descriptor. If a pixel is inside an object, its surroundings (solid square) correspond to those of its neighbor (dotted square). This is true for adjacent pixels in all directions. If a pixel is on the edge of an object, its surroundings differ from its neighbors in one direction, but correspond to the surroundings of its neighbors in the other (perpendicular) direction. A corner pixel has surroundings which are different from all of its neighbors in all directions.

Therefore, a corner in an image I is found at a pixel where two edges of different directions intersect. Corner detection usually uses its eigenvalues of a Hessian matrix of a pixel p. If the magnitude of both eigenvalues is "large" in Eq. (4.40), we say the pixel is at a corner,

$$\mathbf{H}(p) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} \tag{4.40}$$

where partial derivatives are defined,

$$\frac{\partial I(x,y)}{\partial x} = I_x(x,y) = \lim_{\Delta x \to 0} \frac{I(x+\Delta x, y) - I(x,y)}{\Delta x} \tag{4.41}$$

$$\frac{\partial I(x,y)}{\partial y} = I_y(x,y) = \lim_{\Delta y \to 0} \frac{I(x, y+\Delta y) - I(x,y)}{\Delta y} \tag{4.42}$$

and,

$$\frac{\partial I_x(x,y)}{\partial x} = I_{xx}(x,y); \; \frac{\partial I_y(x,y)}{\partial x} = I_{yx}(x,y);$$

$$\frac{\partial I_x(x,y)}{\partial y} = I_{xy}(x,y); \; \frac{\partial I_y(x,y)}{\partial y} = I_{yy}(x,y);$$

Corner detection takes advantages of Harris detector with the cornerness measure [54],

$$\mathbf{G}(p,\sigma) = \begin{bmatrix} L_x^2(p,\sigma) & L_x(p,\sigma)L_y(p,\sigma) \\ L_x(p,\sigma)L_y(p,\sigma) & L_y^2(p,\sigma) \end{bmatrix} \tag{4.43}$$

furthermore,

$$L(p, \sigma) = [I * G_a]p(x, y) \tag{4.44}$$

where I is an image, and $G_a(a > 0)$ is a local convolutional function.

$$\aleph(p, a, \sigma) = \det(\mathbf{G}) - \alpha \cdot Tr(\mathbf{G}) = \lambda_1\lambda_2 - \alpha \cdot (\lambda_1 + \lambda_2)$$

where α is a constant. Corner points have large, positive eigenvalues and would thus have a large Harris measure. Corners are applied to object tracking as shown in Fig. 4.9.

Scale-invariant Fourier transform (SIFT) helps us find the corresponding keypoints from two similar images no matter how the images are zoomed. Keypoints have the steepest gradient than the points around it. These keypoints are very robust in describing objects no matter how we use them in object detection or tracking [90]. SIFT for image matching usually has to experience following steps: (1) corner detection is based on changes of the adjacent regions, the changes of corner points are different from the changes of their neighbors, namely inner points; (2) local neighborhood description is based on the circular region of a corner; (3) corner matching uses the distance of this circular region.

An example is shown in Fig. 4.10; 1993 and 1856 keypoints were detected from the grayscale ones of the left and right images, respectively; 134 matches were

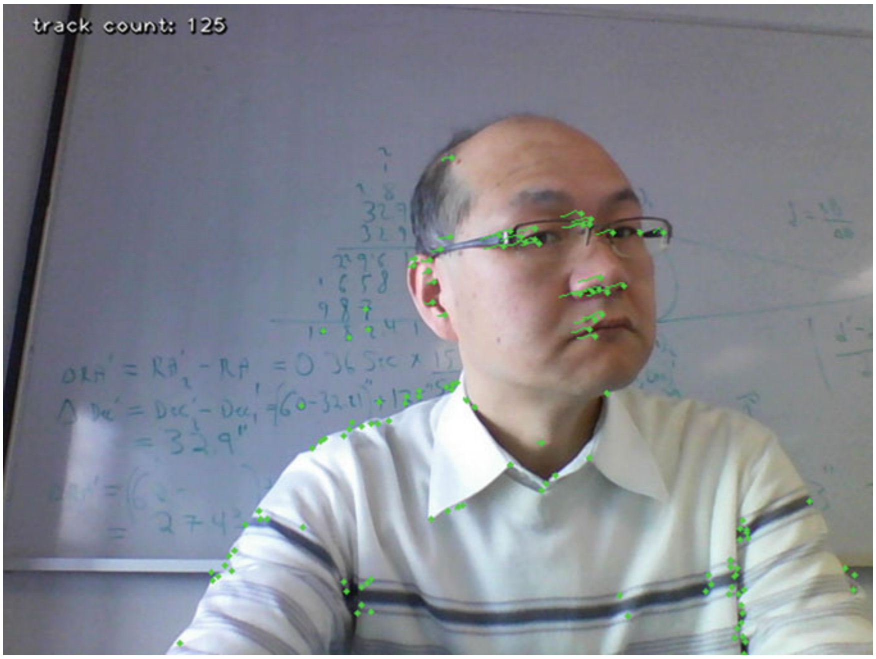


Fig. 4.9 Corners for object tracking

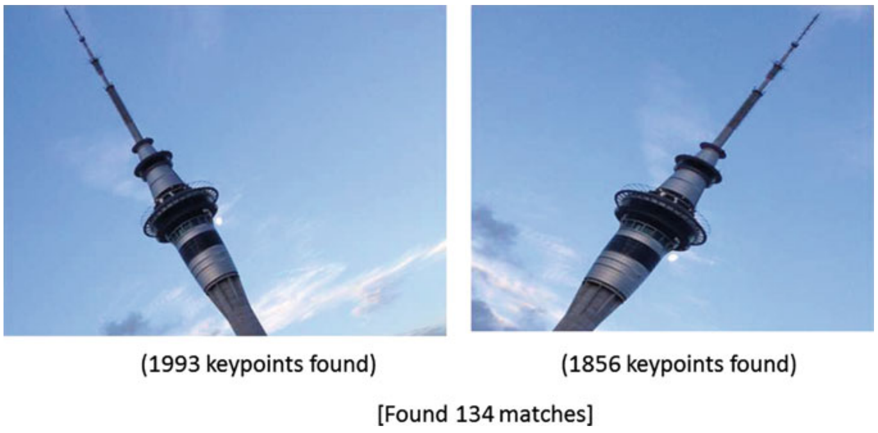


Fig. 4.10 Example of the SIFT

found when we used the SIFT algorithm provided from: http://www.cs.ubc.ca/~lowe/keypoints/.

Speeded-up robust feature (SURF) could improve the SIFT for seven times faster, random sample consensus (RANSAC) separates inliers from outliers through fitting or regression, the hypothetical inliers forms consensus set. Corners of digital images usually are regarded as hypothetical inliers; the inliers selected from the keypoints will be employed for image registration, mosaicking, etc. [54].

- **Distance between Visual Feature Vectors**

After selected all features, we integrate them as elements into one feature vector $V_f = [C, H, T, M, \ldots, W]$, where $C = (\mu, \sigma)$ is color information; μ is average of the colors; σ is the variance; $H = (h_1, h_2, \ldots, h_B)$ refers to histogram which depends on number of bins (B); $T = (c_1, c_2, \ldots, c_n)$ represents texture coefficients; $M = (m_1, m_2, \ldots, m_s)$ describes coefficients of moments; $W = (w_1, w_2, \ldots, w_s)$ depicts frequency coefficients after visual signal decomposition, etc. The visual feature vector V_f has been used for the further computing such as search, retrieval, mining, and reasoning [6].

If we have two computable feature vectors V_1 and V_2 to describe two objects respectively, the two objects are compared through distance of the two vectors $d = (V_1, V_2)$. The inner product between these two vectors usually is used to calculate the cosine value; the value decides how the two images are similar to each other in use of cosine function:

$$d(V_1, V_2) = \cos(\theta) = \frac{V_1 \cdot V_2}{|V_1| \cdot |V_2|} \tag{4.45}$$

If the value is $\cos(\theta) = 1$, $\cos(\theta) \in [0, 1]$, that means the two objects are the same; if $\cos(\theta) = 0$, directions of two visual vectors are perpendicular. Moreover, we calculate the distance by using Eq. (4.46)

$$d = \|V_1 - V_2\|_p = \left(\sum_{i=0}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}} \tag{4.46}$$

This is called p-norm distance, where $V_1 = (x_1, x_2, \ldots, x_n)$ and $V_2 = (y_1, y_2, \ldots, y_n)$.

In Eq. (4.46), if $p = 1$, the metric is called L_1 distance; if $p = 2$, the distance is named as L_2 or Euclidean distance; if $p \to \infty$, the distance is titled as ∞-norm distance L_∞=max$(|x_i - y_i|)$; if $p \to 0$, the distance is thus defined as 0-norm distance L_0=min$(|x_i - y_i|)$;

In mathematics, Euclidean distance is the "normal" distance between two vectors that one would measure it with a ruler. By using this distance, Euclidean space (or even any inner product space) becomes a metric space. Namely, for any $\mathbf{x}, \mathbf{y}, \mathbf{z} \in M$:

- $d(\mathbf{x}, \mathbf{y}) \geq 0$
- $d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$
- $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
- $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$

where d is a distance defined on a set M.

Other feature vector distances include Mahalanobis distance in statistics shown in Eq. (4.47),

$$d(\mathbf{V_1}, \mathbf{V_2}) = \sqrt{(\mathbf{V_1} - \mathbf{V_2})^T S^{-1} (\mathbf{V_1} - \mathbf{V_1})} \tag{4.47}$$

where S is the covariance matrix for vector $\mathbf{V_1}$ and $\mathbf{V_2}$.

In mathematics, Hamming distance between two strings of equal length is the numbers of positions at which the corresponding symbols are different. In another way, it measures the minimum number of substitutions which is required to change one string into the other, or the minimum number of errors during the transformation from one string into the other. Hamming distance between these two numbers is denoted by using the number of differences in the binary representation shown in Eq. (4.48).

$$d(s_1, s_2) = \sum_{i=1}^{n} \delta(c_{1i}, c_{2i}) \tag{4.48}$$

where $s_1 = (c_{11}c_{12} \cdots c_{1n})_2$ and $s_2 = (c_{21}c_{22} \cdots c_{2n})_2$, $\delta(c_{1i}, c_{2i}) = \begin{cases} 0 & c_{1i} = c_{2i} \\ 1 & c_{1i} \neq c_{2i} \end{cases}$

4.2 Object Segmentation

Object segmentation is the first step of object analytics which separates different regions encapsulated in an image. The first simple approach is for segmenting a color image by using statistical result in a color space as shown in Fig. 4.11. The result tells us the color components of this image and how the colors are distributed; color distribution of the sample image is shown in Fig. 4.12. Based on the color distribution from the sample image, we segment the image into different regions; each region is represented by using the color as its class.

Mathematically, we partition an image Ω into a finite number of regions S_i, $i = 1, \ldots, n$; the region S_i satisfies:

- $S_i \neq \emptyset, \forall i \in \{1, 2, \ldots, n\}$
- $\cup_{i=1}^{n} S_i = \Omega$
- $S_i \cap S_j = \emptyset, \forall i, j \in \{1, 2, \ldots, n\}$ with $i \neq j$

Approach I. Color-based segmentation using the color space; we need to:

- Acquire an image.
- Calculate sample colors in RGB color space.
- Classify each pixel using the nearest neighbor rule.
- Display results of the nearest neighbor classification.
- Mark each pixel using the color class number it belongs to.

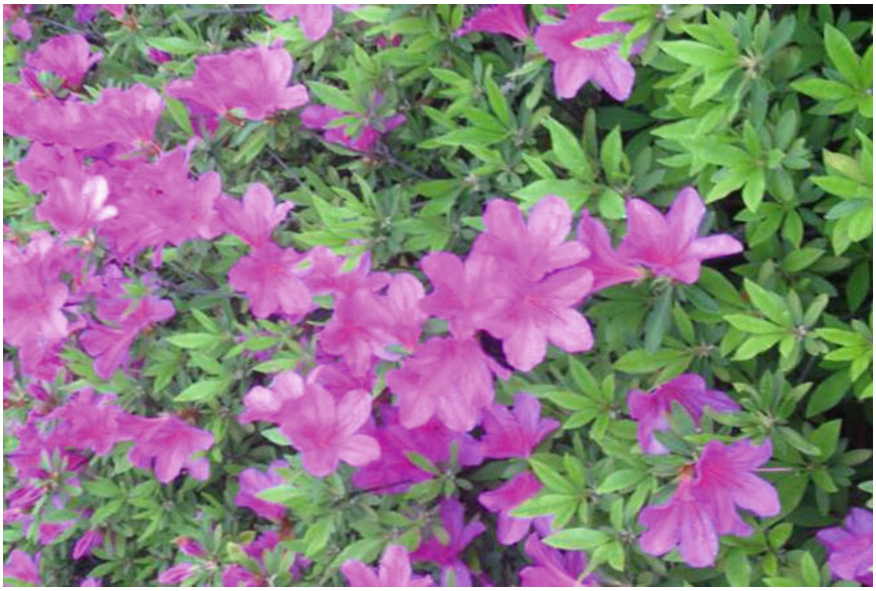


Fig. 4.11 A color image with flowers

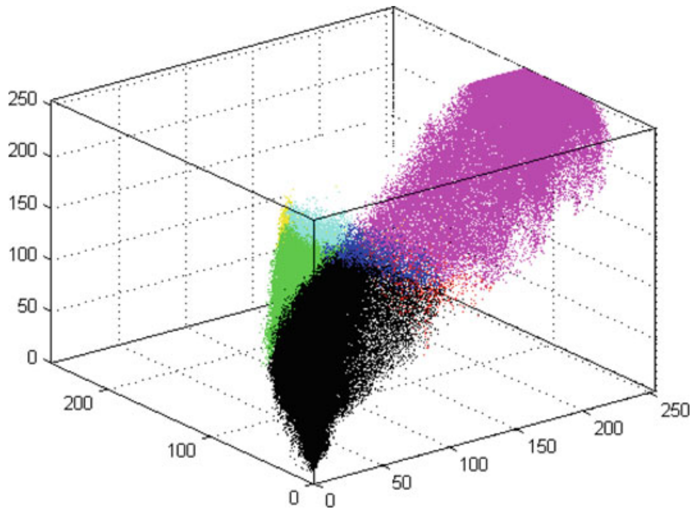


Fig. 4.12 Color statistical result for image segmentation

Approach II. Marker-controlled watershed segmentation
The second approach of image segmentation is to use watershed segmentation. The result is shown in Fig. 4.13, the steps of this algorithm are listed as below. We need to:

- Import in a color image as the input and convert it into grayscale.
- Use gradient magnitude as the segmentation function.
- Mark the foreground objects.
- Compute background markers.
- Visualize the result.

We provide a result of watershed image segmentation using the well-known OpenCV platform in Fig. 4.13. We manually assign regions on a color image, the watershed algorithm finds the similar regions using the gradient intensity. Once a peak reaches, the boundary is found, and the image will be segmented (Fig. 4.14).

Approach III. Texture segmentation using filtering
The steps of this kind of object segmentation are to,

- Import an image as the input.
- Create texture image.
- Create rough mask for the bottom texture.
- Use rough mask to segment the top texture.
- Display segmentation results.

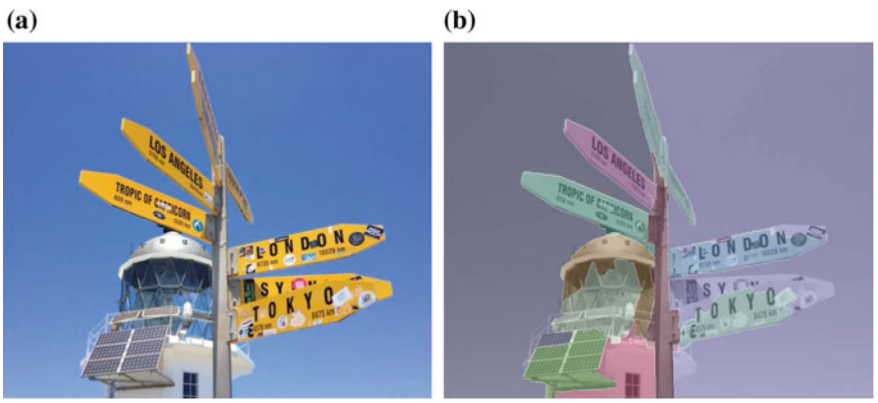


Fig. 4.13 Original image (**a**) and the segmented image (**b**) using the watershed segmentation algorithm of OpenCV

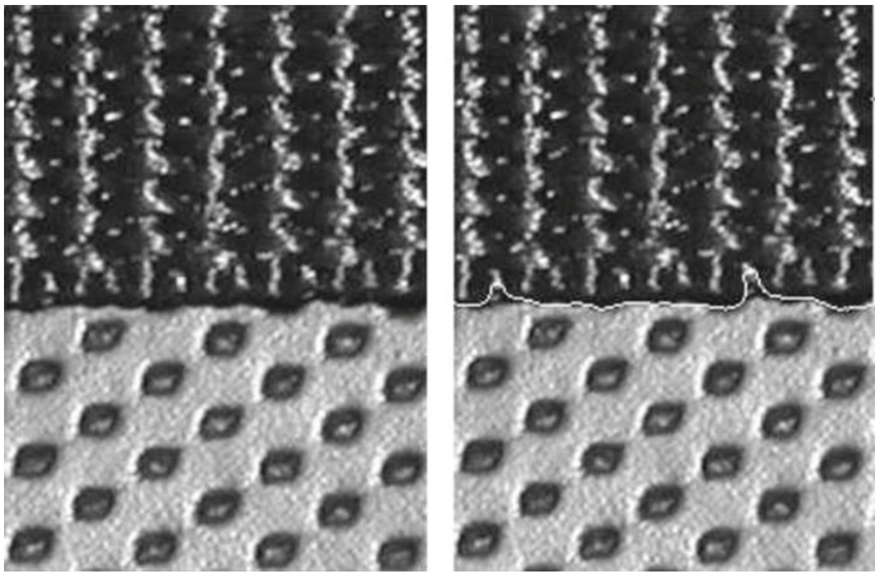


Fig. 4.14 Result of texture-based image segmentation using MATLAB

Approach IV. Markov random field (MRF) has been employed to image segmentation. Figure 4.15 is an example of image segmentation using the algorithm from the Web site: https://www.inf.u-szeged.hu/~kato/software/mrfdemo.html. (a) is a color image for segmentation, and (b) is the segmented image after assigned 5 classes for the image.

Markov random field (MRF) is defined as,

$$p(f_i | f_{\mathscr{S}-\{i\}}) = p(f_i | f_{\mathscr{N}_i}) \tag{4.49}$$

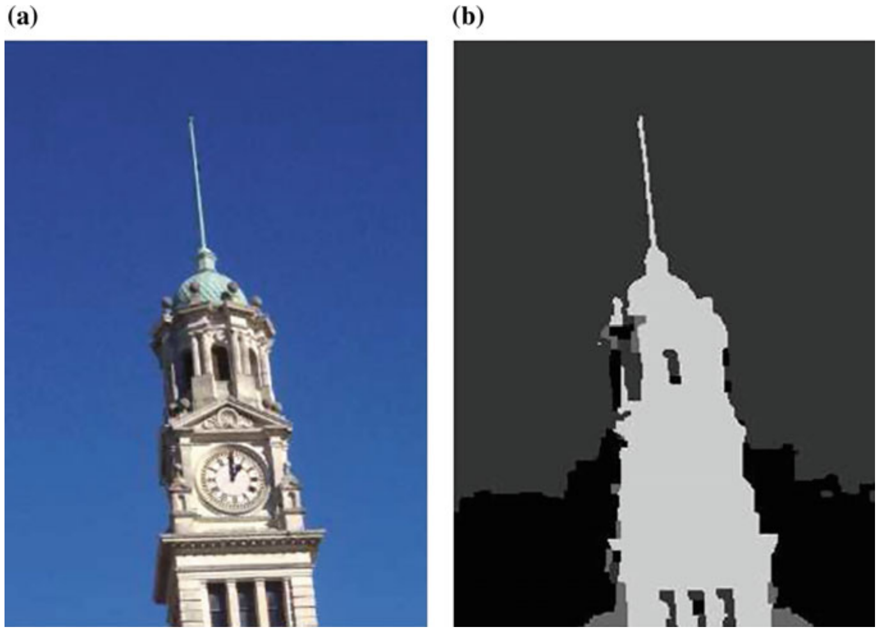


Fig. 4.15 Image segmentation using Markov random field

where $p(f) > 0, f_{\mathscr{N}_i} = \{f_{i'} | i' \in \mathscr{N}_i\}$. $\mathscr{N}$ is a neighborhood system and $f \in \mathbf{F}$ is the Markovianity. $\mathscr{N} = \{\mathscr{N}_i | \forall i \in \mathscr{S}\}$, $\mathscr{N}_i$ is the set of sites regarding to $i \notin \mathscr{N}_i$. Moreover, $i' \in \mathscr{N}_i \iff i \in \mathscr{N}_{i'}$ [60].

In the 2D case, $\mathscr{S}(i,j)$ is the lattice, and each site (i,j) has four neighbors

$$\mathscr{N}_{i,j} = \{(i-1,j), (i+1,j), (i,j-1), (i,j+1)\}. \tag{4.50}$$

For example, edges correspond to abrupt changes or discontinuities between neighboring areas. If $g(\cdot)$ is the truncated quadratic potential function, the pixel values near the edges are through minimizing

$$f^* = \arg\min_f E(\mathbf{F}) \tag{4.51}$$

and,

$$E(\mathbf{F}) = \sum_{i \in \mathscr{S}} (f_i - d_i)^2 - \sum_{i \in \mathscr{S}} \sum_{i' \in \mathscr{N}_i} \lambda_i g(f_i - f_{i'}) \tag{4.52}$$

4.3 Object Annotation, Classification, and Recognition

OpenCV is a computer vision platform which is able to detect and recognize moving objects [22,53]. A face recognition result based on OpenCV is demonstrated in Fig. 4.16 [94–96].

In face recognition, we need firstly to detect a human face using OpenCV, and then recognize the face using a classifier assisted by a training set. In face recognition, in order to increase accuracy, irrelevant moving objects in the background of the scene have to be removed. Precision and recall are calculated to compare the results before and after moving object removal (MOR) [13].

A regular definition of automatic license plate number recognition is to use digital image processing and computer vision to recognize each character in pictures containing plate number information automatically [2,4,14,18].

In general, basic modules of an automatic license plate number recognition system consist of plate detection [102], known as plate number extraction [64,68,74], character segmentation, and character recognition [19,55]. In order to implement the functions of each module, numerous techniques have been developed and presented. Leveraging the efficiency and costs, computer vision and digital image processing is broadly adopted in practice [3,4,15,18].

The target of character feature extraction [65] is to find a particular transition that can find out the minimum features which represent the original whole data. As

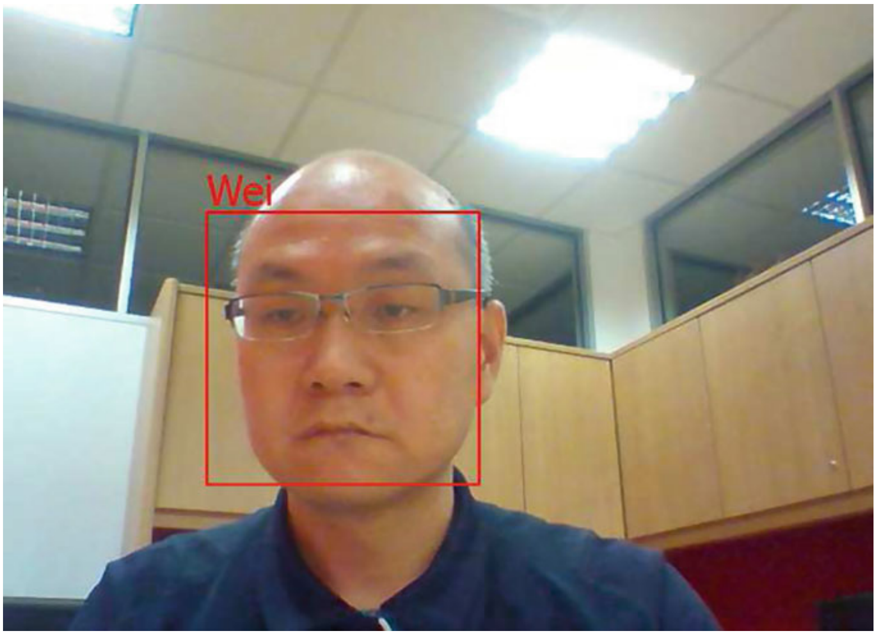


Fig. 4.16 An example of human face recognition

a result, the processing will be accelerated and the storage space will be minimized. The popular methods of feature extraction include zoning, projection histogram, distance profile, background directional distribution, and Fourier descriptors. Using a combination of features will increase efficiency of the character recognition. Hence, these features used in this example are extracted by zoning method and Fourier descriptors. The thought of zoning method is to calculate the pixel number in a particular area which is divided by using a grid. After obtained the numbers of pixels in each predefined area, a normalization will be conducted.

A desired automatic license plate recognition (LPR) [18,55] system adopts digital image processing techniques to locate and recognize the characters on car plate number and output the results as a textual string or other formats that can be easily understood in semantics [3,68,73]. The LPR system has been applied to various applications which require automatic control of the presence and identification of a motor car by using its plate number, such as stolen vehicles, automatic electronic toll collection (ETC), automated parking attendant, traffic ticketing management, security control, and others [5,11,51]. A plate number recognition system usually consists of five important modules: image acquisition, image preprocessing, plate number detection [64], image segmentation, and character recognition [3,4,98].

Plate number detection model plays a pivotal role in plate number recognition [55, 68]. If the position of a plate number in an image cannot be located accurately at very beginning, the following steps will not be continued correctly; thus, the plate number may appear at anywhere within the image [81,104]. To detect the location of a plate number [104], the particular features for plate number recognition have to be considered and determined [64]. The candidate plate number region is extracted and further verified if the plate number is correctly located in the image [61,65].

In plate number scan, a RGB image will be converted to HSV color space [58, 88,97,100]. A binary image will be obtained by detecting red pixels in HSV color space in accordance with prior knowledge of hue and saturation of the red color. After that, morphological operations such as closing and erosion are applied to filter out the noises [63,92,103]. The small redundant regions will be omitted by using image opening operation.

Normally, the two red regions indicated taillights of a car will be displayed in a filtered binary image. The two red taillights should appear at the same vertical level. Hence, the conditions are set to examine whether the detected region is required; the decision condition is whether the distance between two centroids of the detected areas is big enough. If the value is less than the predefined threshold, the detected regions are two taillights in the same line [94–96].

Template matching is defined as a technique in digital image processing to match a given image using multiple templates [35]. The plate number recognition is to compare each given image and the templates so as to find out which template is the best to match the given image [98,101]. There is a slew of matching functions which are employed to measure the similarity between the given image and the templates. The general matching functions include sum of square differences (SSD), normalized cross correlation (NCC), and mean absolute difference (MAD).

$$d_{SSD}(I, I_T) = \sum_{i=1}^{W}\sum_{j=1}^{H}[I(i,j) - I_T(i,j)]^2 \tag{4.53}$$

where I is the given image, I_T is the template, and W and H are the height and width, respectively.

$$d_{MAD}(I, I_T) = \frac{1}{W \times H}\sum_{i=1}^{W}\sum_{j=1}^{H}[I(i,j) - I_T(i,j)]^2 \tag{4.54}$$

$$d_{NCC}(I, I_T) = \frac{1}{W \times H}\sum_{i=1}^{W}\sum_{j=1}^{H}\frac{[I(i,j) - \overline{I}(i,j)] \cdot [I_T(i,j) - \overline{I_T}(i,j)]}{\sigma_I \sigma_{I_T}} \tag{4.55}$$

where $\overline{I}$ and $\overline{I_T}$ are the means, σ_I and σ_{I_T} are the standard deviations for image I and template I_T respectively.

In real instances of license plate recognition [59], when a road is uneven with bends, a vehicle will be running with shaky. Consequently, the plate is also unstable and tilted with rotations. In this case of plate correction, Hough transform was used to detect the straight lines of characters from the acquired plate images; the tilt angle of a straight line in the horizontal direction was calculated, and the angle of rotation was detected. The image was rotated back to the horizontal direction using the angle detected. After the rotation, a bilinear interpolation will be applied to reconstruct the corrected image.

The methods of license plate character recognition were based on template matching and artificial neural networks. In this case of plate recognition, the genetic algorithm (GA) was used as a search method, the weight of each network is a gene position of the chromosome, the length of chromosome corresponds to all values of the network; the accuracy of GNN-based recognition is higher than that of the backpropagation (BP) algorithm.

Artificial neural networks (ANNs) and deep learning (e.g., CNN and RNN) [28, 56,57] have been employed to character recognition [43,69]. The neural network was developed with multilayer feedforward backpropagation algorithm using one hidden layer [16]. ANNs have been applied to classify the number plate from color images. An example of automatically number plate recognition is shown in Fig. 4.17 [94–96] (Fig. 4.18).

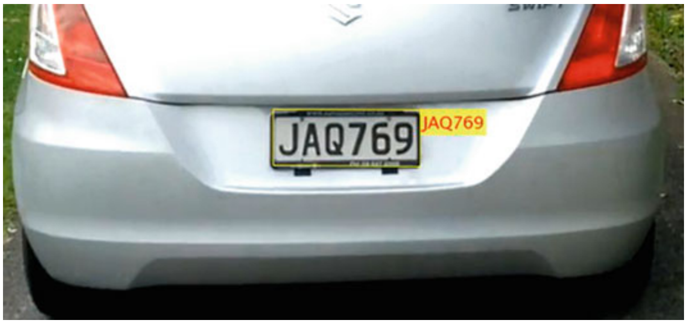


Fig. 4.17 An example of car back number plate recognition

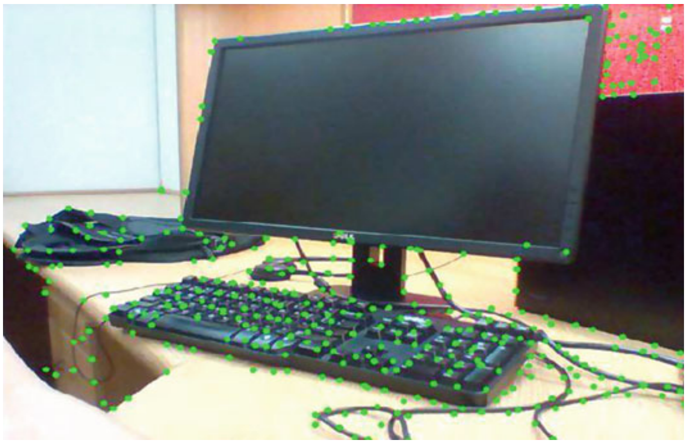


Fig. 4.18 An example of point tracking

4.4 Object Locating and Tracking

Assume we have segmented all visual objects from an image; we need to locate and track the objects [30,61]. The tracking algorithms include point tracking, kernel tracking, silhouette (contour and shape) tracking, etc. [90]. In point tracking, deterministic and probabilistic approaches are employed. Probabilistic approaches are statistically based.

Kernel tracking is template- and multiview-based. The template matching is based on calculations pixel by pixel. In general, approaches of this kind are very slow in computing. Multiview algorithms include two parts, namely view subspace and classifier [7].

Silhouette tracking includes contour evolution and shape matching. Contour evolution refers to state-space methods and direction minimization. Direction minimization encompasses variational approach and heuristic approach.

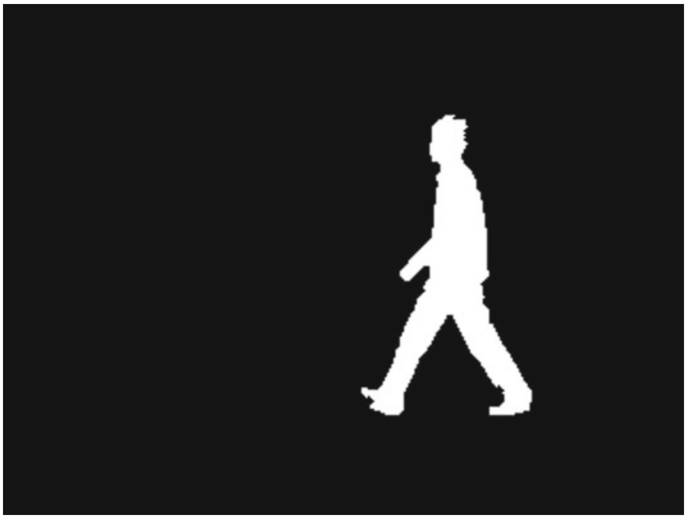


Fig. 4.19 An example of silhouette related to human gait

Silhouette is represented as a solid shape of a single color; an example of silhouette is shown in Fig. 4.19. The interior of a silhouette is featureless, and the whole is typically presented on a light background. It was used to describe cut papers which were stuck to a backing in a contrasting color and often framed.

Blob is a very similar concept in computer vision. Informally, a blob is a region of an image in which some properties are constant or approximately constant; all the points in a blob can be considered to be similar to each other. Two kinds of methods are used to detect blobs: differential methods and method-based local extrema.

MPEG videos already stored motion vectors in themselves which were calculated by using the technique called optical flow which is an approach to describe motions in a video [75]. The motion vector includes motion estimation of objects; therefore, we are able to use the forward and backward motion vectors to track a moving object [67]. Because the motion vectors are already saved in our video footages during the compression time, that will greatly save our computing time.

Suppose optical flow $\mathbf{u} = [u, v]^\top$, visible replacement starts at $p = (x, y)$ and ends at $p = (x + u, y + v)$. Optical flow aims at 2D motion estimation. 2D motion vectors form a vector field; a motion vector field is dense if it contains motion vectors at all pixels; otherwise, it is sparse.

From Newton–Leibniz formula and Taylor expansion , we know if

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t)$$

then,

$$\delta x \cdot \frac{\partial I}{\partial x}(x, y, t) + \delta y \cdot \frac{\partial I}{\partial y}(x, y, t) + \delta t \cdot \frac{\partial I}{\partial t}(x, y, t) = 0$$

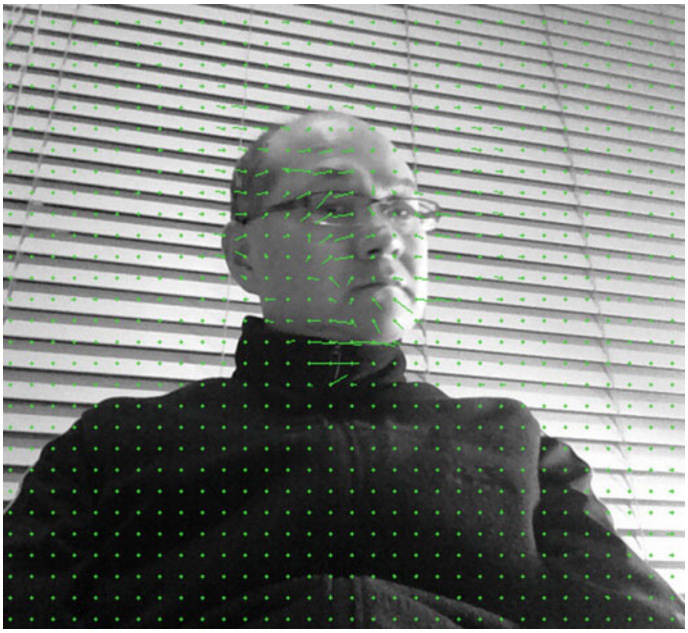


Fig. 4.20 Optical flow detection using OpenCV

hence,

$$\frac{\delta x}{\delta t} \cdot \frac{\partial I}{\partial x}(x, y, t) + \frac{\delta y}{\delta t} \cdot \frac{\partial I}{\partial y}(x, y, t) + \frac{\partial I}{\partial t}(x, y, t) = 0$$

If we assume,

$$\mathbf{u}(x, y, t) = (u(x, y, t), v(x, y, t)) = \left(\frac{\delta x}{\delta t}, \frac{\delta y}{\delta t}\right)$$

$$\mathbf{g}(x, y, t) = (I_x(x, y, t), I_y(x, y, t) = \nabla_{x,y} I$$

then,

$$\mathbf{u}(x, y, t) \cdot \mathbf{g}(x, y, t) = -I_t(x, y, t)$$

Namely, optical flow equation is,

$$\mathbf{u} \cdot \mathbf{g} = -I_t$$

where $\mathbf{g}$ is the gradient vector. Following this, the typical algorithms for seeking optical flow $\mathbf{u} = [u, v]^\top$ are Horn–Schunck algorithm and Lucas–Kanade algorithm; interested readers could find a computer vision book to understand the details. An example for calculating optical flow is shown in Fig. 4.20.

Typical algorithm for object tracking is mean shift which is based on template matching. Usually, we begin from the starting position of a model in the current frame; we search the neighborhood in next frame and find the best candidate by maximizing a similarity function (usually 10% step length will be taken). Once it is found, we use the best position as the new start, search for the next best region in the next frame, and so on; finally, we are able to get the goal of object tracking. The advantage of this algorithm is its simplicity and ease for implemented as others, but the disadvantage is its slowness and ineffectiveness.

The additional algorithms for object tracking also include Bayesian filtering, Kalman filtering [36], particle filtering [17,17], and data association.

The objective of Kalman filtering is "Noisy data in, hopefully less noisy data out." Therefore, for a linear dynamic system $\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x}$, $\mathbf{A}$ is a constant $n \times n$ matrix.

From $e^x = 1 + \sum_{i=1}^{\infty} \frac{x^i}{i!}$, we have the state transition matrix $\mathbf{F}_{\triangle t} = e^{\triangle t \mathbf{A}} = I + \sum_{i=1}^{\infty} \frac{\Delta t^i \mathbf{A}^i}{i!}$.

For a discrete system,

$$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \mathbf{w}_t \tag{4.56}$$

$$\mathbf{y}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}_t \tag{4.57}$$

where $\mathbf{B}$ is the control matrix, $\mathbf{u}_t$ is a system control vector, $\mathbf{w}_t$ is the noise vector, $\mathbf{H}$ is the observation matrix, $\mathbf{y}_t$ is the noise observations, and $\mathbf{v}_t$ is the noise observation vector.

Kalman filtering updates our knowledge based on experienced prediction errors and observations, we use the improved knowledge for reducing prediction error. In predict phase,

$$\begin{cases} \hat{\mathbf{x}}_{t|t-1} = \mathbf{F}\hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}\mathbf{u}_t \\ \mathbf{P}_{t|t-1} = \mathbf{F}\mathbf{P}_{t-1|t-1}\mathbf{F}^\top + \mathbf{Q}_t \end{cases} \tag{4.58}$$

In update phase,

$$\begin{cases} \mathbf{z}_t = \mathbf{y}_t - \mathbf{H}\hat{\mathbf{x}}_{t|t-1} \\ \mathbf{S}_t = \mathbf{H}\mathbf{P}_{t|t-1}\mathbf{H}^\top + \mathbf{R}_t \\ \hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\tilde{\mathbf{z}}_t \end{cases} \tag{4.59}$$

In optimal Kalman gain,

$$\begin{cases} \mathbf{K}_i = \mathbf{P}_{t|t-1}\mathbf{H}^\top\mathbf{S}_t^{-1} \\ \mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_{t|t-1} \end{cases} \tag{4.60}$$

The matrix

$$\mathbf{K}_i = \mathbf{P}_{t|t-1}\mathbf{H}^\top\mathbf{S}_t^{-1} \tag{4.61}$$

minimizes the mean square error $E[(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t})^2]$ which is equivalent to minimize the trace of $\mathbf{P}_{t|t}$. The matrix $\mathbf{K}_i$ is known as the *optimal Kalman gain.*

If we have the continuous model matrix **A** for the given linear dynamic process $\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x}$. The alternative model for predict phase is much straightforward,

$$\begin{cases} \dot{\hat{\mathbf{x}}}_{t|t-1} = \mathbf{A}\hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}\mathbf{u}_t \\ \mathbf{P}_{t|t-1} = \mathbf{A}\mathbf{P}_{t-1|t-1}\mathbf{A}^\top + \mathbf{Q}_t \end{cases} \tag{4.62}$$

A particle represents a feature in a multidimensional space, the dimensions of this space combine locations with descriptor values in a parameter vector. A particle filtering can track parameter vectors over time or within a space based on evaluating consistency with a defined model. A condensation algorithm is used to analyze a cluster of weighted particles for identifying a winning particle.

The iterative condensation process decides which of the randomly generated particles is taken as a result for the next image row. One iteration of condensation is called resampling, the goal is to improve the quality of the particles. A particle with a high weight is very likely to survive the resampling process. Resampling takes all the current weighted particles as input and outputs a set of weighted particles.

Image classifiers are grouped into two categories, namely learning-based or parametric classifiers such as artificial neural networks (ANNs) which rely on a training period and nonparametric classifiers such as k-nearest neighbor (k-NN) which do not require any training time, can handle a large number of classes, and can avoid the over-fitting problem [43,45,46].

The k-nearest neighbor classifier and other nonparametric classifiers have been made primarily due to performance diversity. Multilayer perceptron (artificial neural network)-based (parametric) classifiers outperform nearest neighbor, Bayesian and minimum-mean-distance (nonparametric) classifiers especially with noise. Learning-based classifiers including ANNs also outperform nonparametric classifiers with regard to error rate. An issue with ANNs is that high levels of accuracy come at the expense of considerable training time.

Support vector machine (SVM) is now generally thought as being superior to the single-layer ANN with regard to generalized accuracy in image classification. This refers to the SVM as having higher levels of accuracy when classifying an unknown test set.

4.4.1 Support Vector Machine (SVM)

In computer vision, unsupervised learning is to find hidden structure in unlabeled data. This distinguishes unsupervised learning from supervised learning and reinforcement learning. Supervised learning is a machine learning task of inferring a function from labeled training data. The training data consists of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples.

The typical supervised learning algorithms include support vector machine (SVM) and decision trees. For SVM,

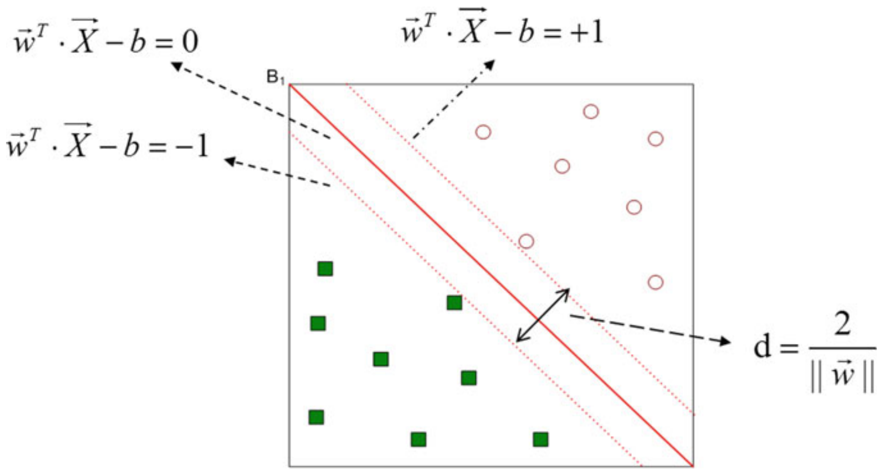

$\vec{w}^T \cdot \vec{X} - b = 0$
$\vec{w}^T \cdot \vec{X} - b = +1$
$\vec{w}^T \cdot \vec{X} - b = -1$
B1
$d = \frac{2}{\|\vec{w}\|}$

Fig. 4.21 Margin distance between two classes of SVM classification

Problem: Given a vector dataset, no matter how high the dimension is, how to find a linear hyperplane (decision boundary) that will separate the data?

Solution: It is better to find a hyperplane that maximizes the margin.

If a line is $W \cdot X - b = 0$, the margin will be between the two lines $W \cdot X - b = +1$ and $W \cdot X - b = -1$; the distance between the two parallel lines is $d = \frac{2}{\|W\|}$ as shown in Fig. 4.21.

The SVM is to maximize the margin of the given samples which is shown in Eq. (4.63).

$$\left\{\frac{2}{\|W\|}\right\}_{\min} \Leftrightarrow \left\{\frac{W}{\|2\|}\right\}_{\max} \tag{4.63}$$

$s.t.$

$$f(x) = \begin{cases} 1 & W \cdot x - b \geq 1 \\ -1 & W \cdot x - b < 1 \end{cases} \tag{4.64}$$

Furthermore, the SVM problem is written as:

$$\frac{1}{\|W\|}_{\min}, \; s.\,t.\, f(w^T x_i + b) \geq 1, i = 1, \ldots, n$$

where $\frac{1}{\|W\|}$ is the geometric margin, and $y = f(w^T x + b)$ is defined as the functional margin.

The SVM is known to generalize well even in high-dimensional spaces under small training sample conditions. The SVM is a binary classifier which determines the hyperplane that best separates two classes of information from the infinite number of hyperplanes that may exist. The optimal separating hyperplane is the one that gives the largest margin between the two classes. Therefore, we define a Lagrange function as,

$$L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (f(w^T x_i + b) - 1)$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ is the Lagrange multiplier. Hence,

$$\frac{\partial L(w, b, a)}{\partial w} = 0 \Rightarrow \ w = \sum_{i=1}^{n} \alpha_i x_i y_i$$

and,

$$\frac{\partial L(w, b, a)}{\partial b} = 0 \Rightarrow \ \sum_{i=1}^{n} \alpha_i x_i = 0$$

In the best case, these two classes are linearly separable; however, in most real-world scenarios, this is not true. To remedy this, the nonlinear SVM was created by adding a kernel function to the algorithm [7]. This allows the data to be mapped to a higher-dimensional space in a way that allows it to be linearly separable; meanwhile, the SVM is applicable to a wide range of image classification problems (Fig. 4.22).

$$w = \sum_{i=1}^{n} \alpha_i x_i y_i \tag{4.65}$$

$$f(x) = w^T x + b = \left(\sum_{i=1}^{n} \alpha_i x_i y_i \right)^T x + b \tag{4.66}$$

$$f(x) = \sum_{i=1}^{n} y_i < x_i, x > +b \tag{4.67}$$

In higher dimension, we have,

$$f(x) = \sum_{i=1}^{n} y_i \kappa(x_i, x) + b \tag{4.68}$$

where $\kappa(x_i, x)$ is the kernel function in SVM which maps data points of a low-dimensional space to the high-dimensional one for classification [21].

A kernal function is a nonnegative real-valued integrable function satisfying:

$$\int_{-\infty}^{+\infty} \kappa(u) du = 1 \tag{4.69}$$

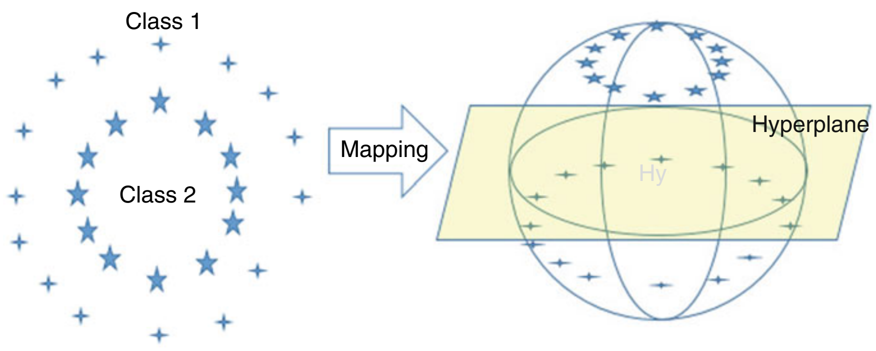


Fig. 4.22 SVM mapping for a lower-dimensional space to the higher-dimensional one

Fig. 4.23 Confusion matrix for binary classification

		Actual class	
		Yes	No
Predicted class	Yes	True Positives	False Positives
	No	False Negatives	True Negatives

and,

$$\kappa(u) = \kappa(-u), u \in (-\infty, +\infty) \tag{4.70}$$

Gaussian function, sigmoid function, cosine function, logistic function, etc., could be used as a kernel. For example, Gaussian kernel is,

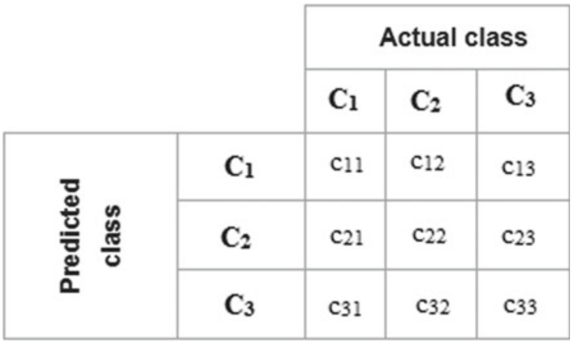
$$\kappa(x_1, x_2) = \exp\left(-\frac{\|x_i - x\|^2}{2\sigma^2}\right) \tag{4.71}$$

In a supervised learning, we typically need labeled training dataset, namely ground truth. Therefore, we split the dataset into positive and negative parts. For example, University of California Irvine, USA, (UCI) and National Institute of Standards and Technology (NIST) provided such labeled datasets in their Web sites for public downloading.

In a supervised learning, a typical classifier will classify a test dataset into two classes (Yes or No); these classes will construct a confusion matrix which consists of actual class and predict class in true positive (*tp*), false positive (*fp*), true negative (*tn*), and false negative (*fn*) as shown in Fig. 4.23. A more general confusion matrix is shown in Fig. 4.24, where c_{ij} is the number of samples predicted to be classified into class i, actually classified into class j after compared with the ground truth.

In a supervised learning, precision (PR) and recall(RC) are defined as below:

$$PR = \frac{tp}{tp + fp} \tag{4.72}$$



		Actual class		
		C_1	C_2	C_3
Predicted class	C_1	c_{11}	c_{12}	c_{13}
	C_2	c_{21}	c_{22}	c_{23}
	C_3	c_{31}	c_{32}	c_{33}

Fig. 4.24 Confusion matrix for classifying multiple classes

$$RC = \frac{tp}{tp + fn} \tag{4.73}$$

where tp, fp, fn, and tn are the true positive (a hit in classification), false positive (false alarm), false negative (missing classification), and true negative (correct rejection) respectively. The tp, fp, fn, and tn show among the search results how many search results reflect the ground truth exactly. Furthermore, F-measure or F-score (F), G-measure (G), accuracy (AC), sensitivity (SE) and specificity (SP), and miss rate (MR) are listed as:

$$F = \frac{2 \cdot PR \cdot RC}{PR + RC} \tag{4.74}$$

$$G = \sqrt{PR \cdot RC} \tag{4.75}$$

$$AC = \frac{tp + tn}{tp + tn + fp + fn} \tag{4.76}$$

$$SE = \frac{tp}{tp + fn} \tag{4.77}$$

$$SP = \frac{tn}{tn + fp} \tag{4.78}$$

$$MR = \frac{fn}{tp + fn} \tag{4.79}$$

For example, If we have $tp = 10, fp = 20$, $tn = 30$, and $fn = 40$, therefore,

$$PR = \frac{tp}{tp + fp} = \frac{10}{10 + 20} = \frac{1}{3} \tag{4.80}$$

$$RC = \frac{tp}{tp + fn} = \frac{10}{10 + 40} = \frac{1}{5} \tag{4.81}$$

$$AC = \frac{tp+tn}{tp+tn+fp+fn} = \frac{10+30}{10+30+20+40} = \frac{2}{5} \tag{4.82}$$

$$SE = \frac{tp}{tp+fn} = \frac{10}{10+40} = \frac{1}{5} \tag{4.83}$$

$$SP = \frac{tn}{tn+fp} = \frac{30}{30+20} = \frac{3}{5} \tag{4.84}$$

$$MR = \frac{fn}{tp+fn} = \frac{40}{10+40} = \frac{4}{5} \tag{4.85}$$

$$F = \frac{2 \cdot p \cdot r}{PR+RC} = \frac{1}{4} \tag{4.86}$$

$$G = \sqrt{PR \cdot RC} = \frac{\sqrt{15}}{15} \tag{4.87}$$

Note: F is harmonic mean (average) of recall and precision, and G is geometric mean (average).

A receiver operating characteristic curve (ROC) is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate ($0 \leq TPR = \frac{tp}{tp+fn} \leq 1$) against the false positive rate ($0 \leq FPR = \frac{fp}{fp+tn} \leq 1$) at various threshold settings. ROC is a comparison of two operating characteristics (*TPR* and *FPR*) as the criterion changes. The area under the curve is abbreviated as AUC. A ROC curve is shown in Fig. 4.25.

The often used software for data classification is Weka and R programming languages which include multiple statistical algorithms for classification, clustering, and regression (Fig. 4.26).

In Weka, the recommended file format is Attribute-Relation File Format (ARFF) which consists of American Standard Code for Information Interchange (ASCII) codes and describes a list of instances sharing a set of attributes.

Figure 4.27 shows an ARFF file opened by using a text editor, where "@RELATION iris" indicates the data is related to iris, "@ATTRIBUTE sepallength REAL" shows the attribute of the field "sepallength" is real, "@ATTRIBUTE class {Iris-setosa, Iris-versicolor, Iris-virginica}" tells us the "class" field is the labels of the iris including "Iris-setosa," "Iris-versicolor," "Iris-virginica," and the important information is applied to algorithm training and test. "@DATA" refers to the records that will start from the below. The record "5.1, 3.5, 1.4, 0.2, Iris-setosa" means the first record of this table is consisting of four real numbers and one label. The records will be listed till end of this file.

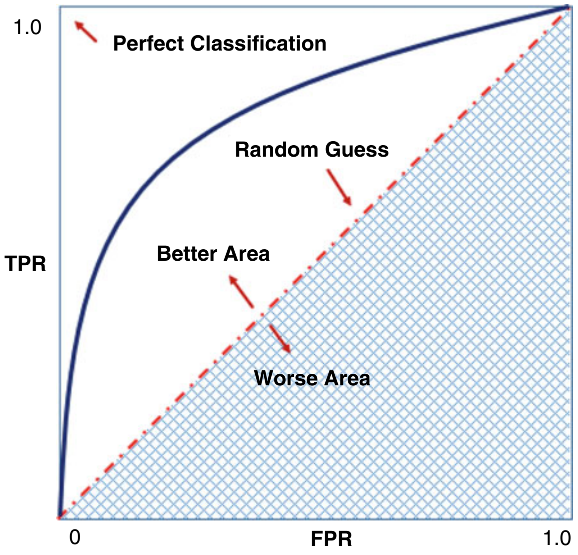
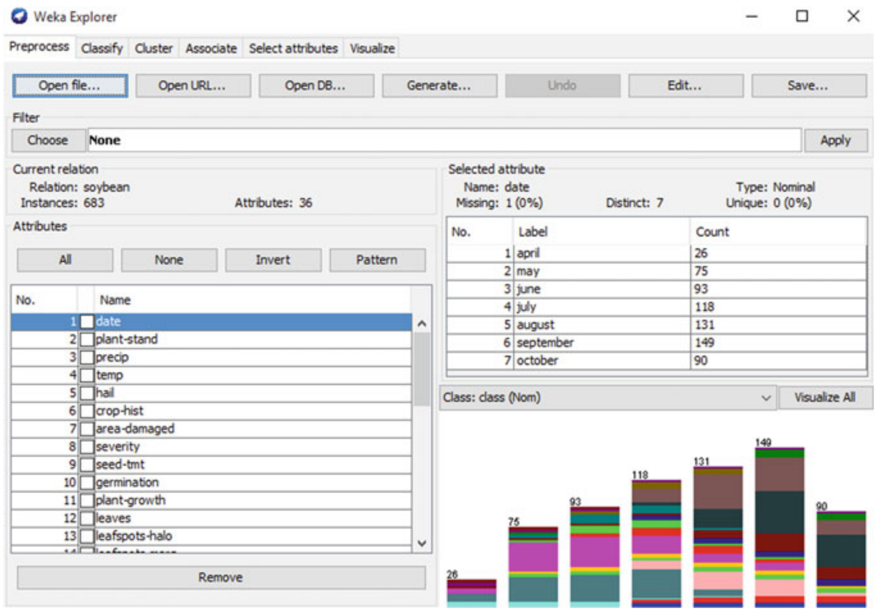


Fig. 4.25 A ROC curve in supervised learning



Filter: None

Current relation
Relation: soybean
Instances: 683 Attributes: 36

Attributes

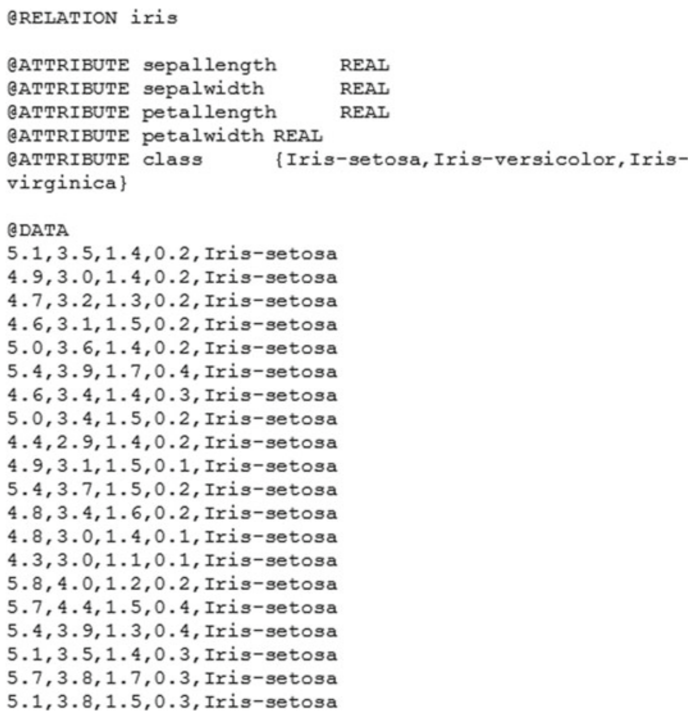
No.	Name
1	date
2	plant-stand
3	precip
4	temp
5	hail
6	crop-hist
7	area-damaged
8	severity
9	seed-tmt
10	germination
11	plant-growth
12	leaves
13	leafspots-halo

Selected attribute
Name: date Type: Nominal
Missing: 1 (0%) Distinct: 7 Unique: 0 (0%)

No.	Label	Count
1	april	26
2	may	75
3	june	93
4	july	118
5	august	131
6	september	149
7	october	90

Class: class (Nom)

Fig. 4.26 Interface of the software Weka



```
@RELATION iris

@ATTRIBUTE sepallength     REAL
@ATTRIBUTE sepalwidth      REAL
@ATTRIBUTE petallength     REAL
@ATTRIBUTE petalwidth REAL
@ATTRIBUTE class      {Iris-setosa,Iris-versicolor,Iris-
virginica}

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
```

Fig. 4.27 Format of the ARFF text file

4.4.2 Artificial Neural Networks (ANN)

Artificial neural network (ANN) is a biologically-inspired model which consists of neurons and connections using weights [43–46]. ANN constitutes of the neuronal structure, training and recall algorithms which are a connectionist model. Because of connections or links, the connection weights are the "memory" of the ANN system [46–48,85].

A standard model of ANNs is shown in Fig. 4.28. For the inputs $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and the associated weights $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, $w_i \geq 0$, we have a summation function $u = f(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{n} (w_i \cdot x_i)$ and the activation function $\alpha = s(u)$ to calculate the activation level of a neuron; the output function g is used to compute the output signal value emitted through the output(axon) of the neuron $o = g(\alpha)$. The output values of a neuron are within [0, 1].

ANNs are thought as nonlinear classification [20]. For example, the discriminant function of a three-layer neural network is,

$$g_k(x) \equiv z_k = f\left(\sum_{j=1}^{nh} w_{kj} \cdot f\left(\sum_{i=0}^{d} w_{ji}x_i + w_{j0}\right) + w_{k0}\right) \tag{4.88}$$

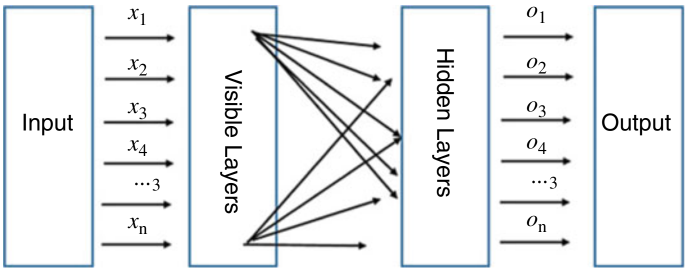


Fig. 4.28 A standard model of ANNs

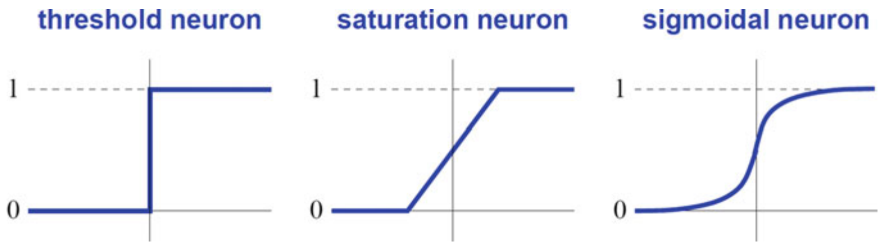


Fig. 4.29 Activation functions in artificial neural network

Kolmogorov Theorem. Any continuous function $g(x)$, $\mathbf{X} = (x_1, x_2, \ldots, x_d) \in [0, 1]^d$, $d \geq 2$ could be represented by,

$$g_k(x) = \sum_{j=1}^{2n+1} \xi_j \left(\sum_{i=0}^{d} \psi_{ij}(x_i) \right) \tag{4.89}$$

From Kolmogorov theorem, we know that ANNs have the ability for nonlinear classification. In ANNs, the activation function s in history had adopted the following functions shown in Fig. 4.29, respectively.

- Hard-limited threshold function,

$$s(\delta) = \begin{cases} 1 \; \delta \geq \varepsilon \\ 0 \; \delta < \varepsilon \end{cases} \tag{4.90}$$

  where $\varepsilon > 0$ is the threshold.
- The linear saturated function,

$$\pi(y) = \begin{cases} \pi(y) = y & y \in [0, 1] \\ 0 & y < 0 \\ 1 & y > 1 \end{cases} \tag{4.91}$$

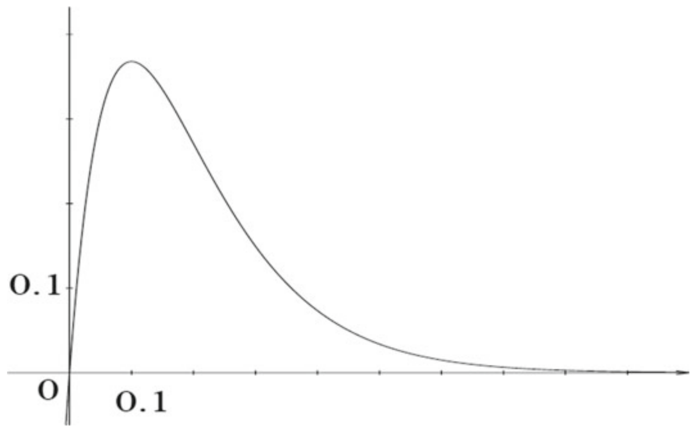


Fig. 4.30 Alpha function used in spiking neural network

where $y = 0$ is the saturated threshold.

- Sigmoid function,

$$s(u) = \frac{1}{1 + e^{-u}} \tag{4.92}$$

The sigmoid function has s-shape, and it is monotonically increasing, continuous, and smooth.

- α-function,

$$f(x) = \begin{cases} x \cdot \frac{1}{e^{\alpha \cdot x}} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{4.93}$$

This function is usually used for the third generation (3G) of neural network: spiking neural network (SNN) [29,43,86] shown in Fig. 4.30.

In summary, the model of ANN is an assembly of interconnected nodes and weighted links, the output is to sum up each of its input value according to the weights of its links.

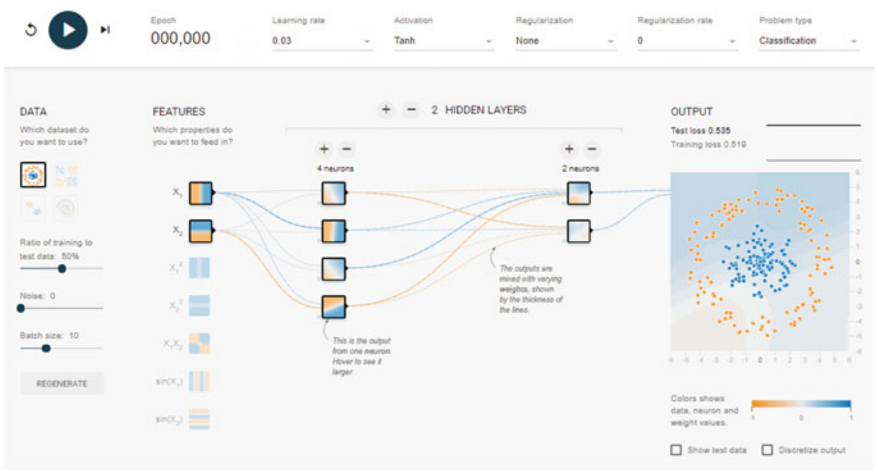
Step 1. Initialize the weights $(w_1, w_2, \ldots, w_n)$.

Step 2. Adjust the weights in such a way that the output of ANN is consistent with class labels of training examples.

- Objective function:

$$\varepsilon = \frac{1}{2} \sum_i (o_i - g(w_i, x_i))^2 \tag{4.94}$$

where o_i is an output, x_i is an input, w_i is the corresponding weight, and $g(\cdot)$ is the training function. In ANNs, the objective function usually is an average related to the input and output. Equation (4.94) is quadratic cost or sum squared error.


Epoch 000,000
Learning rate 0.03
Activation Tanh
Regularization None
Regularization rate 0
Problem type Classification

DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10

FEATURES
Which properties do you want to feed in?

2 HIDDEN LAYERS
4 neurons
2 neurons

This is the output from one neuron. Hover to see it larger.

The outputs are mixed with varying weights, shown by the thickness of the lines.

OUTPUT
Test loss 0.535
Training loss 0.519

Colors shows data, neuron and weight values.

Show test data
Discretize output

Fig. 4.31 Playground of deep learning

Other objective functions include cross-entropy cost, exponential cost, Kullback–Leibler (KL) divergence, etc. [12,28].

- Weights:
  The goal of ANN computation is to find the weights w_i, $i = 1, 2, \dots, n$ that minimize the above objective function or cost function in Eq. (4.94), $\frac{\partial \varepsilon(\mathbf{W})}{\partial \mathbf{W}} = 0$.
- Algorithms:
  The algorithms to train ANN weights and achieve the minimum objective function typically include gradient descent, Newton's method, conjugate gradient, quasi-Newton method, and Levenberg–Marquardt algorithm in mathematics. For example, the Newton's method for seeking the weights is,

  $$w_{i+1} = w_i - m \cdot \frac{g(x_i)}{g'(x_i)} \tag{4.95}$$

  where m is a factor, we call it as learning rate.
  Therefore, gradient descent (i.e., method of steepest descent) is written as,

  $$w_{i+1} = w_i - m \cdot \nabla g(w_i) \tag{4.96}$$

  where $-\nabla g(w_i)$ is the direction of normal vector.

In order to understand these concepts well, a software called Tinker based on TensorFlow was developed, and its interface is shown in Fig. 4.31.

Compared to the peer algorithms, ANN has the weights as its memory and satisfies the requirements of event detection and recognition on spatiotemporal relationship related to time serial analysis [70]. ANNs have the merits to replace FSM, HMM, and SVM in machine learning and pattern classification [43]. Recent years, ANNs exhibit its mighty strength in deep learning [57] and big data analysis [37,86].

NeuCube is a framework for the development of spiking neural network (SNN) systems in data mining, pattern recognition, and predictive data modeling with complex and large data, especially for the spatiospectro-temporal data (SSTD) [29,49]. KEDRI from the Auckland University of Technology (AUT), New Zealand, has developed the Neurogenetic Cube system recently which has the architecture of a STDM [48]. It is a sophisticated framework of methods that facilitates efficient solutions to the problems through meticulous and accurate selection test of most suitable methods and parameters for a STDM. The deSNN has the most powerful capability to cope with the spatiotemproal data in the term of accuracy [44]. NeuCube is based on brain-like neural networks and aims at solving the problems from the viewpoint of pattern recognition and classification [38]. NeuCube could be applied to vehicle monitoring, detection, and object recognition [49].

4.4.3 Deep Learning

Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. It is said that perceptron (IBM704) in 1957 is the beginning of the era of machine learning [66,82]. In the following years, the algorithms such as delta rule, also called the least mean square (LMS) method [84], XOR logic function [66], automatic differentiation(AD) [31], multilayer perception (MLP), decision tree [83], support vector machine [10], AdaBoost [23], and random forest [8] are thought as the milestones of the development of machine learning.

Deep learning was thought starting from restricted Boltzmann machine (RBM) in 1986 [20] and has been further developed in 1995 by using convolution neural network (CNN) for handwriting recognition, which has been implemented with several rounds of convolution and subsampling; then, full connections and Gaussian connections have been applied to the neural network layers [28,57]. In 2006, deep belief network(DBN) [32] was successfully developed which has pushed the deep learning research greatly forward.

The famous application of deep learning was AlphaGo [89] from Google DeepMind in 2016. The computer program to play board game Go (Weiqi) has defeated human professional Go players on the 19×19 board. AlphaGo's algorithm is based on reinforcement learning which has the steps: (1) policy network, (2) fast rollout, (3) value network, (4) Monte Carlo tree search (MCTS).

Traditional machine learning is "shallow" not so "deep" which is based on training dataset (labeled) and test dataset, feature extraction from feature engineering (e.g., SIFT, HoG, etc.), classifier selection (e.g., SVM, AdBoost, etc.), and evaluations of classification results (e.g., precision, recall, ROC, AUC, etc.). "Shallow" (linear) classifier operating on raw pixels could not possibly distinguish difference and output different objects in the same category. "Shallow" classifiers require a good feature extractor that solves the selectivity; namely, those are selective to the aspects of the image that are important for discrimination, but invariant to irrelevant aspects [42,52,57,93].

Deep learning has powerful ability of nonlinear processing using a cascade of multiple layers for feature transformation and *end-to-end* learning. A deep learning architecture is a multilayer stack of simple modules, and it computes nonlinear input-output mappings. Deep learning has to experience the steps such as feature mapping, pooling, etc. Currently, the typical deep learning methods include [1,57], RNN (LSTM [33], GRU) [34], R-CNN [26], fast R-CNN [27], and faster R-CNN [80], YOLO, YOLO9000 [78], SSD [62], etc.

• *ConvNets*(*CNNs*)
ConvNets(CNNs) include the steps: local connections, shared weights, pooling, and the use of many layers; meanwhile, RNNs can be seen as very deep feedforward networks in which all the layers share the same weights [34,56,57].

In CNNs, $\forall x_1, x_2 \ldots, x_n \in \mathbf{R}^+$,

$$y = W^\tau h + b; \tag{4.97}$$

$$h^{(i)} = g(w^{(i)}x + b^{(i)}) \tag{4.98}$$

where $g(\cdot)$ could be:

- ReLU function: $g(z) = \max(0, z)$
- Logistic function: $g(z) = \frac{1}{1+e^{-z}}$
- Tanh function: $g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- Sigmoid function: $g(y) = \frac{1}{1+e^y}$
- Softmax function: $g(y_i) = softmax(y_i) = \frac{\exp(y_i)}{\sum_j \exp(y_j)}$

Given $(h_{i,j}^{(k)})_{m\times m}$ at level k, $g(\cdot)$ is a nonlinear function, and a convolutional operation (3×3 or 7×7) is,

$$h_{i,j}^{(k+1)} = g(a^{(k)} \cdot h_{i,j}^{(k)} + b^{(k)} \cdot h_{i+1,j}^{(k)} + c^{(k)} \cdot h_{i,j+1}^{(k)} + d^{(k)} \cdot h_{i+1,j+1}^{(k)}) \tag{4.99}$$

For average pooling with downsampling,

$$\bar{h}^{(k+1)} = \frac{1}{4}(a^{(k)} \cdot h_{i,j}^{(k)} + b^{(k)} \cdot h_{i+1,j}^{(k)} + c^{(k)} \cdot h_{i,j+1}^{(k)} + d^{(k)} \cdot h_{i+1,j+1}^{(k)}) \tag{4.100}$$

For a maxpooling with downsampling,

$$h_{\max}^{(k+1)} = \max(a^{(k)} \cdot h_{i,j}^{(k)}, b^{(k)} \cdot h_{i+1,j}^{(k)}, c^{(k)} \cdot h_{i,j+1}^{(k)}, d^{(k)} \cdot h_{i+1,j+1}^{(k)}) \tag{4.101}$$

A loss function is,

$$J(\theta) = -\mathscr{E} \log P_\theta(y|x) \tag{4.102}$$

or,

$$J(\theta) = -\mathscr{E}[\|y - f(x,\theta)\|^2] \tag{4.103}$$

where $\mathscr{E}(\cdot)$ is the expectation,

$$\theta = \underset{\theta}{\arg\min} J(\theta) \tag{4.104}$$

Hence,

$$\frac{dJ(\theta)}{d\theta} = 0 \tag{4.105}$$

Other loss functions include:

- 0∼1 loss function: $L(Y, f(X)) = \begin{cases} 1 & Y \neq f(X) \\ 0 & Y = f(X) \end{cases}$
- Square loss function: $L(Y, f(X)) = (Y - f(X))^2$
- Absolute loss function: $L(Y, f(X)) = |Y - f(X)|$
- Logarithm loss function: $L(Y, p(Y|X)) = -\log p(Y|X)$
- Average loss function: $\bar{L} = \frac{1}{m}\sum_{i=1}^{m} L(x_i, y_i)$, where the set $T = \{(x_i, y_i)\}(i = 1, 2, \ldots, m)$ is the training dataset.

CNN has been applied to image noise removal [63]. Compared with traditional image denoising methods such as average filtering, Wiener filtering, and median filtering, the advantage of using this CNN model is that the parameters of this model can be optimized through network training, whereas in traditional image denoising, the parameters of these algorithms are fixed and cannot be adjusted during the filtering, namely lack of adaptivity. Meanwhile, the parallel processing ability of neural networks makes it possible for image denoising and speedup image denoising process.

• *Recurrent Networks*

RNNs are a family of neural networks for processing sequential data. Most recurrent networks can process sequences with a variable length. From a dynamical system driven by external $x^{(t)}$, we have

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta) = g^{(t)}(x^{(t)}, x^{(t-1)}, \ldots, x^{(1)}) \tag{4.106}$$

where $t = 1, 2 \ldots, \tau$, h is the state. Recurrent neural networks produce an output at each time step and have recurrent connections between hidden units. For $i = 1, 2, \ldots, \tau$,

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \tag{4.107}$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)}) \tag{4.108}$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V} \cdot \mathbf{h}^{(t)} \tag{4.109}$$

$$\hat{\mathbf{y}}^{(t)} = \text{softmax}(\mathbf{o}^{(t)}) \tag{4.110}$$

where **b** and **c** are vectors, and **U**, **V**, **W** are weight matrices. The loss function L is,

$$L = \sum_{t=1}^{\tau} L^{(t)} = \sum_{t=1}^{\tau} \log p(\mathbf{y}^{(t)} | \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t)}\}) \tag{4.111}$$

Long short-term memory (LSTM) is a RNN model for the short-term memory which can last for a long period of time [24,25]. An LSTM unit consists of four gates: input gate, cell, forget gate, and output gate. LSTM is well suited to classify, process, and predict time series given time lags of unknown size and duration between important events. LSTM (memory) cell stores a value (or state), for either long or short time periods. LSTM gates compute an activation, often using the logistic function. LSTM was developed to deal with the exploding and vanishing gradient problem [33].

$$f_t = \sigma_g(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f)$$

$$i_t = \sigma_g(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i)$$

$$o_t = \sigma_g(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o)$$

$$c_t = f_t \cdot c_{t-1} + i_t \circ \sigma_c(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c)$$

$$h_t = o_t \circ \sigma_h(c_t)$$

where x_t and h_t are input and output vectors, $c_0 = 0$, $h_0 = 0$; f_t, i_t, and o_t are activation functions of forget, input, and output gates; W, U, b are weight matrices and bias vectors; c_t is the cell state vector; '$\circ$' is Hadamard product, namely $A_{m\times n} \circ B_{m\times n}$=$(a_{ij})_{m\times n} \circ (b_{ij})_{m\times n} = (a_{ij} \cdot b_{ij})_{m\times n}$. $\sigma_g(\cdot)$, $\sigma_c(\cdot)$ and $\sigma_h(\cdot)$ are activation functions.

Gated recurrent unit (GRU) model based on the LSTM is a big change as forget gate is integrated with the input gate and turns them into a single update gate [33]. GRU is a relatively successful variant of LSTM, its number of parameters is smaller than average LSTM, and the model will converge earlier in the training. Also, the GRU does not require an initialization operation to achieve good results. For a fully gated unit, initially, $t = 0$, $h_0 = 0$,

$$z_t = \sigma_g(W_z \cdot x_t + U_z \cdot h_{t-1} + b_z) \quad (\textit{update gate})$$

$$r_t = \sigma_g(W_r \cdot x_t + U_r \cdot h_{t-1} + b_r) \quad (\textit{reset gate})$$

$$\tilde{h}_t = \sigma_h(W_h \cdot x_t + U_h(r_t \circ h_{t-1}) + b_h) \quad (\textit{new memory})$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t \quad (\textit{hidden state})$$

where x_t and h_t are input and output vectors; W, U, and b are weight matrices and vector; '$\circ$' is the Hadamard product; $\sigma_g(\cdot)$ and $\sigma_h(\cdot)$ are sigmoid and tanh activation functions.

• *Recursive Networks*

For recursive neural networks,

$$\mathbf{h}^{(t)} = \mathbf{W}^{\tau}\mathbf{h}^{(t-1)} \tag{4.112}$$

As the power methods in recursion,

$$\mathbf{h}^{(t)} = (\mathbf{W}^{t})^{\tau}\mathbf{h}^{(0)} \tag{4.113}$$

If eigenvalues of $\mathbf{W}$ exist, then

$$\mathbf{W} = \mathbf{Q}\Lambda\mathbf{Q}^{\tau} \tag{4.114}$$

If $\mathbf{Q}$ is orthogonal, then

$$\mathbf{h}^{(t)} = \mathbf{Q}^{\tau}\Lambda^{t}\mathbf{Q}\mathbf{h}^{(0)} \tag{4.115}$$

where $\Lambda = diag(\lambda_1, \ldots, \lambda_m)$, $\lambda_i \neq 0$, when $|\lambda_i| < 1$, RNN is contractive.

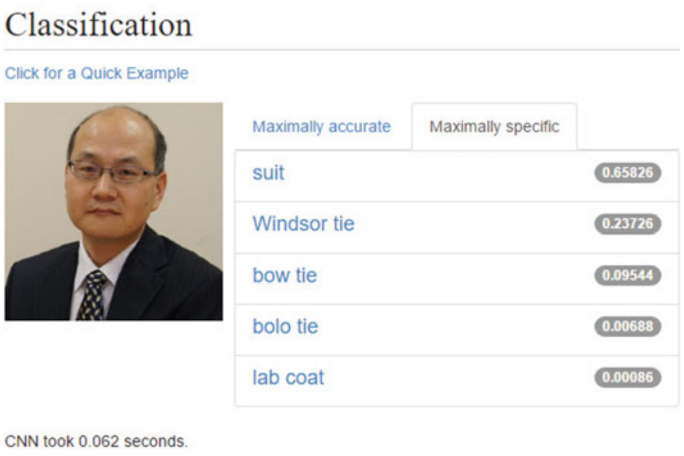
• *R-CNN*

R-CNN [26] refers to region-based convolutional neural networks which generates potential bounding boxes in an image and runs a classifier on these proposed boxes. Post-processing is used to refine the bounding boxes so as to eliminate duplicate detections and rescore the boxes based on other objects in the scene.

The biggest problem of R-CNN [26] is that the training time and test time are very long because it needs to get $1000 \sim 2000$ proposals first and save them to disk, and also these proposals need to be calculated in all the former layers which need lots of repetitions. In addition, the fully connected layer is expected that all the vectors will have the same size, so all the proposals need to be resized using crop or wrap; both strategies are not suitable because the crop may cause that the proposals are not fully extracted and the wrap could change scales of objects.

Fast R-CNN [27] overcomes several problems of R-CNN. R-CNN decreases the consumptions of time and space. What fast R-CNN has done is to replace ROI pooling layer using the pooling layer 5, and softmax function is applied to classification. The softmax is one extension of logistic regression to the multiclass classification problem.

Faster R-CNN [80] was proposed to improve the training speed of the fast R-CNN. From R-CNN to faster R-CNN, the four steps of object detection are finally unified into one network. Faster R-CNN does not use selective search to get region proposals. Instead, it takes use of a region proposal network to carry out the same task. There has not repetition, all the calculations are performed by using GPUs.


Classification

suit	0.65826
Windsor tie	0.23726
bow tie	0.09544
bolo tie	0.00688
lab coat	0.00086

CNN took 0.062 seconds.

Fig. 4.32 Screenshot of the deep learning software: Caffe

• *SSD*
Single Shot MultiBox Detector (SSD) [62] is very similar to the faster R-CNN and simultaneously produces a score for each object category in each box. It skips the proposal step and predicts bounding boxes and confidences for multiple categories directly. SSD uses default boxes of different aspect ratios on each feature location from multiple feature maps at different scales.

• *YOLO*
YOLO [77] is one of the fast object detectors with regression, 45 frames per second, its mAP is up to 57.9%. YOLO creates an $S \times S$ grid cells; each cell will be responsible for the object which falls into the cell. Every cell will predict the bounding box and the confidence score of this box. For evaluating YOLO on VOC (Pascal VOC datasets), a 7×7 bounding box and 20 labeled classes are defined, which means it only extracts 98 proposals. YOLO is faster than R-CNN [26] which needs 2000 proposals. YOLO9000 [78] is a real-time framework for detection more than 9000 object categories by jointly optimizing detection and classification.

YOLO has been applied to flame detection in surveillance [87]. YOLO used the whole image instead of a regional proposal to train and test. When compared object detection to a real-time model, YOLO has an overwhelming advantage. When fire flames have entirely different color features compared to the training set, shallow learning may have difficulties to detect them. However, deep learning demonstrates its superior performance in this case; it is not influenced by the changes of flames, thanks to its merits of the fine-grained adaptivity.

An example of object classification of deep learning from Caffe [40] Demos is shown in Fig. 4.32. In this example, a photograph uploaded to the Web site has been analyzed and explained with semantic concepts.

• *Problems of Backpropagation*

SGD refers to stochastic gradient descent in deep learning. The loss function of SGD is $J(\theta) = L(f_\theta(x_i), y_i)$; (x_i, y_i) are samples ($i = 1, \ldots, m$); w.r.t., θ in $\frac{\partial J(\theta)}{\partial \theta} = 0$,

$$\theta_j := \theta_j - \alpha \cdot \frac{\partial}{\partial \theta_j} J(\theta_j); \forall j = 0, 1, 2, \ldots, n; \forall i = 1, 2, \ldots, m.$$

where α is learning rate.

For the global optimization, there are saddle points existing in the backpropagation procedure [28]. To find these points, gradient vanishing and gradient exploding problems have to be overcome. The usual ways to solve these problems are through hierarchy of networks, restricted Boltzmann machine (RBM), generative models, long short-term memory (LSTM), and residual networks (ReNets) or redesign the network using gradient clipping, weight regularization, etc.

Regularization is defined as any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error. The regularized objective function is,

$$\hat{J}(\theta; \mathbf{X}, \mathbf{y}) = J(\theta; \mathbf{X}, \mathbf{y}) + \alpha \cdot \Omega(\theta) \tag{4.116}$$

where $\alpha \in [0, \infty]$ is a hyperparameter or regularization rate; θ denotes all of the parameters. The optimized parameters θ^* are obtained by using

$$\theta^* = \arg\min_{\theta} \nabla_\theta \hat{J}(\theta; \mathbf{X}, \mathbf{y}) \tag{4.117}$$

Regularization is helpful to reduce overfitting. L_2 regularization and dropout are two very effective regularization techniques.

$$\Omega(\theta) = \frac{1}{2}\|\mathbf{w}\|_2^2 \tag{4.118}$$

Thus,

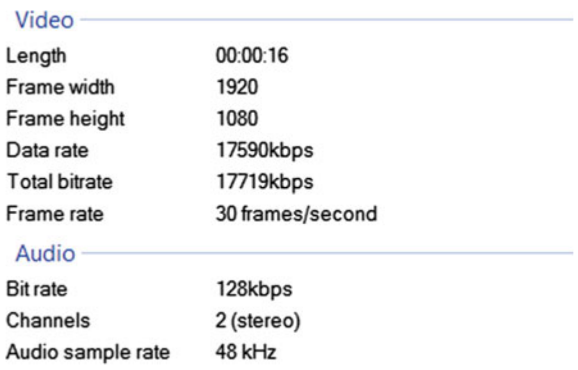
$$\hat{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = J(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \frac{\alpha}{2}\mathbf{w}^\tau \mathbf{w} \tag{4.119}$$

The gradient,

$$\nabla_{\mathbf{w}} \hat{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \nabla_{\mathbf{w}} J(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha \cdot \mathbf{w} \tag{4.120}$$

To update the weights,

$$\mathbf{w} \leftarrow \mathbf{w} - \varepsilon \cdot \nabla_{\mathbf{w}} \hat{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}), \varepsilon \in (0, 1) \tag{4.121}$$



Video	
Length	00:00:16
Frame width	1920
Frame height	1080
Data rate	17590kbps
Total bitrate	17719kbps
Frame rate	30 frames/second
Audio	
Bit rate	128kbps
Channels	2 (stereo)
Audio sample rate	48 kHz

Fig. 4.33 Metadata of a video with audio track

4.5 Metadata of Surveillance Data

Metadata is not the data of the object but relates to that object, such as file size, time stamps, attributes, and GPS information. Metadata is not the file content; it is called data of data. Meta attributes include time stamp, file index, tags, file size, etc.

For a photography, we could find the EXIF items, and the items include camera ID, camera maker (who made the camera), camera model, exposure time for this photograph, focal length, ISO speed, flash mode, saturation, sharpness, light sources for the camera to take the photograph at that moment. An example of metadata of a video with audio track is shown in Fig. 4.33.

4.6 Questions

Question 1. What are the key issues in object recognition?
Question 2. What components are included in a visual feature vector?
Question 3. Please explain the below concepts in object recognition.
(1) Training set and test set
(2) Ground truth
(3) Detector
(4) Classifier
(5) Accuracy, precision, and recall
(6) Confusion matrix
(7) ROC curve
(8) F-measure and G-measure
Question 4. Please explain k-nearest neighbors algorithm (k-NN).
Question 5. Please explain the simplest form of Bayes' rule (theorem).
Question 6. What are the differences between deep learning and shallow learning?
Question 7. Why Gabor transform is important in computer vision?

References

1. Alom MZ, Alam M, Taha TM, Iftekharuddin KM (2017) Object recognition using cellular simultaneous recurrent networks and convolutional neural network. In: International joint conferences on neural networks (IJCNN), pp 2873–2879
2. Anagnostopoulos IE, Psoroulas ID, Loumos V, Kayafas E (2008) License plate recognition from still images and video sequences: a survey. IEEE Trans Intell Transp Syst 9(3):377–391
3. Bai H, Liu C (2004) A hybrid license plate extraction method based on edge statistics and morphology. In: IEEE ICPR, pp 831–834
4. Bailey DG, Irecki D, Lim B, Yang L (2002) Test bed for number plate recognition applications. In: IEEE international workshop on electronic design, test and applications. https://doi.org/10.1109/DELTA.2002.994684
5. Beymer D, McLauchlan P, Coifman B, Malik J (1997) A real-time computer vision system for measuring traffic parameters. In: IEEE CVPR, pp 495–502
6. Bimbo A (1999) Visual information retrieval. Morgan Kaufmann Publishers, San Francisco
7. Boser E, Guyon M, Vapnik N (1992) A training algorithm for optimal margin classifiers. In: The fifth annual workshop on computational learning theory, pp 144–152
8. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
9. Bui H, Venkatesh S, West G (2001) Tracking and surveillance in wide-area spatial environments using the abstract Hidden Markov Model. Pattern Recognit 15(1):177–195
10. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
11. Comelli P, Ferragina P, Granieri MN, Stabile F (1995) Optical recognition of motor vehicle license plates. IEEE Trans Veh Technol 44(4):790–799
12. Cover T, Thomas J (2006) Elements of information theory, 2nd edn. Wiley, New Jersey
13. Cui W, Yan W (2016) A scheme for face recognition in complex environments. Int J Digit Crime Forensics 8(1):11
14. Cui Y, Huang Q (1997) Automatic license extraction from moving vehicles. In: International conference on image processing, pp 126–129
15. Deb K, Chae HU, Jo KH (2008) Parallelogram and histogram based vehicle license plate detection. In: International conference on smart manufacturing application, pp 349–353
16. Dhoble K, Nuntalid N, Indiveri G, Kasabov N (2012) Online spatio-temporal pattern recognition with evolving spiking neural networks utilizing address event representation, rank order, and temporal spike learning. In: International joint conference on neural networks IEEE IJCNN, pp 554–560
17. Drazen D, Lichtsteiner P, Hfliger P, Delbrck T, Jensen A (2011) Toward real-time particle tracking using an event-based dynamic vision sensor. Exp Fluids 51:1465–1469
18. Du S, Ibrahim M, Shehata M, Badawy W (2013) Automatic license plate recognition (ALPR): a state-of-the-art review. IEEE Trans Circuits Syst Video Technol 23(2):311–25
19. Duan TD, Du TH, Phuoc TV, Hoang NV (2005) Building an automatic vehicle license plate recognition system. In: International conference on computer science, pp 59–63
20. Duda R, Hart P, Stork D (2001) Pattern classification. Wiley, New York
21. Elisseeff A, Weston J (2001) A kernel method for multi-labeled classification. In: International conference on neural information processing systems: natural and synthetic, pp 681–687
22. Ferryman JM, Maybank SJ, Worrall AD (2000) Visual surveillance for moving vehicles. IJCV 37(2):187–197
23. Freund Y, Schapire R, Abe N (1999) A short introduction to boosting. J Jpn Soc Artif Intell 14:1612
24. Gers F, Schmidhuber J, Cummins F (2000) Learning to forget: continual prediction with LSTM. Neural Comput 12(10):2451–2471
25. Gers F, Schraudolph N, Schmidhuber J (2003) Learning precise timing with LSTM recurrent networks. J Mach Learn Res 3:115–143

26. Girshick R et al (2015) Region-based convolutional networks for accurate object detection and segmentation. IEEE PAMI 28(1):142–158
27. Girshick R (2015) Fast R-CNN. In: ICCV'15, pp 1440–1448
28. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Boston
29. Gupta A, Long N (2007) Character recognition using spiking neural networks. In: International joint conference on neural networks, pp 12–17
30. Hampapur A, Brown L, Connell J, Ekin A, Haas N, Lu M, Merkl H, Pankanti S (2005) Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking. IEEE Signal Process 22(2):38–51
31. Hecht-Nielsen R (1989) Theory of the backpropagation neural network. In: IJCNN, pp 593–605
32. Hinton E, Simon O, Teh Y (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
33. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
34. Hochreiter S, Bengio Y, Frasconi P, Schmidhuber J (2001) Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. A field guide to dynamical recurrent network. Wiley-IEEE Press
35. Huang YP, Lai SY, Chuang WP (2004) A template-based model for license plate recognition. In: IEEE international conference on networking, sensing and control, pp 737–742
36. Huang S, Hong J (2011) Moving object tracking system based on Camshift and Kalman filter. In: International conference on consumer electronics, communications and networks, pp 1423–1426
37. Huang T (2014) Surveillance video: the biggest big data. Comput Now 7(2)
38. Huzlu H, Kasabov N, Shamsuddin S, Widiputra H, Dhoble K (2011) An extended evolving spiking neural network model for spatiotemporal pattern classification. In: International joint conference on neural networks, pp 2653–2656
39. Jain R, Kasturi R, Schunck B (1995) Machine vision. McGraw-Hill, New Jersey
40. Jia Y et al, Caffe: convolutional architecture for fast feature embedding. In: ACM MM'14
41. Jiao Y, Weir J, Yan W (2011) Flame detection in surveillance. J Multimed 6(1):22–32
42. Jin C, Chen T, Ji L (2013) License plate recognition based on edge detection algorithm. In: International conference on intelligent information hiding and multimedia signal processing, pp 395–398
43. Kasabov N (1996) Foundations of neural networks, fuzzy systems, and knowledge engineering. The MIT Press, Boston
44. Kasabov N (2010) Knowledge extraction from evolving spiking neural networks with rank order population coding. Int J Neural Syst 20(6):437–445
45. Kasabov N (2010) To spike or not to spike: a probabilistic spiking neuron model. Neural Netw 23(1):16–19
46. Kasabov N, Dhoble K, Nuntalid N, Mohemmed A (2011) Evolving probabilistic spiking neural networks for spatiotemporal pattern recognition: a preliminary study on moving object recognition. In: ICONIP, pp 230–239
47. Kasabov N (2012) Evolving, probabilistic spiking neural networks and neurogenetic systems for spatio-and spectro-temporal data modelling and pattern recognition. In: WCCI, pp 234–260
48. Kasabov N (2014) NeuCube: a spiking neural network architecture for mapping, learning and understanding of spatiotemporal brain data. Neural Netw 52:62–76
49. Kasabov N (2017) From multilayer perceptrons and neuro-fuzzy systems to deep learning machines: which method to use? Sur Int J Inf Technol Sec 9(20):3–24
50. Katznelson Y (2004) An introduction to harmonic analysis, 3rd edn. Cambridge University Press, Cambridge

51. Kertesz A, Kertesz V, Muller T (1994) An on-line image processing system for registration number identification. In: IEEE world congress on computational intelligence, pp 4145–4148
52. Kim S, Kim D, Ryu Y, Kim G (2002) A robust license-plate extraction method under complex image conditions. In: International conference on pattern recognition, pp 216–219
53. Kinjal A, Darshak G (2012) A survey on moving object detection and tracking in video surveillance system. Int J Soft Comput Eng 2(3):44–48
54. Klette R (2014) Concise computer vision. Springer, London
55. Kwafinicka H, Wawrzyniak B (2002) License plate localization and recognition in camera pictures. In: The symposium on methods of artificial intelligence, pp 243–246
56. LeCun Y, Bengio Y (1995) The handbook of brain theory and neural networks. Convolutional networks for images, speech, and time series. MIT Press, Cambridge, pp 255–258
57. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444
58. Lee ER, Kim PK, Kim HJ (1994) Automatic recognition of a car license plate using color image processing. In: IEEE international conference on image processing, pp 301–305
59. Li P, Nguyen M, Yan W, Rotation correction for license plate recognition. In: ICCAR'18, pp 400–404
60. Li S (2009) Markov random field modeling in image analysis, 3rd edn. Springer, London
61. Lin C-C, Huang W-H (2007) Locating license plate based on edge features of intensity and saturation subimages. In: International conference on innovative computing, information and control, pp 227–227
62. Liu W et al (2016) SSD: single shot multibox detector. In: ECCV'16, pp 21–27
63. Liu Z, Yan W, Yang B, Image denoising based on a CNN model. In: ICCAR'18
64. Mahini H, Kasaei S, Dorri F (2006) An efficient features-based license plate localization method. In: International conference on pattern recognition, pp 841–844
65. Megalingam RK, Krishna P, Somarajan P, Pillai VA, Hakkim RU (2010) Extraction of license plate region in automatic license plate recognition. In: International conference on mechanical and electrical technology, pp 496–501
66. Minsky M, Papert S (1987) Perceptrons: an introduction to computational geometry. MIT Press, Cambridge
67. Murray D, Basu A (1994) Motion tracking with an active camera. IEEE Trans Pattern Anal Mach Intell 16(5):449–459
68. Naito T, Tsukada T, Yamada K, Kozuka K, Yamamoto S (2000) Robust license-plate recognition method for passing vehicles under outside environment. IEEE Trans Veh Technol 49(6):2309–2319
69. Nijhuis JAG, Ter Brugge MH, Helmholt KA, Pluim JPW, Spaanenburg L, Venema RS, Westenberg MA (1995) Car license plate recognition with neural networks and fuzzy logic. IEEE Int Conf Neural Netw 5:2232–2236
70. Oh S, Hoogs A, Perera A, Cuntoor N, Chen C, Lee T et al (2011) A large-scale benchmark dataset for event recognition in surveillance video. In: IEEE conference on computer vision and pattern recognition, pp 3153–3160
71. Ojala T, Pietikainen M, Maenpaa T (2002) Multiresolution gray scale and rotation invariant texture classification with local binary patterns. IEEE Trans Pattern Anal Mach Intell 24(7):971–987
72. Oren M, Papageorgiou C, Sinham P, Osuna E, Poggio T (1997) Pedestrian detection using wavelet templates. In: IEEE CVPR, pp 193–199
73. Parisi R, Di Claudio E, Lucarelli G, Orlandi G (1998) Car plate recognition by neural networks and image processing. In: IEEE international symposium on circuits and systems, pp 195–198
74. Patel C, Shah D, Patel A (2013) Automatic number plate recognition system (ANPR): a survey. Int J Comput Appl 69(9):21–33
75. Pless R, Brodsky T, Aloimonos Y (2000) Detecting independent motion: the statics of temporal continuity. IEEE PAMI 22(8):768–773

76. Rao R, Yip C, Britanak V (2007) Discrete cosine transform: algorithms, advantages, applications. Academic Press, San Diego
77. Redmon J et al (2016) You only look once: unified, Real-time object detection. In: CVPR'16
78. Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. In: CVPR'2017
79. Ren Y (2017) Banknote recognition in real time using ANN (Master Thesis). Auckland University of Technology, New Zealand, Auckland
80. Ren S, He K, Girshick R, Sun J (2017) Faster R-CNN: towards real-time object detection with region proposal networks. IEEE PAMI 39(6):1137–1149
81. Roomi S, Anitha M, Bhargavi R (2011) Accurate license plate localization. In: International conference on computer, communication and electrical technology, pp 92–97
82. Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev 65(6):386
83. Ross Q (1986) Induction of decision trees. Mach Learn 1(1):81–106
84. Rumelhart E, Hinton E, Williams J (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536
85. Schliebs S, Kasabov N, Defoin-platel M (2010) On the probabilistic optimization of spiking neural networks. Int J Neural Syst 20(6):481–500
86. Schliebs S, Kasabov N (2013) Evolving spiking neural network - a survey. Evolv Syst 4(2):87–98
87. Shen D, Xin C, Nguyen M, Yan W (2018) Flame detection using deep learning. In: ICCAR'18, pp 416–420
88. Shi X, Zhao W, Shen Y (2005) Automatic license plate recognition system based on colour image processing. In: Computational science and its applications, pp 1159–1168
89. Silver D, Huang A, Maddison J et al (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529(7587):484–489
90. Stauffer C, Eric W, Grimson L (2000) Learning patterns of activity using real-time tracking. IEEE PAMI 22(8):747–757
91. Stringa E, Regazzoni CS (1998) Content-based retrieval and real-time detection from video sequences acquired by surveillance systems. In: IEEE ICIP, pp 138–142
92. Suryanarayana P, Mitra SK, Banerjee A, Roy AK (2005) A morphology based approach for car license plate extraction. In: IEEE INDICON, pp 24–27
93. Tarabek P (2012) A real-time license plate localization method based on vertical edge analysis. In: Federated conference on computer science and information systemsm, pp 149–154
94. Wang J (2016) Event-driven traffic ticketing system. Masters Thesis, Auckland University of Technology, Auckland
95. Wang J, Bacic B, Yan W (2017) An effective method for plate number recognition. Springer Multimedia Tools and Applications (Online Publication)
96. Wang J, Yan W (2016) BP-neural network for number plate recognition. Int J Digital Crime Forensics 8(3)
97. Wang X, Zhou M, Geng G (2004) An approach of vehicle plate extract based on HSV colour space. Comput Eng 17
98. Wei W, Li Y, Wang M, Huang Z (2001) Research on number-plate recognition based on neural networks. In: IEEE Signal processing society workshop, pp 529–538
99. Wu H-HP, Chen H-H, Wu R-J, Shen D-F (2006) License plate extraction in low resolution video. Int Conf Pattern Recognit 1:824–827
100. Xu J-F, Li S-F, Yu M-S (2004) Car license plate extraction using color and edge information
101. Yoo JH, Chun BT, Shin DP (1994) A neural network for recognizing characters extracted from moving vehicles. In: World congress on neural networks, pp 162–166

102. Yuan Y, Zou W, Zhao Y, Wang X, Hu X, Komodakis N (2017) A robust and efficient approach to license plate detection. IEEE Trans Image Process 26(3):1102–1114
103. Zhai X, Benssali F, Ramalingam S (2010) License plate localisation based on morphological operations. In: International conference on control automation robotics and vision, pp 1128–1132
104. Zheng D, Zhao Y, Wang J (2005) An efficient method of license plate location. Pattern Recognit Lett 26(15):2431–2438

5 Biometrics for Surveillance

5.1 Introduction to Biometrics

Biometrics [63,64] comprehends fingerprints [110], hand geometry [25,112], earlobe geometry [121], retina and iris patterns, voice waves, DNA, signatures, etc. [3, 7,89]. The biometric information is captured from human physical body and natural behaviors such as gait, keystroke, voice, lipreading [118], human signature, [114], etc. In general, everybody has his or her own biometrics but different individuals absolutely have clearly distinct computable features. The covering and discriminative attributes are the basestone what we should study for biometrics. In the following, we will detail each biometric one by one.

Human Face. Face recognition is one of the most important biometrics in computer vision [14,71]. It has been broadly used in fields such as surveillance, information security, identification systems, and law enforcement systems. In face recognition, the system needs to firstly detect a human face and then recognize the face using a classifier such as SVM assisted by a training set [102]. The classical algorithms comprise principal component analysis (PCA) [149], linear discriminant analysis (LDA), etc. [8,23,31,33,36,86]. Automated face analytics such as face detection, face recognition, and facial expression recognition are useful in recent security and forensics.

In PCA algorithm, we need to calculate eigenvalues [73,132,149],

$$\mathbf{X}^{-1}A\mathbf{X} = \Lambda = diag\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$$

where $0 \le \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ are roots of the Eq. (5.1), $\mathbf{X}$ is an eigenvector matrix,

$$f(\lambda) = 0 \tag{5.1}$$

W. Q. Yan, *Introduction to Intelligent Surveillance*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-10713-0_5

where $f(\lambda)$ is the characteristic polynomial,

$$f(\lambda) = det(A - I\lambda) = \sum_{i=0}^{n} a_i \lambda^i (a_n \neq 0)$$

For example, if a matrix is,

$$\mathbf{A} = \begin{pmatrix} 1 & 3 \\ 0 & 2 \end{pmatrix}$$

then

$$f(\lambda) = det(\mathbf{A} - I \cdot \lambda) = \begin{vmatrix} 1-\lambda & 3 \\ 0 & 2-\lambda \end{vmatrix} = (1-\lambda)\cdot(2-\lambda) = 0$$

Hence,

$$\lambda_1 = 1; \lambda_2 = 2;$$

Face recognition is a nonintrusive method, facial images are probably the most commonly used biometric characteristics to make out personal identity. The applications of face recognition range from a static controlled "mugshot" authentication to a dynamic, uncontrolled face identification in a cluttered background [21,49,132].

For the details of face recognition:

- Location and shape of facial components, such as eyes, eyebrows, nose, lips and chin, and the spatial relationships, are shown in Fig. 5.1, respectively [3,6].
- Security authentication systems require a fixed and simple background or special illumination. These systems also have difficulties in matching face images captured from two drastically different views. It is questionable whether the human face, without any contextual information, is a sufficient basis for recognizing a person from a large number of identities with an extremely high level of confidence.
- A face recognition system should be automated; an example of human face detection based on OpenCV (http://opencv.org/) is shown in Fig. 5.2.

However, as one of the object recognition problems in digital image processing, face recognition still suffers from problems such as luminance changes, pose changes, making-up, complex environments, head rotation, and aging issues [46,106,108]. Most of these problems are still under investigation.

OpenCV is a software platform in computer vision, and human face detection and parts detection have been well developed; the training results are available for detecting human mouth, left eye and right eye, noise detection, etc. [6]. OpenCV including source code could be downloaded from http://opencv.org.

OpenCV adopts the cascade face detection algorithm which is extremely fast with efficient feature selection; the face detector is a scale and location invariant detector

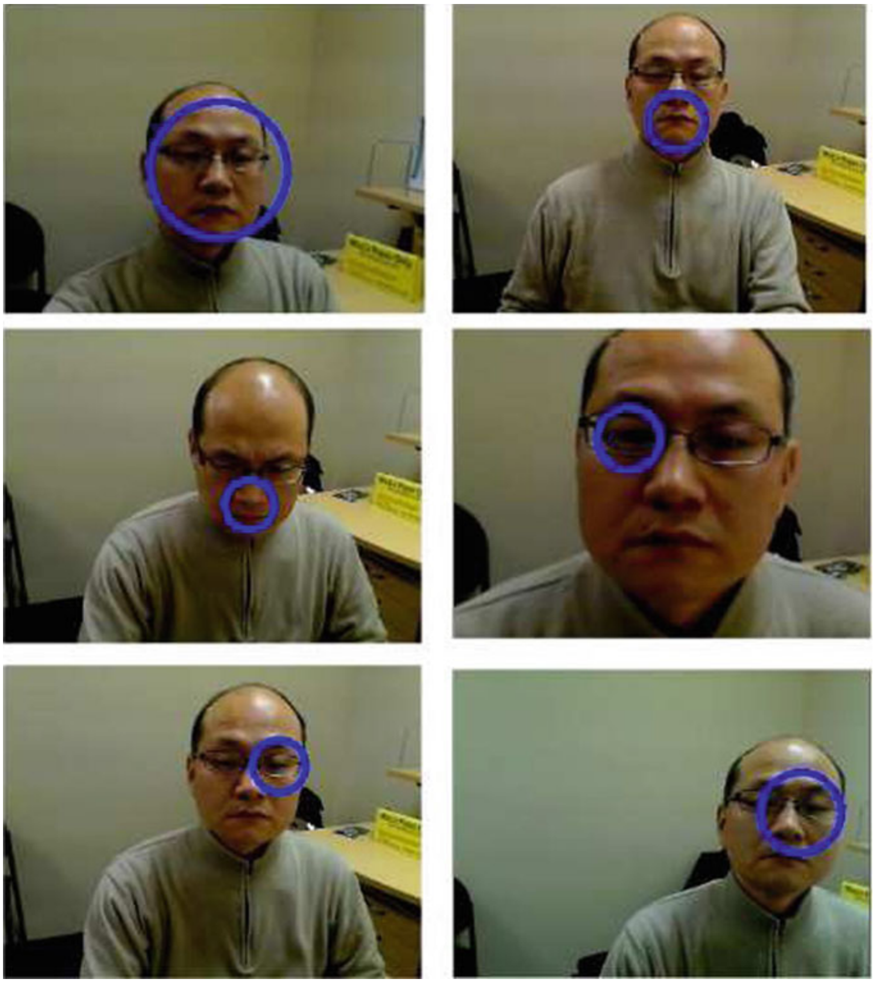


Fig. 5.1 Detection of human facial parts using OpenCV

which has been well trained. Instead of scaling the image itself (e.g., pyramid filters, etc.), it scales the features [135].

In face detection, Viola–Jones algorithm could be applied to detect other types of objects such as cars and hands. The algorithm takes use of Haar feature selection, creates integral image as the main feature, and adopts AdaBoost training algorithm. The salient contrast at the eye region and nose bridge region of human face is assumed as the feature for the human face detection adopted for AdaBoost training. The integral image has been applied to multiresolution-based face region search [141]; hence it is hierarchal. The cascade classifier takes out the regions with failure detection; those only successfully passed the classification will be remained for further consideration.

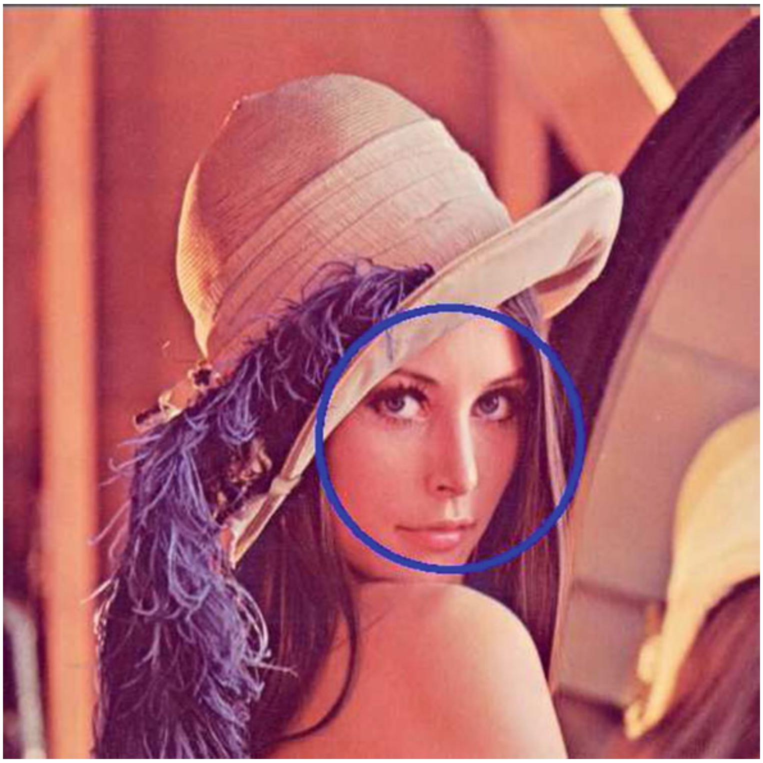


Fig. 5.2 OpenCV for human face detection

Since the detected regions will be merged together after the detection, in OpenCV applications or mobile Apps [4], we only see one rectangle of the overlapped regions.

Meanwhile, Viola–Jones algorithm is most effective one only on frontal images of a given face; it can hardly cope with 45° face rotation around both the vertical and horizontal axes [145]. It is sensitive to lighting conditions; we might get multiple results of the same face due to overlapping sub-windows [2].

Fingerprint. Our human uses fingerprints for personal identification in long history, and the matching accuracy using fingerprints has been proved extremely high [89]. A fingerprint is the pattern of ridges and valleys on the surface of a fingertip. Fingerprints of identical twins are different and so are the prints on each finger of the same person. Accuracy of currently available fingerprint recognition systems is adequate for security authentication. Multiple fingerprints of a person provide additional information to allow for large-scale identification involving millions of identities.

Iris. Iris is annular region of our eye bounded by the pupil and sclera on either side. Visual texture of the iris is formed during fetal development and stabilizes during the first two years of life [3]. The complicated iris texture carries very distinctive information for personal recognition. The iris-based recognition systems are promising and point to the feasibility of large-scale identification systems. Each iris is believed to be distinctive like fingerprints, even the irises of identical twins are expected to be different. It is extremely difficult to surgically tamper the texture of an iris. Although the early iris-based recognition systems required considerable user participation and were expensive, the newest systems have become more user-friendly and cost-effective. While iris systems have a very low false acceptance rate (FAR) compared to other biometric traits [129], false reject rate (FRR) of these systems could be very high [131].

Keystroke. This behavioral biometrics is expected to offer sufficient discriminatory information that permits identity verification. Keystroke dynamics is a behavioral information; for individuals, one may expect to observe large variations in typical typing patterns [62]. Furthermore, the keystrokes of a person could be monitored unobtrusively as the person is entering. However, this biometric permits "continuous verification" of an individual over a period of time.

Signature. The way a person signs his or her name is known to be characteristic of that individual. Signatures require contact with the writing instrument and an effort on the part of users that has been accepted in government, legal, and commercial transactions as a method of authentication. Signatures are a behavioral biometrics influenced by physical and emotional conditions of the signatories. Signatures vary substantially: Even successive impressions of the signature are significantly different [114]. Furthermore, professional forgers may be able to reproduce signatures that fool the system.

Voice. Voice is a combination of physical and behavioral biometrics [93]. The features of an individual's voice are based on shape and size of the appendages (e.g., vocal tracts, mouth, nasal cavities, and lips) that are used in synthesis of the sound. These physical characteristics of human speech are invariant for an individual, but speech of the same person changes over time due to aging or medical conditions, emotional state, etc. Voice is also not very distinctive and may not be appropriate for large-scale identification. A disadvantage of voice-based recognition is that speech features are sensitive to a number of factors such as background noise. Speech recognition is most appropriate in phone-based applications but the voice over phone is typically degraded in quality due to digital communication [17].

Lipreading. Lipreading is the process of observing lip movements of a speaker with the aim to interpret speech, especially when recorded voice may not be available or full of noises [29, 118]. The detection of lipreading is to find the approximate location of the lip in motion picture sequences. It is dependent on face detection to position lips. Usually, a color image in RGB color space is used to determine where the lip pixels are and how the clips are moving in spatiotemporal domain [3].

The features of lipreadings contain contours based on edge and position of the keypoints of moving lips and outward shapes or texture of lips. Active appearance model (AAM) is usually employed in order to extract the internal and external lip contours. The AAM comprises 12 points on the internal lip contour and 16 points on the external lip contour; meanwhile, viseme grouping has been highlighted in lipreading as well [41].

Lip recognition [41] adopts the methods such as template matching, hidden Markov model (HMM) [16,150], time-delayed neural network model, self-organizing map neural network (SOMNN) model, and mixed recognition methods. Because a simple recognition method has low recognition rate, a plenty of lipreading systems are to mix a variety of methods together so as to improve the precision and recall. The combination of HMM and ANN recognition methods has been proposed to achieve a better result [67,77,96,127].

Biometrics have the unique features from other evidences which have been widely applied to digital surveillance and forensics. Biometrics have been employed to identify and authenticate after authorization [15]. Identification means whether the biometric information is from the same person. After identification, we authenticate the person authorization to access the specific domain.

On the other hand, biometric is individual and personal based; therefore, it has ethics and privacy problems; permissions must be authorized before any public or official usages [42]. The biometrics of children, women, and disabled are extremely emphasized [13].

5.2 Biometrics Characteristics

5.2.1 Measurement and Analysis

Human bodies have different features which are measurable and computable, such as lip motion, gait and gesture, and body or sign language [5]. The characteristics help special individuals in special environments, such as military, noisy stadium, or quiet required spots.

Biometrics have computable features which are applied to find the unique person, namely identification. Nowadays, most passports are with indispensable biometric information for border check in airports or railway stations which is a kind of popular authentication. The features of biometrics should be discriminative and covering in pattern recognition. We observed that the biometrics have the following features:

1. *Convenient and ubiquity*. We usually bring biometrics anywhere without special requirements. The biometrics are with our human body essentially and individually.
2. *Automatically available and measurable*. Biometrics are available and measurable without harmfulness and are used for calculating once the features are captured and ready for use.
3. *Unalterability, unreplaceable or unchangeable*. Most of human biometrics will accompany with us forever after our birth. Even though we are growing up or becoming old, the biometrics will not be altered.

5.2.2 Fingerprint

The pattern of a fingerprint minutiae forms a valid representation of fingerprint [89]. This representation is compact and captures a significant component of information in fingerprints; compared to others, minutiae extraction is relatively robust to various sources of fingerprint degradation. Most types of minutiae in fingerprint images are stable and can be reliably identified automatically. The most widely used features are based on ridge ending and ridge bifurcation.

Given a representation scheme (e.g., minutiae distribution) and a similarity measure (e.g., string matching), there are two approaches for determining the individuality of fingerprints. Among empirical approaches, representative samples of fingerprints are collected, and a typical fingerprint matcher is used for calculating similarity; accuracy of the matcher on samples provides an indication of the uniqueness of fingerprint with respect to the matcher. However, there are known problems and costs associated with the collection of representative samples.

In police station, if policemen want to search a person, usually they use the computable features. Namely, if a person is passing by a camera, most of his information will be recorded and stored into a computer for searching in real time.

A majority of applications of biometrics are face detection and recognition. The results have been used in mobile technology [4]. From an obtained human face, a computer could infer the person's age [130], ethics, gender, skin [72], hairstyle, etc. If the computer has a watch list, biometric [40] information will help us find out the relevant records for verification and identification [15].

In biometrics, the challenges in research include acquisition conditions (illumination, expression), additional variations (disguises, occlusions), and aging. In indoor environment, the biometric information is acquired very easily, but in outdoor, the information will be changed with illumination and weather conditions [99].

5.2.3 Palm Print

Palm print refers to the area from fingertips to the wrist palm; it features from wrinkles, ridge ending, triangular point, etc. Palm print has bigger area and texture; we could use easily available device to capture the resources. On the other hand, using palm print for people identification is much robust and reliable.

There are four different types of methods which are used for palm print analysis, namely structure-based methods, statistics-based methods, subspace-based methods, and coding-based methods. In the early stage, structure-based methods mainly used direction and position of the principle line and wrinkles on a palm achieve the recognition. Statistics-based methods are using mean and variance of palm prints, etc., as the features to identify the local statistics and global statistics [146]. Subspace-based methods mainly include independent component analysis (ICA) [86], principal component analysis (PCA), and linear discriminant analysis (LDA). Coding-based methods using Gabor wavelet and Fourier transform have higher accuracy and faster matching in speed which are regarded as the best method among these four [137].

Palm print recognition as one of the biometric recognitions has been developed for decades [137, 146, 147]. The achievements from palm print recognition have also made a significant progress, especially in the contactless approaches and algorithms related to palm print recognition [151]. The methods for evaluating palm print recognition are true acceptance rate (TAR), false rejection rate (FRR), false acceptance rate (FAR), and discriminating index [7], which also include feature size, feature extraction, matching time, etc.

The features of palm prints are categorized into geometrical feature, principal lines, wrinkle, delta points, and minutiae. Specifically, geometrical features represent the characteristics of palm such as width and length of a human palm. Principal lines are evident and primary lines in the palm such as lifeline, love line, and heart line. Wrinkles are the other lines in palm, excluding the principal lines. The definition of delta points is a special district in the center of a palm.

5.2.4 Human Face

The state of the art of biometric technology for face detection is using AdaBoost which seeks the best approach for human face detection and recognition. The typical algorithm for biometrics is principal component analysis (PCA) [8], linear discriminate analysis (LDA) [84], Fisher's linear discriminate [85], and elastic graph matching (EGM).

AdaBoost is adaptive by iteratively reducing misclassifications. The weights of misclassified descriptors are adjusted for the benefit of generating a combined application of the given weak classifiers at the end.

Linear discriminant analysis (LDA) and the related Fisher's linear discriminant are methods used to find a linear combination of features which characterizes or separates two or more classes of objects. The combination may be used as a linear classifier or more commonly for dimensionality reduction before later classification. LDA is closely related to the analysis of variance and regression, which also expresses one dependent variable as a linear combination of other features or measurements.

In biometrics, PIE refers to pose, illumination, and expression for live recognition. 3D morphable face model is for independent pose recognition, low-dimensional spherical harmonic representations, and illumination-insensitive face recognition. Currently, the multi-PIE face database from Carnegie Mellon University in Pittsburgh is the most extensive one in terms of systematic variation of imaging parameters, including expressions, and therefore is the most suitable dataset. Other well-known databases include Yale Face Database, CMU FIA database, MIT CBCL database, NIST Mugshot Identification Database, etc. (http://www.face-rec.org/databases/) [106, 108].

For human face detection, two issues should be taken into considerations: (1) reduction in data dimensionality because of the famous curse of dimensionality problem and (2) feature selection. The typical algorithm of face detection could be listed as below:

Step 1: Prepare the data.
Step 2: Subtract the mean of the overall faces.
Step 3: Calculate the covariance matrix.
Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix.
Step 5: Select the principal components.
Step 6: Compare the faces reconstructed using the selected principal components.

In Viola–Jones face detection, we observed that the eye region of our face is darker than the upper cheeks; this has been reflected in Haar feature which is applied to face detection.

Generally, we have the following four steps for face detection using the Haar feature: (1) Haar feature selection, (2) creating integral image, (3) AdaBoost training algorithm, and (4) cascaded classifiers.

The advantages of Viola–Jones face detection include (1) extremely fast feature computation; (2) efficient in feature selection; (3) scale and location invariant detector; (4) instead of scaling the image itself (e.g., pyramid filters), we scale the features; and (5) we could use it to detect other types of objects such as cars and hands.

The disadvantages of Viola–Jones face detection are thought as: (1) The detector is most effective only on frontal images of faces; (2) it can hardly cope with 45° face rotation around both the vertical and horizontal axes; (3) it is sensitive to lighting conditions [2].

5.2.5 Gender Profiling

Although it is a fundamental task for surveillance applications to determine gender, however, normal video algorithms for gender profiling (usually face profiling) have three drawbacks. (1) The profiling result is always uncertain. (2) For a time-lasting gender profiling algorithm, the result is not stable. The degree of certainty usually varies, sometimes even to the extent that a male is classified as a female and vice versa. (3) A robust profiling results in the cases that a person's face is invisible; other features, such as body shape, are required. These algorithms may provide different recognition results at the very least which will provide different degrees of certainties.

Dempster–Shafer (DS) theory is a popular framework to deal with uncertain or incomplete information from multiple sources. This theory is capable of modeling incomplete information through ignorance. For combining different pieces of information, DS theory distinguishes two cases, i.e., whether a piece of information is from distinct or nondistinct sources. Therefore, gender profiling results from the same classifier, e.g., face-based, are considered from nondistinct sources, while profiling results from different classifiers are naturally considered as from distinct sources.

5.2.6 EBGM

Elastic bunch graph matching (EBGM) is an algorithm in computer vision for recognizing objects from an image based on a graphical representation. It has been prominently used in face recognition but also for gestures and other object classifications. EBGM is defined as an optimization problem of two-dimensional warping specifying corresponding pixels between subjected images.

EBGM is an extension to elastic graph matching for object classes with a common structure, such as faces in identical pose [55]. All instances of such a class are represented by the same type of graph. From these graphs, a bunch graph of the same structure is created with the nodes representing local textures of any objects in the class, e.g., all variants of a left eye, and the edges represent average distances between the node locations, e.g., the average distance between the two eyes. Thus, a bunch graph is an abstraction for representing object classes rather than individual objects.

EBGM is only applied to objects with a common structure, such as faces in frontal pose, sharing a common set of landmarks like the tip of a nose or a corner of an eye [107]. For recognition of arbitrary objects, in the absence of landmarks, the graphs are required to be dynamic with respect to both shape and attributed features. A graphic dynamics that allows generic object representations, model or bunch graphs, emerges from a collection of arbitrary objects. The idea is to extract typical local texture arrangements from the objects and provide the rules to compose them as needed to represent new objects.

5.2.7 Twin Recognition

Nowadays, biometrics apply behavioral characteristics to identify individuals. Among the identifications, twins are very tough to be identified because they resemble each other very much [10,88,134].

Human visual system can work better than computational machines in twin recognition. Human is good at finding facial marks such as moles and scars so as to effectively identify twins [126]. In addition, if human beings have much time to examine the difference carefully, the performance will become much better [12].

Recently, twin recognition in association with human earmark technology becomes a new class of relatively stable biometrics that has drawn attention of a number of researchers. Human ear recognition as a biometrics does not have significant changes; therefore, it could be regarded as an effective biometrics [30]. As other pattern recognition methods, ear recognition has its advantages and disadvantages. Compared to face recognition, ear recognition is seldom affected by human emotions and aging. Methods of ear recognition have twofold: statistical-based method and geometric-based method [75].

Statistical-based methods analyze a human ear image by using statistical tools [121]. The ear image is treated as a matrix, and then, the method like PCA is taken to deal with the features and reduce the redundancy of the data. PCA as

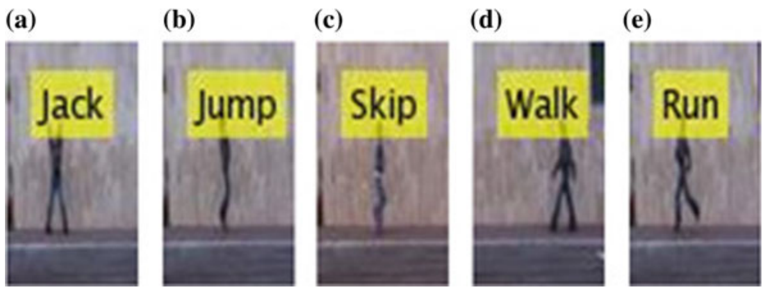


Fig. 5.3 Results of human behavior recognition

a statistical-based method was developed to extract features. The related work also analyzes performance with the changes in human aging, illumination, and pose [78]. In addition, features of human faces and ears are extracted by using PCA.

Geometric-based methods take use of the shapes of human ear to identify twins. There are two very convenient methods, concentric circle method (CCM) and contour tracing method (CTM). The center point of a human ear is its center of mass, concentric circles are created by this point and predefined radius. Thus, the intersections of these circles and ear edges are thought as keypoints of the ear. CTM is the method to trace the ear edges so as to find the contour of bifurcation, intersection, and ending. In the work, CCM is found better than that of CTM [70,92].

5.2.8 Pedestrians

Pedestrians [39,116] are normal moving objects on street who have the characteristics such as walking with legs, human body, and head. While a person is moving along the street, the scene and background also keep changing [87].

An example of pedestrian recognition is shown in Fig. 5.3.

To achieve video classification of human exercising activities (walking, skipping, running, jacking, and jumping) [48,56,69], we consider visual feature extraction having spatial and temporal information between adjacent video frames. HOG is such a visual feature which is adopted for object recognition in computer vision. As we know HOG descriptors provide a better performance than others for human behavior detection and recognition.

HOG features for human detection have been trained and tested after normalization, gradient computation, and spatial organization [32]. The main idea of HOG descriptor is to calculate the occurrences of gradient orientation in localized portions of an image. The implementation of HOG descriptors has been achieved by segmenting the image into small connection regions which are called "cells." Each cell generates a histogram of oriented gradients or cell edge direction of the pixel; the combination of these histograms can be applied to express descriptor.

HOG descriptor has the advantages which effectively describe local shape of the image. By using the number of bins and cell size of the histogram, HOG is able to reflect an image character of the local region, to describe precise feature information, and to keep features invariant.

Local binary pattern (LBP) is an operator to describe the local texture feature of an image. The advantage of this operator is the characteristic of rotational invariance and grayscale invariance. Therefore, LBP is a simple and effective algorithm for feature extraction. The extraction of LBP depends on the grayscale. For one target pixel, there are more than eight pixels around it; if the grayscale level of surrounding pixels is greater than the grayscale level of the pixel in the middle, they will be marked by '1.' On the contrary, they will be marked by '0.' The eight-bit binary number will be the LBP feature of the pixel in the central. For LBP feature extraction in practice, the original image will be divided into a number of cells with a fixed size (16×16). After that, we apply this method to obtain the eight-bit binary number and then calculate the histogram for each cell and applied normalization to the histograms. Finally, we compose a feature vector of the whole image by using the histogram as the vector elements.

In pedestrian detection [153], we usually need to locate an object candidate, provide specific split function, and outline a post-processing step [116]. Localizing regions of interest (ROI) is to identify potential candidates for bounding boxes by passing them through the trees of RDF (Random Decision Forest).

The results of classification could provide more than one candidate window around a person in an image. For tracking and positioning [53], it is meaningful to merge them into one. Each positive window has the corresponding probability assigned by using strong classifier generated by RDF; the one with the highest probability is chosen. Mean shift could be applied to specify the final detected region of an object [47].

5.2.9 Suspects

If we put human biometrics: face, hair, motion, and actions together, we could find a wanted person automatically in police station by using biometrics. On the bulletin of a wanted person or social networks, usually the suspect's information is shown to the public. The suspect's features include gender, age, nationality, language, height, weight, color of eyes, color of hair, length of the hair, eyebrows, nose, mouth, etc. These features could be calculated from surveillance videos and images by using the algorithms provided in this chapter. Once the features have been saved in a database, it helps us search for the wanted person in airports, railway stations, and ferry terminals.

As the safety of individuals is always one of the main foci of public security, it appears to be crucially significant for tracking suspects under surveillance. However, it is difficult for suspects being monitored exclusively by human beings. With the rapid development of technologies in intelligent surveillance, officers started identifying and detecting traces of suspects by using intelligent surveillance systems [43]. Specifically, there are two main parts in suspect identification: human feature extraction and feature matching for assigned suspects [15]. Well-known techniques related to feature extraction are feature point detection, edge detection, scale invariance feature transform (SIFT), and representation of colors in computer vision [71].

5.3 Human Actions, Behaviors, and Gestures

Surveillance is a reality; human actions include walking, jogging, running, boxing, waving, clapping, etc. [60,60,142]. Actions and behaviors are closely related to time line and have life-span. We therefore could draw our behaviors in a spatiotemporal volume; trajectories of an object also could be used to present human behavior and actions [74,94].

Gesture recognition is composed of multiple technologies and techniques that arise from the use of physical input of human body or hand movement [34,115,139] and later on identifying gestures [80] through cameras without the need of a physical input device [5,11,24,95].

There are two types of gestures: one is static gesture, which has the abilities to recognize a shape or pattern of a hand [25,59,104,140], and body limb or facial recognition; the models are used including template matching, neuronal networks, or pattern recognition techniques [128].

Another gesture recognition type is dynamic recognition [44,152]. This is a series of postures being recognized over a short period of time; in this type of recognition, each video frame composes the posture; the sequence of video frames defines the gesture to interact with the computer [79,125]; dynamic recognition techniques are used such as time compressing templates, dynamic time warping, hidden Markov (HMM) models [16], deep learning [80], conditional random fields (CRF), time delay neural networks, and particle filtering [58]. Finite state machine (FSM) also could be set up for detecting human gestures.

Human actions and behaviors have mean and variance [60,61]. Mean is available from the average of human behavior after repetitions; meanwhile, variances could tell us how far each behavior or action is from the mean.

The typical method of human actions and behavior analysis is moment; we use Mahalanobis distance to find the difference; we could pick the nearest one. Usually, we calculate the moments in each action class with a Gaussian distribution (diagonal covariance) and then measure the Mahalanobis distance $d = \sqrt{(x-y)^T S^{-1}(x-y)}$ to all classes; finally, we pick up the nearest one. Recently, deep neural networks and sequential tensor decomposition have been applied to human action recognition [53].

Optical flow is based on video motion estimation [45]. In MPEG, motion vectors which reflect the optical flow are stored in MPEG video. Using the motion vectors, we could estimate object or human motion and track the objects in surveillance videos [54,138].

In static gesture recognition, the method made use of dynamic time warping (DTW) algorithms combined with k-NN classification. This combination was chosen because it could be implemented with ease and adapt to different users and varying gesture types [136,148].

Vision-based gesture recognition has been investigated for a long time, Fujitsu Laboratories completed the identification of 46 gesture symbols in 1991. A fingertip with a bright visual colored glove is applied to finger gesture recognition as an input; seven gestures can be successfully identified [82].

A static-hand recognition of international sign language was implemented by using principal component analysis (PCA). The methods are able to recognize 25 hand gestures [100].

Support vector machine (SVM) is a classical classifier in machine learning. In finger gesture recognition, it was usually applied to decide when to start the finger tracking by using a range of colors, shapes, and motions. The standard SVM algorithm was developed for pattern classification.

Artificial neural networks (ANNs) simulate our human brain in pattern classification which train the network nodes deployed on multiple layers as the necessary elements for finger gesture recognition [98]. ANNs have the characteristics of pattern classification with anti-interference. An ANN network in gesture recognition has been developed by using a three-layer neural network, including 13 input nodes, 100 hidden nodes, and 42 output nodes [40].

A sign gesture recognition system was developed by using recurrent neural networks (RNNs) in deep learning; the system can recognize 42 symbols.

After ANN algorithms were put forward to practice, it has been greatly improved and generalized, including replacement of error function, dynamic adjustment of network topology, learning rate, and factor parameters. The future development of ANNs can further reduce the complexity to enhance the extractability of ANN training and the applicability of the algorithms [40,57,83].

5.4 Human Privacy and Ethics

Privacy is human right to control what happens with personal data [19,20,52]. The meaning of privacy may differ throughout cultures, but the general concept is that privacy means wanting to keep information unnoticed or unidentified from the general public [66,111].

Surveillance privacy protection (SPP) is a realistic [101] in the real world today [26]. The surveillance data carries confidential or privacy information which is related to personal secrets, children, relatives, location and time privacy, appearance and wearing, emotions and intentions, etc. [37]. Therefore, surveillance privacy does exist [76].

In publishing data privacy over World Wide Web, preserving data privacy usually refers to release more anonymous records, such as k-anonymity; meanwhile, we could keep the k records having l-diverse. The difference of these l-diverse records could have t-closeness in probability or entropy. Analogously, in surveillance privacy, the sensitive regions of a picture usually are obscured by mosaics or blurred mask to enhance anonymities. The masks may be calculated by using pixelization so as to reach the effects such as l-diverse and t-closeness in histogram and entropy in digital image processing.

Surveillance data privacy preservation is different from censorship systems of a state [91,113]. In censorship, a rating system usually classifies movies or TV dramas, computer games, and literature into several classifications: general audiences (G), parental guidance suggested (PG), restricted (R), etc. The media that contained objectionable, harmful, sensitive, politically incorrect, or inconvenient media will be banned by a government or an organization [22].

Due to privacy issue, the concern about the conflict between security and privacy for surveillance is on the rise [120]. Therefore, over the past few years, research work to protect privacy in surveillance systems has been made [90]. They have proposed various approaches for privacy-protecting goal including using distortion filters to pixelize, blur, blackout, silhouette, clear, replace by generic object, and mask the object containing sensitive information which may explore privacy [9,123].

Privacy of surveillance video has been modeled by using the famous sigmoid function. The parameters used in this model are "who," "when," and "where" information; this is the reason why the applications are from surveillance. The detected pedestrian's face/head of surveillance video is obscured by encrypting with a unique key derived from a master key for privacy preservation purpose. A privacy preservation method that adopts adaptive data transformation involving the use of selective obfuscation and global operations to provide robust privacy was proposed. The approach intelligently hides evidence in a video without much compromise with quality [144].

Privacy of surveillance images may include human faces, number plate, street number, biometrics such as fingerprint and iris. Usually, these images are thought to have privacy information and should be taken with special care in media sharing [38, 81,97,124].

5.4.1 Pixelization and Gaussian Blurring

Pixelization is a technique used for modifying images or videos for privacy; it is achieved by noticeably lowering resolution in ROI, using a square block of pixels with its average [28]. The primary purpose is to use for censorship. It is commonly used in television news to obscure the object containing sensitive information such as the proper name of people, locations, or any other inappropriate discourse [27]. The advantage of using pixelization in surveillance system for privacy protection is very simple and easy to integrate in existing system [27,28,123]. On the other hand, the disadvantage is that the process is irreversible and the privacy information is lost [122]. The pixelization of image $I(x, y)$ is shown in Eq. (5.2).

$$\bar{I}(x, y) = \frac{1}{b^2}\sum_{i=0}^{b-1}\sum_{j=0}^{b-1} I\left(\left\lfloor \frac{x}{b} \right\rfloor \cdot b + i, \left\lfloor \frac{x}{b} \right\rfloor \cdot b + j\right) \tag{5.2}$$

where image pixel coordinates are x and y, block size is b, and $\lfloor\cdot\rfloor$ indicates the floored division.

Gaussian blurring is an approach widely used for privacy protection in surveillance; it removes details in ROI by using a Gaussian low-pass filter [9]. It is also taking advantage of edge detection [50]. The equation of Gaussian function in one dimension is shown in Eqs. (5.3) and (5.4).

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5.3}$$

where μ is the mean, σ is the variance, $x \sim N(\mu, \sigma)$.

In multidimension, it is shown in Eq. (5.4):

$$G(X) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} e^{-\frac{1}{2}(X-\mu)^T \Sigma^{-1} (X-\mu)} \tag{5.4}$$

where $|\Sigma|$ is the determinant of matrix Σ, $k = |X|$, $X = (x_1, x_2, \cdots, x_k)$, $X \sim N_k(\mu, \Sigma)$.

5.4.2 Audio Progressive Scrambling

Audio information in surveillance [109] like human voice usually has been modified in pitches [65] by using spectrum in digital signal processing [93]; the content could be clearly heard, but the speakers could not be identified; voices of male and female or adult and children especially are swapped [133]. Progressive techniques have been used for MP3 encryption [143]. During descrambling, provided the number of keys, a number of rounds of descrambling performed will decide the audio output quality. With a subset of keys, the audio is descrambled to obtain a low-quality wave. However, an audio clip having original quality is able to be obtained by using all of the keys.

Typical sounds may include privacy information such as footsteps, bath shower, toilet flushing, baby cooing, signature writing, and keyboard typing or keystroke into considerations and implement the progressive scrambling for the audio clips in spatial domain and frequency domain, respectively. Usually, these audio clips are thought to have the sensitive information and easily make a listener embarrassed.

Current research results have revealed that new privacy enabling technologies (PET) are promising with the prospective to successfully protect individual privacy, without hindering surveillance tasks. The final results confront the common surmise that increased security may overcome a failure of privacy [106, 108].

The W^3 survived assessment model [117] is essentially the foremost and very useful step toward privacy protection of individuals in multicamera video footage [52]. This work does set up directions for future research, for instance, to investigate ways to lessen the privacy loss with the minimum loss of efficacy in video quality [105].

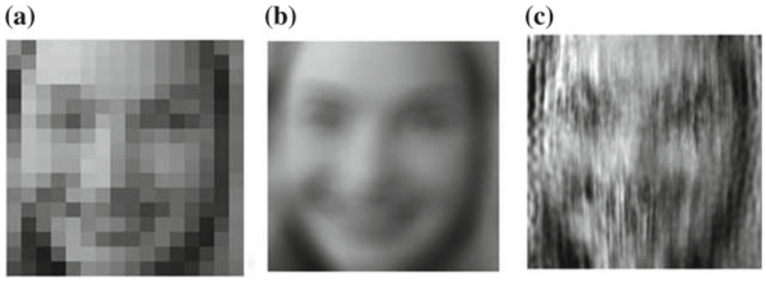


Fig. 5.4 Privacy preservation using: **a** mosaicking **b** Gaussian blurring **c** scrambling

5.4.3 Covering and Overlapping

The traditional ways to preserve privacy in visual surveillance are mosaicking, pixelization, or scrambling human face regions as shown in Fig. 5.4 [51]. But apparently this is not enough since from the acquired clothes, behaviors or gait, even from a contour, silhouette or blob, and several dots, we still are able to discern who this person is. Even if a face is not clearly seen from a very far distance, such as a basketball or soccer player, we are still able to infer who the person is. Particularly from those processed images, photos, or cartoon motion pictures, the similarity is still existential. This rolls out the motivation that the best solution of privacy preservation is to completely remove the privacy information from the video frames. However, the commitment will undoubtedly diminish the utility and visibility of surveillance videos.

Utility refers to video usage for various purposes. When an incident happens, we need to track back and search for the person and objects related to the incident. While conventional ways of privacy protection such as image mosaicking, blurring, and pixelizing easily provoke the content damaging, if the surveillance video frames have been completely obstructed, that is equivalent to acclaim that this video has to be cast aside. Thus, the situation requires us to find a way to leverage the utility, visibility, and privacy of surveillance videos. Our goal of this paper is to resolve this tough problem.

Therefore, we hope to find an effective way for preserving privacy [103]. For instance, in a typically monitored corridor, we use a walking Mickey Mouse to substitute a man for displaying purpose who is walking through from left to right or from right to left, and the man may perambulate to pass this site; thereafter, the mouse will be viewed in the correspondingly sluggish way such as entering, walking, standing, existing, and alarming. If it indeed has an incident, namely the alarming state is activated, only the authorized security staff has the privilege to review the surveillance events, but normally this analogy-based replacement for the purpose of privacy preservation is much reasonable for catering to unauthorized viewers [18].

From our observations, we find that surveillance events have their own patterns owing to the merits such as discriminative and covering. We therefore have the opportunity to seek the typical motion pictures with a specific pattern, such as the

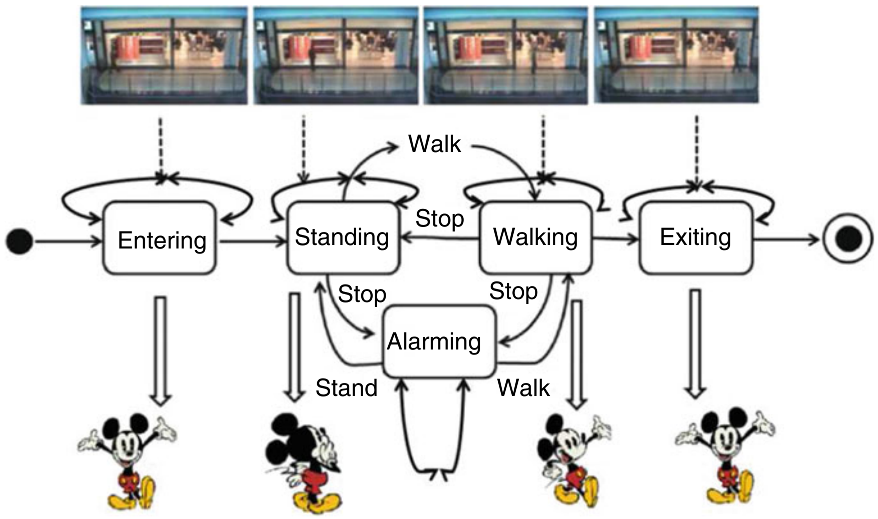


Fig. 5.5 A state diagram and video frames show an instance of privacy preservation in visual surveillance

cartoon GIF pictures which could be played iteratively and are suitable for presenting these surveillance events. Therefore, adjustment of these motion pictures is entailed to match the necessity of surveillance events.

$$d(H_1, H_2) = \sqrt{\frac{1}{3L} \sum_{l,i,k} \frac{|h^l_{1k}(i) - h^l_{2k}(i)|^2}{h^l_{1k}(i) + h^l_{2k}(i)}} \tag{5.5}$$

where an action is represented by a set of nine one-dimensional histograms: h^1_x, h^1_y, h^1_t, $h^2_x, h^2_y, h^2_t, h^3_x, h^3_y, h^3_t$ is the histogram of one component of space-time measurements N^l_k, L is total frame number of an image sequence.

$$(N^l_x, N^l_y, N^l_t) = \frac{(|S^l_x|, |S^l_y|, |S^l_t|)}{\sqrt{(S^l_x)^2 + (S^l_y)^2 + (S^l_t)^2}} \tag{5.6}$$

where (N^l_x, N^l_y, N^l_t) is the normalization of the spatiotemporal vector $(|S^l_x|, |S^l_y|, |S^l_t|)$ which indicates the motion of the video frame at the time t.

We adopted the videos provided in the surveillance dataset: CAVIAR to demonstrate a walker passing through a shop in a mall. The state diagram with video frames depicts the typical events of a walker when passing through a monitored corridor: *entering*, *standing*, *passing*, *alarming*, and *exiting*. The states could be switched between each other due to changes in the guard condition and actions. We detect the state changes; we find cartoon pictures presenting the similar states; finally, the privacy region on the surveillance video frames has been overlapped and the privacy

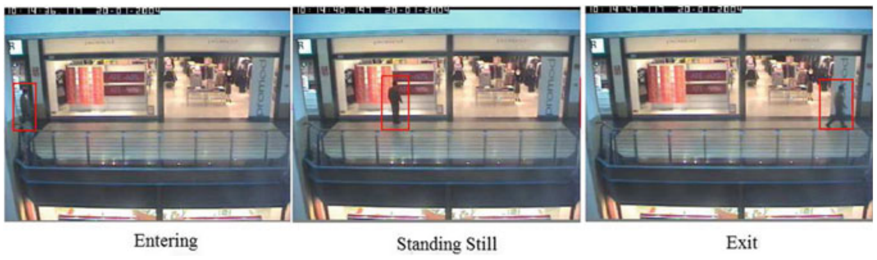


Fig. 5.6 Object tracking from *left* to *right* with standing state

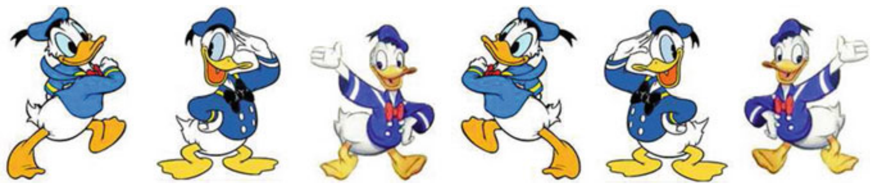


Fig. 5.7 Donald Duck's actions in virtual reality for representing surveillance events

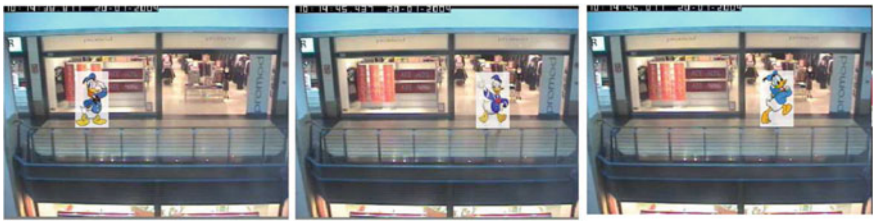


Fig. 5.8 Overlapping the moving object from *left* to *right*

of the event has been removed. This example indicates how we could leverage human privacy in surveillance (Fig. 5.5).

As shown in Figs. 5.6, 5.8, 5.9, and 5.10, we detect the moving object, track the object, and find the state changes of an event in a surveillance scenario.

In videos shown in Figs. 5.6 and 5.9, we detect the "entering," "standing still," and "exiting" states of the surveillance event [68, 119]; therefore, we could cover the moving object using cartoon characters (Fig. 5.6).

In Fig. 5.7, we find the cartoon pictures from public Web sites with swinging the right-hand, swinging the left-hand, and standing still in virtual reality; the six cartoons represent the states of two opposite walking directions: left to right and right to left through the corridor; the actions of cartoon characters could represent the states of surveillance events in real reality.

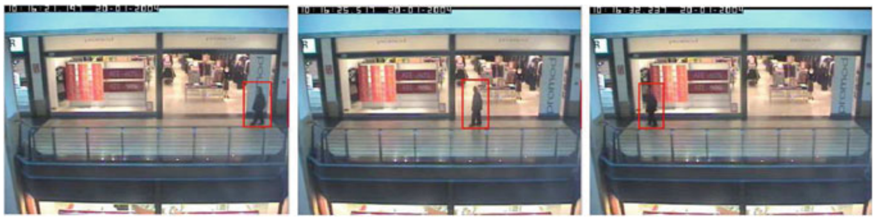


Fig. 5.9 Moving object tracking from *right* to *left*

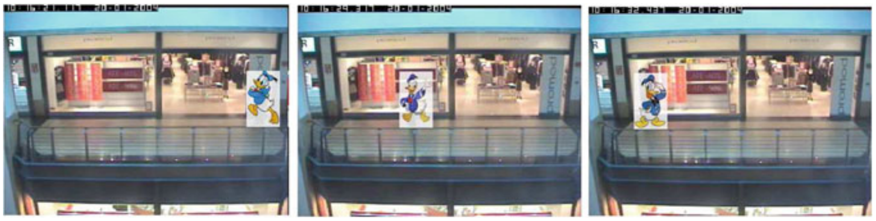


Fig. 5.10 Overlapping the moving object from *right* to *left*

5.5 Questions

Question 1. What are the differences between bioinformatics and biometrics?
Question 2. What are the unique features of biometrics? Why biometrics could be applied to intelligent surveillance?
Question 3. What are the core issues of pattern recognition in biometrics?
Question 4. In your personal experience, which biometric is the most reliable and robust one?

References

1. Adams A, Ferryman M (2012) The future of video analytics for surveillance and its ethical implications. Secur J 28(3):272–289
2. Akhtar Z, Rattani A (2017) A face in any form: new challenges and opportunities for face recognition technology. Computer 50(4):80–90
3. Aleksic PS (2009) Lip movement recognition. Encyclopedia of biometrics. Springer, New York, pp 904–908
4. Antoniou A, Angelov P (2016) A general purpose intelligent surveillance system for mobile devices using deep learning. In: International joint conference on neural networks (IJCNN)
5. Athitsos V, Wang H, Stefan A (2010) A database-based framework for gesture recognition. Pers Ubiquitous Comput 14(6):511–526
6. Azad R, Ahmadzadeh E, Azad B (2015) Real-time human face detection in noisy images based on skin color fusion model and eye detection. Intelligent computing, communication and devices, vol 309. Springer, Berlin, pp 435–447

7. Bansal A, Agarwal R, Sharma RK (2012) FAR and FRR based analysis of iris recognition system. In: IEEE international conference on signal processing, computing and control, pp 1–6
8. Bartlett MS, Movellan JR, Sejnowski TJ (2002) Face recognition by independent component analysis. IEEE Trans Neural Netw 13(6):1450–1464
9. Baym K (2009) A call for grounding in the face of blurred boundaries. J Comput-Mediat Commun 14(3):720–723
10. Behravan H, Faez K (2013) Introducing a new multimodal database from twins' biometric traits. In: Iranian conference on electrical engineering, pp 1–6
11. Billon R, Nedelec A, Tisseau J (2008) Gesture recognition in flow based on PCA analysis using multiagent system. In: Proceedings of the 2008 international conference on advances in computer entertainment technology, pp 139–146
12. Biswas S, Bowyer KW, Flynn PJ (2011) A study of face recognition of identical twins by humans. In: International workshop on information forensics and security, pp 1–6
13. Bowyer W (2004) Face recognition technology: security versus privacy. IEEE Technol Soc 23(1):9–19
14. Bruce V, Young A (1986) Understanding face recognition. Br J Psychol 77:305–327
15. Brunelli R, Falavigna D (1995) Person identification using multiple cues. IEEE Trans Pattern Anal Mach Intell 17(10):955966
16. Bui H, Venkatesh S, West W (2001) Tracking and surveillance in wide-area spatial environments using the abstract Hidden Markov model. Pattern Recognit 15(1):177–195
17. Burton DK (1987) Text-dependent speaker verification using vector quantization source-coding. IEEE Trans Acoust Speech Signal Process 35(2):133–143
18. Calvel C, Ehrette T, Richard G (2005) Event detection for audio-based surveillance system. In: ICME, pp 1306–1309
19. Carrillo P, Kalva H, Magliveras S (2008) Compression independent object encryption for ensuring privacy in video surveillance. In: ICME, pp 273–276
20. Cavallaro A (2007) Privacy in video surveillance. IEEE Signal Process 24(2):168–169
21. Chen C, Vijayan KA, Andrew DB (2015) Robust textural features for real time face recognition. In: Imaging and multimedia analytics in a web and mobile world (SPIE 9408)
22. Chen D, Chang Y, Yan R, Yang J (2007) Tools for protecting the privacy of specific individuals in video. EURASIP J Adv Signal Process 1(1):107–116
23. Chen T, Chen Y, Lee S, Huang F (1998) Discriminant analysis of principal components for face recognition. In: IEEE international conference on automatic face and gesture recognition, pp 336–341
24. Chen X, Koskela M (2013) Online RGB-D gesture recognition with extreme learning machines. In: ACM on international conference on multimodal interaction, pp 467–474
25. Cheng H, Luo J, Chen X (2014) A windowed dynamic time warping approach for 3D continuous hand gesture recognition. In: IEEE international conference on multimedia and expo, pp 1–6
26. Chinomi K, Nitta N, lto Y, Babaguchi N (2008) PriSurv: privacy protected video surveillance system using adaptive visual abstraction. Advances in multimedia modeling, vol 4903. Lecture Notes in Computer Science. Springer, Berlin, pp 144–154
27. Cotton D, Uson M (2007) Image pixelization and dynamic range. Natl Radio Astron Obs 1(1):3–10
28. Cotton D, Uson M (2008) Pixelization and dynamic range in radio interferometry. Astron Instrum 490(1):455–460
29. Chan MT, Zhang Y, Huang TS (1998) Real-time lip tracking and bimodal continuous speech recognition. In: IEEE workshop on multimedia signal processing, pp 65–70

30. Chang K, Bowyer KW, Sarkar S, Victor B (2003) Comparison and combination of ear and face images in appearance-based biometrics. IEEE Trans Pattern Anal Mach Intell 25(9):1160–1165
31. Delac K, Grgic M, Grgic S (2005) Statistics in face recognition: analyzing probability distributions of PCA, ICA and LDA performance results. In: International symposium on image and signal processing and analysis, pp 289–294
32. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: CVPR, pp 886–893
33. Delac K, Grgic M, Grgic S (2005) Independent comparative study of PCA, ICA, and LDA on the FERET data set. Int J Imaging Syst Technol 15(5):252–260
34. Dieckmann U, Plankensteiner P, Wagner T (1997) Sesam: a biometric person identification system using sensor fusion. Pattern Recognit Lett 18(9):827–833
35. Dong Y, Liu Y, Lian S (2016) Automatic age estimation based on deep learning algorithm. Neurocomputing 187:4–10
36. Draper BA, Baek K, Bartlett MS, Beveridge JR (2003) Recognizing faces with PCA and ICA. Comput Vis Image Underst 91(1–2):115–137
37. Dufaux F, Ebrahimi T (2008) H.264/AVC video scrambling for privacy protection. In: IEEE ICIP, pp 1688–1691
38. Dufaux F, Ebrahimi T (2010) A framework for the validation of privacy protection solutions in video surveillance. In: IEEE ICME, pp 66–71
39. Enzweiler M, Gavrila D (2008) Monocular pedestrian detection: survey and experiments. IEEE Trans Pattern Anal Mach Intell 31(12):2179–2195
40. Evermann J, Rehse J-R, Fettke P (2017) Predicting process behaviour using deep learning. Decis Support Syst 100:129–140
41. Faraj M, Bigun J (2007) Lip biometrics for digit recognition. Computer analysis of images and patterns, vol 4673. Lecture Notes in Computer Science. Springer, Berlin, pp 360–365
42. Fraser T (2004) Privacy law and video surveillance: guidance from the ontario courts. Mcinnes Cooper 3(1):10–13
43. Frischholz R, Dieckmann U (2000) Bioid: a multimodal biometric identification system. IEEE Comput 33(2):6468
44. Gadea C, Ionescu B, Ionescu D, Islam S, Solomon B (2012) Finger-based gesture control of a collaborative online workspace. In: IEEE international symposium on applied computational intelligence and informatics, pp 41–46
45. Gavrila M (1993) The analysis of human motion and its application for visual surveillance. Comput Vis Image Underst 73(1):82–98
46. Georghiades AS, Belhumeur PN, Kriegman DJ (2001) From few to many: illumination cone models for face recognition under variable lighting and pose. IEEE Trans Pattern Anal Mach Intell 23(6):643–660
47. Golfarelli M, Maio D, Maltoni D (1997) On the error-reject tradeoff in biometric verification systems. IEEE Trans Pattern Anal Mach Intell 19(7):786796
48. Gong S, Xiang T (2003) Recognition of group activities using dynamic probabilistic networks. In: IEEE ICCV, pp 742–750
49. Gottumukkal R, Asari VK (2004) An improved face recognition technique based on modular PCA approach. Pattern Recognit Lett 25(4):429–436
50. Gouaillier V (2009) Intelligent video surveillance: promises and challenges technological and commercial intelligence report. CRIM Technople Def Secur 3(2):9–68
51. Gu S, Han Q (2006) The application of chaos and DWT in image scrambling. In: Machine learning and cybernetics, pp 3729–3733
52. Gulzar N, Abbasi B, Wu E, Ozbal A, Yan W (2013) Surveillance privacy protection. Intelligent multimedia surveillance: current trends and research. Springer, Berlin, pp 83–105

53. Guo H, Wu X, Feng W (2017) Multi-stream deep networks for human action classification with sequential tensor decomposition. Signal Process 140:198–206
54. Hampapur A, Brown L, Connell J, Ekin A, Haas N, Lu M, Merkl H, Pankanti S (2005) Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking. IEEE Signal Process 22(2):38–51
55. Hanmandlu M, Gupta D, Vasikarla S (2013) Face recognition using Elastic bunch graph matching. In: IEEE applied imagery pattern recognition workshop: sensing for control and augmentation, pp 1–7
56. Haritaoglu I, Harwood D, Davis LS (2000) W4: real-time surveillance of people and their activities. IEEE TPAMI 22(8):809–830
57. Herath S, Harandi M, Porikli F (2017) Going deeper into action recognition: a survey. Image Vis Comput 60:4–21
58. Hsieh CC, Liou DH, Lee D (2010) A real time hand gesture recognition system using motion history image. In: International conference on signal processing systems, vol 2, pp V2–394
59. Hu G, Gao Q (2011) Gesture analysis using 3D camera, shape features and particle filters. In: Canadian conference on computer and robot vision, pp 204–211
60. Iosifidis A, Tefas A (2012) View-invariant action recognition based on artificial neural networks. IEEE Trans Neural Netw Learn Syst 23(3):412–424
61. Ivanov Y, Bobick A (2000) Recognition of visual activities and interaction by stochastic parsing. IEEE Trans Pattern Anal Mach Intell 22(8):852872
62. Ishijima R, Ogawa K, Higuchi M, Komuro T (2014) Real-time typing action detection in a 3D pointing gesture interface. In: Augmented human international conference, pp 20
63. Jain AK (2007) Technology: biometric recognition. Nature 449(7158):38–40
64. Jain AK, Ross A, Prabhakar S (2004) An introduction to biometric recognition. IEEE Trans Circuits Syst Video Technol 14(1):4–20
65. Jian-wei Z, Shui-fa S, Xiao-li L, Bang-jun L (2009) Pitch in speaker recognition. In: International conference on hybrid intelligent systems, pp 33–36
66. Julie S (2000) High-tech surveillance tools and the fourth amendment: reasonable expectations of privacy in the technological age. Am Crim Law Rev 37(1):192–222
67. Kasabov N (1996) Foundations of neural networks, fuzzy systems, and knowledge engineering. The MIT Press, Boston
68. Kieran D, Yan W (2010) A Framework for an event driven video surveillance system. In: IEEE advanced video and signal based surveillance, Boston, pp 97–102
69. Kim P-S, Lee D-G, Lee S-W (2018) Discriminative context learning with gated recurrent unit for group activity recognition. Pattern Recognit 76:149–161
70. Klare B, Paulino AA, Jain AK (2011) Analysis of facial features in identical twins. In: International joint conference on biometrics, pp 1–8
71. Klette R (2014) Concise computer vision. Springer, London
72. Kovac J, Peer P, Solina F (2003) Human skin color clustering for face detection. In: IEEE EUROCON, pp 144–148
73. Kshirsagar VP, Baviskar MR, Gaikwad ME (2011) Face recognition using eigenfaces. In: IEEE computer research and development, pp 302–306
74. Kurakin A, Zhang Z, Liu Z (2012) A real time system for dynamic hand gesture recognition with a depth sensor. In: European signal processing conference, pp 1975–1979
75. Kurniawan F, Shafry M, Rahim M (2012) A review on 2D ear recognition. In: International colloquium on signal processing and its applications, pp 204–209
76. Langheinrich M (2001) Privacy by design - principles of privacy-aware ubiquitous systems. In: UbiComp, pp 273–291
77. Lawrence S, Giles CL, Tsoi AC, Back AD (1997) Face recognition: a convolutional neural-network approach. IEEE Trans Neural Netw 8(1):98–113

78. Le THN, Luu K, Seshadri K, Savvides M (2012) A facial aging approach to identification of identical twins. In: International conference on biometrics: theory, applications and systems, pp 91–98
79. Lee U, Tanaka J (2013) Finger identification and hand gesture recognition techniques for natural user interface. In: Asia Pacific conference on computer human interaction, pp 274–279
80. Li C, Xie C, Zhang B, Chen C, Han J (2018) Deep Fisher discriminant learning for mobile hand gesture recognition. Pattern Recognit 77:276–288
81. Li G, Ito Y, Yu X, Nitta N, Babaguchi N (2009) Recoverable privacy protection for video content distribution. EURASIP J Inf Secur 3(4):2–9
82. Li R, Nguyen M, Yan Q (2017) Morse codes enter using finger gesture recognition. In: IEEE digital image computing: techniques and applications, pp 1–8
83. Li R (2017) Vision-based Morse code recognition. Masters Thesis, Auckland University of Technology, Auckland, New Zealand
84. Li SZ, Lu JW (1999) Face recognition using the nearest feature line method. IEEE Trans Neural Netw 10(2):439–443
85. Li Y, Yuan W, Sang H, Li X (2013) Combination recognition of face and ear based on two-dimensional fisher linear discriminant. In: International conference on software engineering and service science, pp 922–925
86. Liu C, Wechsler H (1999) Comparative assessment of independent component analysis (ICA) for face recognition. In: International conference on audio- and video-based biometrics person authentication, pp 211–216
87. Lu J (2016) Empirical approaches for human behavior analytics. Masters thesis, Auckland University of Technology, New Zealand
88. Mahalingam G, Ricanek K (2013) Investigating the effects of gender and age group based differences in identical twins. In: National conference on computer vision, pattern recognition, image processing and graphics, pp 1–4
89. Maltoni D, Maio D, Jain A, Prabhakar S (2003) Handbook of fingerprint recognition. Springer, New York
90. Martin K, Plataniotis N (2008) Privacy protected surveillance using secure visual object coding. IEEE Trans Circuits Syst Video Technol 18(8):1152–1162
91. Milan P, Jonker W (2007) Security, privacy, and trust in modern data management. Springer, New York
92. Mozaffari S, Behravan H (2011) Twins facial similarity impact on conventional face recognition systems. In: Iranian conference on electrical engineering, pp 1–6
93. Muda L, Begam M, Elamvazuthi I (2010) Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques. J Comput 2(3)
94. Murthy R, Jadon RS (2010) Hand gesture recognition using neural networks. In: International advance computing conference, pp 134–138
95. Nandakumar K, Wan KW, Chan SMA, Ng WZT, Wang JG, Yau WY (2013) A multi-modal gesture recognition system using audio, video, and skeletal joint data. In: ACM international conference on multimodal interaction, pp 475–482
96. Nefian AV, Hayes MH (1999) An embedded HMM-based approach for face detection and recognition. In: International conference on acoustics, speech, and signal processing, pp 3553–3556
97. Newton M, Sweeney L, Malin B (2005) Preserving privacy by de-identifying face images. IEEE Trans Knowl Data Eng 17(2):232–243
98. Nguyen NT, Bui HH, Venkatesh S, West G (2003) Recognizing and monitoring high-level behavior in complex spatial environments. In: IEEE CVPR, pp 1–6
99. Niikura T, Watanabe Y, Komuro T, Ishikawa M (2012) In-air typing interface: realizing 3D operation for mobile devices. In: IEEE global conference on consumer electronics, pp 223–227

100. Nikhil S, Mohan S, Ramya B, Kadambi GR (2010) Design and development of a DSP processor based reconfigurable hand gesture recognition system for real time applications. In: International conference on signal and image processing, pp 39–44
101. Orwell G (2002) Go on, watch me: people are voluntarily surrendering their privacy. The Economist 75(4):1125–1192
102. Osuna E, Freund R, Girosi F (1997) Training support vector machines: an application to face detection. In: IEEE conference on computer vision and pattern recognition, pp 130–136
103. Pankanti S, Hampapur A, Brown L, Tian L, Ekin A, Connell J, Shu F, Lu M (2005) Enabling video privacy through computer vision. IEEE Secur Priv 3(3):50–57
104. Panwar M (2012) Hand gesture recognition based on shape parameters. In: International conference on computing communication and applications, pp 1–6
105. Paruchuri K, Cheung S, Hail W (2009) Video data hiding for managing privacy information in surveillance systems. EURASIP J Inf Secur 8(3):18
106. Pentland A (1991) Face recognition using eigenfaces. In: IEEE CVPR, pp 586–591
107. Pentland A, Moghaddam B, Starner T (1994) View-based and modular eigenspaces for face recognition. In: IEEE conference on computer vision and pattern recognition, pp 84–91
108. Phillips J, Moon H, Rauss J, Rizvi A (2000) The FERET evaluation methodology for face recognition algorithms. IEEE Trans Pattern Anal Mach Intell 22(10):252–274
109. Pollock A (2002) Method of electronic audio surveillance. Law J Libr 5(12):380–385
110. Prabhakar S, Jain A (2002) Decision-level fusion in fingerprint verification. Pattern Recognit 35(4):861874
111. Prabhakar S, Pankanti S, Jain A (2003) Biometric recognition: security and privacy concerns. IEEE Secur Priv 1(2):3342
112. Prasad JS, Saxena A, Javar N, Kaushik KB, Chakraborty P, Nandi GC (2010) Gesture recognition by stereo vision. In: International conference on intelligent interactive technologies and multimedia, pp 155–162
113. Promyarut I, Suvonvorn N, Limsiroratana S (2011) Video scrambling for privacy protection in surveillance system. In: International conference on circuits, system and simulation, pp 177–182
114. Qi YY, Hunt BR (1994) Signature verification using global and grid features. Pattern Recognit 27(12):1621–1629
115. Raheja JL, Shyam R, Kumar U, Prasad PB (2010) Real-time robotic hand control using hand gestures. In: International conference on machine learning and computing, pp 12–16
116. Remagnino P, Baumberg A, Grove T, Hogg D, Tan T, Worral A, Baker K (1997) An integrated traffic and pedestrian model-based vision system. In: BMVC, pp 380–389
117. Saini M, Atrey K, Mehrota S, Kankanhalli S (2012) W^3-privacy: understanding What, When and where inference channels in multi-camera surveillance video. Springer Multimed Tools Appl 68:135–158
118. Sagheer AE, Tsuruta N, Taniguchi R (2004) Arabic lip-reading system: a combination of hypercolumn neural network model with hidden Markov model. In: International conference on artificial intelligence and soft computing, pp 311–316
119. SanMiguel JC, Martinez JM, Garcia A (2009) An ontology for event detection and its application in surveillance video. In: IEEE AVSS, pp 220–225
120. Senior A (2009) Protecting privacy in video surveillance. Springer, London
121. Shailaja D, Gupta P (2006) A simple geometric approach for ear recognition. In: International conference on information technology, pp 164–167
122. Sigal L, Balan O, Black J (2010) Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. IJCV 87(1–2):4–27
123. Socek D, Kalva H, Magliveras S, Marques O, Culibrk D, Furht B (2007) New approaches to encryption and steganography for digital videos. Multimed Syst 13(3):191–204

124. Sohn H, AnzaKu E, Neve W, Ro M, Plataniotis K (2009) Privacy protection in video surveillance systems using scalable video coding. In: IEEE AVSS, pp 424–429
125. Song Y, Demirdjian D, Davis R (2012) Continuous body and hand gesture recognition for natural human-computer interaction. ACM Trans Interact Intell Syst 2(1)
126. Srinivas N, Aggarwal G, Flynn PJ, Vorder Bruegge RW (2012) Analysis of facial marks to distinguish between identical twins. IEEE Trans Inf Forensics Secur 7(5):1536–1550
127. Stergiopoulou E, Papamarkos N (2009) Hand gesture recognition using a neural network shape fitting technique. Eng Appl Artif Intell 22(8):1141–1158
128. Suarez J, Murphy RR (2012) Hand gesture recognition with depth images: a review. In: RO-MAN, pp 411–417
129. Sun Z, Paulino AA, Feng J, Chai Z, Tan T, Jain AK (2010) A study of multi biometric traits of identical twins. In: SPIE biometric technology for human identification, vol 7667. Orlando, USA
130. Suo J, Chen X, Shan S, Gao W, Dai Q (2012) A concatenational graph evolution aging model. IEEE Trans Pattern Anal Mach Intell 34(11):2083–2096
131. Swain M, Dash SK, Dash S, Mohapatra A (2012) An approach for iris plant classification using neural network. Int J Soft Comput 3(1):79
132. Turk MA, Pentland AP (1991) Face recognition using eigenfaces. In: IEEE conference on computer vision and pattern recognition, pp 586–591
133. Valenzise G, Gerosa L, Tagliasacchi M, Antonacci F, Sarti A (2007) Scream and gunshot detection and localization for audio surveillance. In: IEEE AVSS, pp 21–26
134. Vijayan V, Bowyer KW, Flynn PJ, Huang D, Chen L, Hansen M, Kakadiaris IA (2011) Twins 3D face recognition challenge. In: International joint conference on biometrics, pp 1–7
135. Viola P, Jones M (2004) Robust real-time face detection. Int J Comput Vis 57(2):137–154
136. Wang S, Chen L, Zhou Z, Sun X, Dong J (2016) Human fall detection in surveillance video based on PCANet. Multimed Tools Appl 75(19):11603–11613
137. Wei J, Huang DS, Zhang D (2008) Palmprint verification based on robust line orientation code. Pattern Recognit 41(5):1504–1513
138. Wren C, Azarbayejani A, Darrell T, Pentland A (1997) Pfinder: realtime tracking of the human body. IEEE PAMI 19(7):780785
139. Wu J, Cheng J, Zhao C, Lu H (2013) Fusing multi-modal features for gesture recognition. In: ACM on international conference on multimodal interaction, pp 453–460
140. Wu J, Pan G, Zhang D, Qi G, Li S (2009) Gesture recognition with a 3D accelerometer. In: Ubiquitous intelligence and computing, pp 25–38
141. Xu C, Liu Q, Ye M (2017) Age invariant face recognition and retrieval by coupled auto-encoder networks. Neurocomputing 222:62–71
142. Xu D (2006) A neural network approach for hand gesture recognition in virtual reality driving training system of SPG. In: International conference on pattern recognition, vol 3, pp 519–522
143. Yan W, Kankanhalli M (2008) Progressive audio scrambling in compressed domain. IEEE Trans Multimed 10(6):960–968
144. Yan W, Wu X, Liu F, Jain R (2018) Progressive scrambling for social media. IJDCF 10(2):1–13
145. Yuan Q, Gao W, Yao HX (2002) Robust frontal face detection in complex environment. In: IEEE international conference on pattern recognition, pp 25–28
146. Zhang B, Li W, Qing P, Zhang D (2013) Palm-print classification by global features. IEEE Trans Syst Man Cybern Syst 43(2):370–378
147. Zhang D, Shu W (1999) Two novel characteristic in palmprint verification: datum point invariance and line feature matching. Pattern Recognit 32(4):691702
148. Zhang Y (2016) Virtual keyboard based on single finger recognition. Master thesis, Auckland University of Technology, Auckland, New Zealand

149. Zhao W, Chellappa R, Krishnaswamy A, Wen J (1998) Discriminant analysis of principal components for face recognition. In: IEEE international conference on automatic face and gesture recognition, pp 336–341
150. Zhao ZQ, Huang DS, Sun BY (2004) Human face recognition based on multi-features using neural networks committee. Pattern Recognit Lett 25(12):1351–1358
151. Zhang L, Li L, Yang A, Shen Y, Yang M (2017) Towards contactless palmprint recognition: a novel device, a new benchmark, and a collaborative representation based identification approach. Pattern Recognit 69:199–212
152. Zhang T, Feng Z (2013) Dynamic gesture recognition based on fusing frame images. In: International conference on intelligent systems design and engineering applications, pp 280–283
153. Zhu C, Peng Y (2017) Discriminative latent semantic feature learning for pedestrian detection. Neurocomputing 238:126–138

Visual Event Computing I 6

6.1 Definition of an Event

An event is something that happens at a given place and time [27]. This definition has been widely accepted by international communities. An event bridges the gap between cyberspace and real world. The basic components of an event are the famous "5W", namely, who, when, where, what and why. These components have been successfully used in computer vision and artificial intelligence especially in the computations of observation, learning, presentation, and inference.

Event entity refers to event ID, time or duration, location, description, etc. One event is unique; a UUID is assigned for a specific event for the purpose of identification. For an event, the entities time and location only belong to this event; therefore, it is impossible to find two different events with the same time and location, or one event having different locations and time. This is the identical attribute of an event [27].

In intelligent computation, an event usually has twofold: detection and exploration [12,13]. Event detection is based on the pattern of long-time observations. We have a wealth of events happening every day; for example, traffic control at a junction uses green, yellow, and red color traffic lights which always are being switched in the sequence of shifting. Without doubts, after a red light, the next will be yellow, then red, and so on. The story is shown in Fig. 6.1 Once we realize this occurrence, we find the "5W" and record the story as normal or abnormal event [16,22]. If there is an accident captured at this intersection, that means we detect an abnormal event. Event exploration refers to get new events from the known ones. For example, Alice(A) and Bob(B) know how to get Cindy(C)'s home from their owns, and later they could infer the routines how to get Alice's home or Bob's home from current location using the existing events.

Among the events, a telic event is different from atelic one. Telic event is the one that has end point of its time interval; however, atelic events do not have the end point.

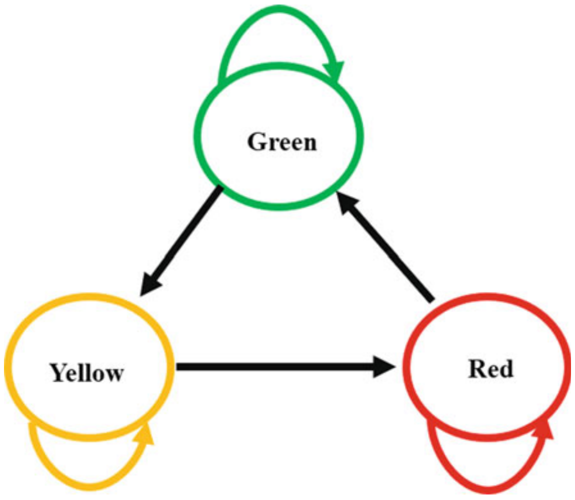


Fig. 6.1 The transitions of traffic lights are thought as events

Meanwhile, atomic events are different from composite one. An atomic event is the elementary one which is not allowed to be divided into further sub-event; it is the simplest and fundamental one. A composite event is defined by using composition of two or more atomic events which is a complex event [24].

Event computing has three very important aspects: operating, storing, and modeling. Event operations refer to unary operations and binary operations. Unary operations have only one operating object, such as projection, selection, renaming, etc., while binary operations have two objects such as union, concatenation, conditional sequence, iteration, aggregation, etc. [27].

Event projection maps one event onto another like vectors in linear algebra. The projection operation is implemented by using inner product which indicates how one event impacts on the other. Apparently two orthogonal events have not too much relationship mutually; however, two vectors conforming to the identical direction take the maximum influence on each other. Mathematically, cosine function $y = \cos(x), x \in (-\infty, +\infty)$ takes effect as it is used for inner product of two vectors $\mathbf{V_1}$ and $\mathbf{V_2}$. $\mathbf{V_1} \cdot \mathbf{V_2} = \|\mathbf{V_1}\| \cdot \|\mathbf{V_2}\| \cos(\alpha), \alpha \in [0, \pi]$.

We use event databases and MPEG-7 to store visual events. In the event databases, we define the fields using the attributes of an event, namely, where, who, when, what and why; the records are used to store each one, the associated information such as metadata, and sensor data are deposited in the relative tables [3,5]. Event search and retrieval are typical operations based on the well-defined SQL language. In SQL, the syntax "$SELECT < records > FROM < database > WHERE < boolean\ \ conditions >$" structure is applied to event search or retrieval from a database.

Boolean conditions satisfy logic laws which are indicated as follows: $\forall A, B \in \{T, F\}$, 'T' means the boolean true, while 'F' refers to the false:

- Implication law: $A \to B \equiv \neg A \vee B$.
- Idempotent law: $A \wedge A \equiv A$.
- Commutative law: $A \wedge B \equiv B \wedge A$.
- Associative law: $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$.
- Absorption law: $A \wedge (A \vee B) \equiv A$.
- Distributive law: $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$.
- Double negation law: $\neg\neg A \equiv A$.
- De Morgan's law: $\neg(A \wedge B) \equiv \neg A \vee \neg B$.
- Excluded middle law: $A \wedge \neg A \equiv F$.
- Identity law: $A \wedge T \equiv A$.
- Domination law: $A \wedge F \equiv F$.

Video Event Representation Language (VERL) is used for event description where it harasses the structure such as sequence, iteration, alternation, and conditional branches to control the program like a normal programming language; among the events, temporal relationships have the types like "before," "begin," "inside," "end," "contain," "during," etc., surveillance events could be presented by using semantic languages in visual, graphical, spatial, or ontological way [8,18].

The purpose of event presentation is to convey main points to audience; event presentation indeed has computing aspects and needs skills [28]. Good event presentations attract audience by using natural languages in the way of multimedia [24,26] such as text, video and audio, diagrams, figures, charts, animation and graphics, etc. [4], a good example is the interface of YouTube or the Gmail which links all relevant content together with ranking. A very special language for event presentations in writing and oral is mathematics which is based on symbols and functions in logic; the mathematical symbols include $\forall$, $\exists$, $\neg$, $|$, $\wedge$, and the algebra system $< \mathbf{0}, \mathbf{1}, +, -, \times, \div >$, etc., these symbols have been adopted and fixed for mathematical presentations such as the modern concepts: group (e.g., Z^+), ring (e.g., polynomial ring) and fields (e.g., real number field or field of reals) which could not be replaced by others.

An event has six aspects: temporal, spatial, causal, experiential, informational, and structural. Temporal and spatial [14,23] denoted as spatial-temporal are well-known fundamental aspects of an event, but triggered by various reasons, namely, consequence is caused by its reasons. "Event driven" means one or multiple events are automatically triggered by conditions derived from other events, and a load of events could be triggered in sequential order by casual information.

6.2 Event Database

Surveillance is real reality which is experiential with human portfolio. Nowadays our daily life is being monitored by surveillance cameras anywhere and anytime; therefore, like the example of traffic lights, we find patterns after long-term observations, we call the patterns as events which are discriminative and covering. Now we save the events in a database; we call this event database as eBase which is easy to be implemented by using programming languages such as PHP, MySQL, Ruby, Python, etc.

In event archiving, we also deposit all event sensors into the eBase, each sensor has its attributes. The sensors connect with servers using IP address and GPS information while capturing data like image, video, and audio; each video, or audio footage has its metadata.

Event detector is a software which is linked to the eBase. The detectors have the ability to link a semantic object and the stories of this object are described by using images, videos, or audio clips captured from surveillance cameras or other sensors.

Users of a surveillance system are allowed to provide comments on an event that are stored into the eBase as evidences of witnesses. In surveillance, witnesses and their comments are regarded as the solidate evidences.

During eBase creation, the database is protected by a password. The eBase security and privacy are generally guaranteed by the system and password. Without a password, the eBase could not be accessed; correspondingly access control is limited. In recent years, data security and privacy of databases have been delved from other aspects [1].

When we log into the eBase, the records are stored with the fields of event attributes. Programming languages such as MySQL, PHP, Python, and even Matlab which could call and run SQL language having the ability to automatically write an event into the eBase as a record. Event database (eBase) is able to be linked to a web page through a web server. Any updates to a database will be shown on the web page timely.

6.3 Event Detection and Recognition

Event is a semantic story of an object; therefore, it has duration related to time series. Thus in event detection and recognition, we usually use the methods having temporal and spatial relationships to detect and recognize events [11,12,14,17,23].

Event recognition can be split into twofold, which encompasses model-based approaches and appearance-based approaches [24]. In the first, Bayesian networks typically have been used to recognize simple events or static postures from video frames; meanwhile, hidden Markov model (HMM) [21] also has been applied to human behavior recognition [7,9,10]. Appearance-based approaches are based on salient regions of local variations in both spatial and temporal dimensions [11,14,17]. Boosting is adopted to learn for a cascade of filters for visual event detection [26]. In addition, grammar-based and statistics-based methods could also be categorized by using dimension of sampling support, characteristics and mathematical modeling.

Bayesian network is a graphical model for representing conditional independencies between a set of random variables. Dynamic Bayesian Network (DBN) is based on Bayes' theorem to represent sequences of variables. For example, when we see the scene of grass wet, we may infer the reasons from the given conditions such as

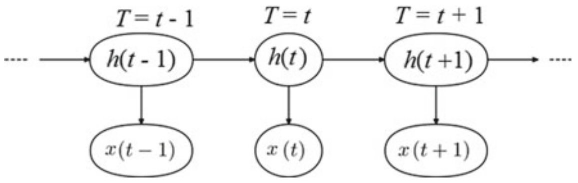


Fig. 6.2 HMM model

sprinkler or raining. Meanwhile, for raining there are a handful of reasons; for sprinkler there are other possibilities as well. One primary reason may trigger happenings of others; hence, these reasons may form a directed graphical probabilistic network, and we call it as Bayesian network. Mathematically, a Bayesian network is shown as Eq. (6.1),

$$p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{N} p(x_i | x_{\pi_i}) \tag{6.1}$$

where π_i is the parent of i in the graph.

Kalman filter is usually implemented for filtering using continuous state linear dynamic systems. Most of time, Kalman filtering is used as a filter in information processing; like other filters, those noises will be filtered out by using Kalman filter, and only the main part of the processing information will be left. The mathematical description of Kalman filtering could be found from Sect. 4.4 related to object tracking [20].

A hidden Markov model (HMM) is a statistical approach that the system being modeled is premised to be a Markov process with unobserved (hidden) states. A HMM is presented as the simplified dynamic Bayesian network [6]. As shown in Fig. 6.2, in order to infer state $x(t+1)$ at the time $t+1$, we use the states $h(t+1)$ and $h(t)$ at t, respectively. An HMM was designed for predicting discrete state sequences.

In the simplest Markov Model, like a Markov chain [21], the state S is directly visible to the observer and therefore the state transition probabilities are the only parameters. In a HMM, the state is not directly visible, but output O, dependent on the states, is perceivable. Each state has a probability distribution over the possible output. Therefore, the sequence generated by HMM reflects the gradual changes of the state sequence. Note that the word "hidden" refers to the state sequence through which the model passes.

Given a HMM model $\lambda = (A, B, \pi)$, the parameters refer to:

- State: $\mathbf{S} = \{S_1, S_2, \ldots, S_N\}$
- Observation: $\mathbf{V} = \{v_1, v_2 \ldots, v_M\}$
- Transition matrix: $\mathbf{A} = (a_{ij})_{N \times N}, a_{ij} \equiv p(q_{t+1} = S_j | h_t = S_i)$
- Emission probabilities: $\mathbf{B} = (b_j(m))_M, b_j(m) \equiv p(O_t = v_m | h_t = S_j)$
- Initial probabilities: $\Pi = (\pi_i)_N, \pi_i \equiv p(h_1 = S_i)$
- Output: $\mathbf{O} = \{O_1 O_2 \ldots O_T\}$
- Latent variables: $\mathbf{H} = \{H_1 H_2 \ldots H_T\}$

Given a HMM model $\lambda = (A, B, \pi)$ [2], the Forward-Backward procedure answered the question: which output of HMM $p(O|\lambda)$ is the best one given λ? Viterbi algorithm tells us which path $H^* = \underset{\mathbf{H}}{\arg\max}\, p(\mathbf{H}|\mathbf{O}, \lambda)$ is the the best one in the unfolded lattice of HMM, whilst Baum–Welch algorithm is used to seek $\lambda^* = \underset{\lambda}{\arg\max}\, p(\chi|\lambda)$.

Therefore, the Baum–Welch algorithm is often used to estimate the parameters of HMMs, Wikipedia provides an example for estimating the initial probability, transition and emission matrices using Python, see: https://en.wikipedia.org/wiki/Baum-Welch_algorithm.

The Viterbi's algorithm can be visualized by a trellis diagram. The Viterbi path is essentially the shortest path through this trellis. Wikipedia provides an example for what is the most likely sequence of health conditions of the patient after several days' observations, see: https://en.wikipedia.org/wiki/Viterbi_algorithm.

Conditional random fields (CRFs) are undirected probabilistic models designed for segmenting and labeling sequence data [15, 19]. When we use CRFs for event detection and recognition, we deploy the tasks of those layers; after training and testing, we obtain the parameters and the semantic events from detection and recognition.

In Markov random field (MRF), the label set f is said to be a CRF, given d (observation),

$$p(f_i|d, f_{\mathscr{S}-\{i\}}) = p(f_i|d, f_{\mathscr{N}_i}) \tag{6.2}$$

$$P(f|d) = \frac{1}{Z}\exp\left\{-\sum_{i\in\mathscr{S}} V_1(f_i|d) - \sum_{i\in\mathscr{S}}\sum_{i'\in\mathscr{N}_i} V_2(f_i, f_{i'}|d)\right\}$$

where $V_1(\cdot)$ and $V_2(\cdot)$ are called association and interaction potentials, respectively.

Deep Markov random field (DMRF) [25] could be applied to texture synthesis, image superresolution, etc. In DMRF, the hidden state h_u and the pixel x_u together form an MRF; each hidden state h_u connects to the neighboring states h_v, the neighboring pixels x_v, and the pixel at the same location. The dependencies are reflected in the function $\zeta(x_u, h_u)$, $\phi(h_u, h_v)$ and $\psi(h_u, x_v)$. A MRF is therefore expressed as

$$p(\mathbf{x}, \mathbf{h}) = \frac{1}{Z}\prod_{u\in V} p(x_u, h_u) \prod_{(u,v)\in E} (\phi(h_u, h_v)\psi(h_u, x_v)\psi(h_v, x_u)) \prod_{u\in V} \lambda(h_u) \tag{6.3}$$

where V and E are the sets of vertices and edges of the MRF field, respectively. Z is the partition function as well as $\lambda(h_u)$ is the regularization function.

$\zeta(x_u, h_u)$ reflects how the pixel values are generated from the hidden states which is subject to Gaussian Mixture Model. $\phi(h_u, h_v)$ and $\psi(h_u, x_v)$ are fully connected.

Table 6.1 An example of Bayesian theorem

Event	Description	Probability
A	Someone has disease "C"	0.02
B	Someone has disease "H"	0.10
$B\|A$	Someone has disease "H" given disease "C"	0.80
$A\|B$	Someone has disease "C" given disease "H"	0.16

According to the Expectation-Maximization algorithm, the posterior distribution of $\mathbf{h}_i$ (in E-steps) and optimized parameter θ (in M-steps) could be iteratively calculated.

$$\hat{\theta} = \arg\max_{\theta} \frac{1}{n} \sum^{n} E_{p(x_i|h_i,\theta)} \cdot \log p(x_i, h_i|\theta) \tag{6.4}$$

6.4 Event Classification

For event classifiers, the basic one is Bayesian classifier which is based on the famous Bayes' theorem shown as Eqs. (6.5) and (6.6) while Bayes' theorem remarkable as the basestone of modern pattern classification:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{p(B)} \tag{6.5}$$

$$P(B|A) = \frac{P(A|B) \cdot P(B)}{p(A)} \tag{6.6}$$

The Eqs. (6.5) and (6.6) could be explained as a prior modified by using likelihood compared to evidence so as to get the posterior.

$$Posterior = \frac{Prior \times likelihood}{evidence} \tag{6.7}$$

Fundamentally, we have posterior probability based on the prior, likelihood and evidence shown as as Eq. (6.7). For example in Table 6.1,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{0.80 \times 0.02}{0.10} = 0.16$$

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} = \frac{0.16 \times 0.10}{0.02} = 0.80$$

An application of Bayes' theorem is Naive Bayesian Classifier for classifying documents, it has been applied to spam email filtering:

$$p(D|S) = \prod_i p(w_i|S) \tag{6.8}$$

$$p(D|\neg S) = \prod_i p(w_i|\neg S) \tag{6.9}$$

where D is a document for classification, S represents a spam document, and w_i is the word included in a document. Applying Bayes' theorem to this application, the possibility of a document is classified into spam or not is given by:

$$p(S|D) = \frac{p(D|S) \cdot p(S)}{p(D)} = \frac{p(S)}{p(D)} \prod_i p(w_i|S) \tag{6.10}$$

$$p(\neg S|D) = \frac{p(D|\neg S) \cdot p(\neg S)}{p(D)} = \frac{p(\neg S)}{p(D)} \prod_i p(w_i|\neg S) \tag{6.11}$$

Now we combine Eqs. (6.10) and (6.11) together,

$$\frac{p(S|D)}{p(\neg S|D)} = \frac{p(S) \prod_i p(w_i|S)}{p(\neg S) \prod_i p(w_i|\neg S)} \tag{6.12}$$

For the purpose of simplifying the calculation, we have,

$$\ln \frac{p(S|D)}{p(\neg S|D)} = \ln \frac{p(S)}{p(\neg S)} + \sum_i \ln \frac{p(w_i|S)}{p(w_i|\neg S)} \tag{6.13}$$

Hence, if $\ln \frac{p(S|D)}{p(\neg S|D)} > 0$, i.e., $p(S|D) > p(\neg S|D)$, the document is not spam, or else it is a spam one.

6.5 Event Clustering

Clustering is regarded as the procedure: given a set of data points, the data are grouped into several clusters so that within one cluster, the data points are more similar to another; data points in another cluster are less similar. Usually a distance between two vectors known as similarity measurements is used for clustering; the typical measure $d = |\mathbf{X} - \mathbf{Y}|$ between two vectors $\mathbf{X} = (x_1, \ldots, x_n)$ and $\mathbf{Y} = (y_1, \ldots, y_n)$ includes:

- Euclidean distance,

$$d = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

- Minkovski distance,

$$d = \left(\sum_{i=1}^{N} (x_i - y_i)^p \right)^{\frac{1}{p}}$$

- Chebyshev distance,

$$d = \lim_{p \to \infty} \left(\sum_{i=1}^{N} (x_i - y_i)^p \right)^{\frac{1}{p}} = \max |p_i - q_i|$$

- Manhattan distance,

$$d = \lim_{p \to 0} \left(\sum_{i=1}^{N} (x_i - y_i)^p \right)^{\frac{1}{p}} = \sum_{i=1}^{N} |p_i - q_i|$$

- Mahalanobis distance,

$$d_i = (x_i - y_i) \cdot \Sigma^{-1} \cdot (x_i - y_i)^T$$

where $(x_i, y_i), i = 1, 2, \ldots, N$ are the data points. In clustering, two typical clustering approaches are,

- Partitional clustering: a division groups data objects into nonoverlapping subsets so that each data object is in exactly one subset.
- Hierarchical clustering: a set of nested clusters are organized as a hierarchical tree.

k-means clustering is a kind of partitional clustering approaches. In k-means clustering, (1) each cluster is associated with a centroid (center point); (2) each point is assigned to the cluster with the closest centroid; (3) the number of clusters K must be specified. The algorithm of k-means is shown as the following algorithm (3).

6.6 Questions

Question 1. What is an event? What are the components of an event?

```
Input   : Data points;
Output: Clusters;
Initialization;
Select the number of clusters $K$ as the initial centroid;
while Centroid changes existing do
    (1) Form $K$ clusters by assigning all points to the closest centroid;
    (2) Recompute each centroid of each cluster;
    (3) Find the difference of centroid of each cluster;
end
```

Algorithm 3: K-means clustering algorithm

Question 2. What is event computing? What are the event operations?

Question 3. How to store visual events in MPEG-7?

Question 4. What is the k-means clustering?

References

1. Adams A, Ferryman M (2012) The future of video analytics for surveillance and its ethical implications. Secur J 28(3):272–289
2. Alpaydin E (2010) Introduction to machine learning, 2nd edn. The MIT Press, Boston
3. Argano M, Gidwani T, Yan W, Issa F (2012) A comprehensive survey of event analytics. Int J Digit Crime Forensics 4(3):33–46
4. Calvel C, Ehrette T, Richard G (2005) Event detection for audio-based surveillance system. In: IEEE ICME, pp 1306–1309
5. Chen D, Tao Z, Ma G (2008) Application of wireless sensor networks for monitoring emergency events. In: IEEE conference on sensors, pp 518–521
6. Gong S, Xiang T (2003) Recognition of group activities using dynamic probabilistic networks. In: IEEE ICCV, pp 742–750
7. Haritaoglu I, Harwood D, Davis LS (2000) W^4: real-time surveillance of people and their activities. IEEE TPAMI 22(8):809–830
8. Huang M, Liu Z (2012) Layered event ontology model. In: International conference on fuzzy systems and knowledge discovery, pp 948–951
9. Iosifidis A, Tefas A (2012) View-invariant action recognition based on artificial neural networks. IEEE Trans Neural Netw Learn Syst 23(3):412–424
10. Ivanov Y, Bobick A (2000) Recognition of visual activities and interaction by stochastic parsing. IEEE PAMI 22(8):852–872
11. Ke Y, Sukthankar R, Hebert M (2005) Efficient visual event detection using volumetric features. In: IEEE ICCV, pp 166–173
12. Maryam K, Reza K (2012) An analytical framework for event mining in video data. Artif Intell Rev 41(3):401–413
13. Naphade M, Huang T (2002) Discovering recurrent events in video using unsupervised methods. In: International conference on image processing, vol 2, p 13

14. Niebles J, Wang H, Li F (2008) Unsupervised learning of human action categories using spatial-temporal words. Int J Comput Vis 79(3):299–318
15. Oh S, Hoogs A, perera A, Cuntoor N, Chen C, Lee JT et al (2011) A large-scale benchmark dataset for event recognition in surveillance video. In: IEEE conference on computer vision and pattern recognition, pp 3153–3160
16. Popoola O, Wang K (2012) Video-based abnormal human behavior recognition: a review. IEEE Trans Syst Man Cybern Part C Appl Rev 42(6):865–878
17. Rui Y, Anandan P (2000) Segmenting visual actions based on spatio-temporal motion patterns. In: IEEE Conference on computer vision and pattern recognition, pp 111–118
18. SanMiguel JC, Martinez JM, Garcia A (2009) An ontology for event detection and its application in surveillance video. In: IEEE AVSS, pp 220–225
19. Sigal L, Balan A, Black M (2010) Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. IJCV 87(1–2):4–27
20. Stauffer C, Eric W, Grimson L (2000) Learning patterns of activity using real-time tracking. IEEE PAMI 22(8):747–757
21. Tran S, Davis L (2008) Event modeling and recognition using markov logic networks. In: ECCV, pp 610–623
22. Tziakos I, Cavallaro A, Xu L (2010) Event monitoring via local motion abnormality detection in non-linear subspace. Neurocomputing 73(10):1881–1891
23. Velipasalar S, Brown L, Hampapur A (2006) Specifying, interpreting and detecting high-level, spatio-temporal composite events in single and multi-camera systems. In: IEEE computer society conference on computer vision and pattern recognition workshop, p 110
24. Westermann U, Jain R (2007) Toward a common event model for multimedia applications. IEEE MultiMed 14(1):19–29
25. Wu Z, Lin D, Tang X (2006) Deep Markov random field for image modelling. In ECCV 2016, pp 295–312
26. Xie L, Sundaram H, Campbell M (2008) Event mining in multimedia streams. Proc IEEE 96(4):623–647
27. Yan W, Kieran D, Rafatirad S, Jain R (2011) A comprehensive study of visual event computing. Springer Multimed Tools Appl 55(3):443–481
28. Zelnik-Manor L, Irani M (2001) Event-based analysis of video. In: IEEE CVPR, vol 2, pp 123–130

7 Visual Event Computing II

7.1 Event Search and Retrieval

After events are stored in the event database (eBase), we could create an index based on these records. The index plays a dominant role and saves the time of information retrieval [1,2]. This has simplified and hastened the relevant operations and processing, especially those over the Internet.

7.1.1 Lexical Search and Retrieval

On the site of distributed Web servers, the index for search and retrieval needs *tokenization*, namely, segmenting the input text into words after dropping stop words such as articles (‘a,’ “an,” and “the”), adjectives (e.g., “red,” “blue,” “yellow,” “cold,” “hot,” etc.), prepositions (e.g., “of,” “when,” “where,” etc.), and pronouns (e.g., ‘I,’ “you,” “she,” “he,”“her,” etc.) [1].

In linguistics, stop words are the ones which are filtered out before or after natural language processing (in text). Stop words usually have very highly statistical occurrences with very low functionalities, such as words: “the,” “is,” “at,” “which,” and so on. Stop words cause issues which may affect retrieval when searching for phrases that include them, particularly in names such as “the who,” or “take that.” Some search engines remove the most common words including lexical ones such as “want” from a query in order to improve performance.

The general strategy for determining a stop list is to sort the terms by collecting the word frequency which is the total number of times each term appears in the document collection and then to take out the most frequent terms for their semantic content related to the documents being indexed. As a stop list, the members of this list are discarded during indexing. An example of a stop-word list is ‘a,’ “an,” and,

W. Q. Yan, *Introduction to Intelligent Surveillance*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-10713-0_7

"the." Using the list significantly reduces the number of postings that a system has to deposit.

In linguistic computations, the *textual normalization* process means to capitalize or make words in low case; after processed, the words are in uniform. For an example, the word "student" will be uppercased as "Student" and so on so forth [1].

The operation *stemming* means to find the roots of words such "fishing," "fisher," "fished," removing the useless strings that only uses the word "fish" for indexing; stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly, and often includes the removal of derivational affixes.

In natural language processing (NLP), *lemmatization* is to get the semantic roots of each word such as "good," "better," "best." Lemmatization usually refers to work properly with the use of a vocabulary and morphological analysis of words, normally aims to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma [11]. Linguistic process for stemming or lemmatization is often tackled by an additional plug-in component to the indexing process, and a number of such components exist in open source.

In computer science, an *inverted index* (also referred to as posting file or inverted file) is an index data structure storing a mapping from content, such as words or numbers, to its locations in a database or in an XML file, or a set of documents. The purpose of an inverted index is to allow fast full text search, at a cost of increased processing when a document is added into the database. The inverted file may be the database itself, rather than its index which is the most popular data structure in document retrieval systems. Inverted index uses the position of each word in the sentences of an article to find a sentence in context. Given a keyword sequence, the positions of relevant words will be found at the positions of intersections of all words. Intersecting the posting list leads to the searching results [11].

For example, suppose we have the sentence 0, "what is it," sentence 1, "it is a banana," and sentence 2, "it is," then we have the posting list as "a" $=\{1\}$, "banana" $=\{1\}$, "is" $=\{0, 1, 2\}$, "it" $=\{0, 1, 2\}$, "what" $=\{0\}$; the sentence "what is it?" could be found from the position intersecting $\{0\} \cap \{0, 1, 2\} \cap \{0, 1, 2\} = \{0\}$.

There are two main variants of inverted indices. A record-level inverted index (or inverted file index or just inverted file) contains a list of references to documents for each word. A word-level inverted index (or fully inverted index or inverted list) additionally contains the positions of each word within a document. The latter form offers more functionality (like phrase searches) but needs more time and space to be processed.

Data structure of inverted index is a central component of a typical search engine indexing algorithm. The goal of a search engine is to optimize the speed of query: finding the documents where a word occurs. Once a forward index is developed, which stores the list of words per document, it is next to develop an inverted index. Querying the forward index would require sequential iteration through each document and each word to verify a matching document. The time, memory, and processing resources to perform such a query are not always technically realistic. In-

Table 7.1 An example of $tf - idf$ with tf, df and idf of the terms

Terms	tf (doc1)	tf (doc2)	tf (doc3)	$\cdots\cdots$	df	idf
"Algorithm"	4,250	3,400	5,100	$\cdots\cdots$	850	0.07
"The"	50,000	43,000	55,000	$\cdots\cdots$	1,000	0.00
"Book"	7,600	4,000	2,000	$\cdots\cdots$	400	0.40
"Surveillance"	600	0	25	$\cdots\cdots$	25	1.6

Table 7.2 An example of $tf - idf$ with the scores and ranks of terms

Terms	tf_idf (doc1)	tf_idf (doc2)	tf_idf (doc3)	$\cdots\cdots\cdots$
'Algorithm'	299.97	239.98	359.96	$\cdots\cdots$
"The"	0.00	0.00	0.00	$\cdots\cdots$
"Book"	3,024.34	1,591.76	795.88	$\cdots\cdots$
"Surveillance"	961.24	0.00	40.05	$\cdots\cdots$
Scores	4285.57	1831.74	1195.89	$\cdots\cdots$
Ranking	1	2	3	$\cdots\cdots$

stead of listing the words per document in the forward index, the structure of inverted index is developed which lists the documents per word.

For keyword-based search, we also use the score of inverse document frequency based on index and Eqs. (7.1) and (7.2) for ranking, where tf is term frequency, idf refers to inverse document frequency, df represents document frequency, N is document number, d is the documents, and q is the key-word based query [11].

$$idf = \ln\left(\frac{N}{df + 1}\right) \tag{7.1}$$

$$tf\_idf = tf \times idf \tag{7.2}$$

The score for a query q is calculated as,

$$S(q, d) = \sum (tf\_idf) \tag{7.3}$$

The queried documents will be listed according to the sum of the scores in the ranking. An example of four terms in three documents of a fictional collection $N = 1,000$ is shown in Tables 7.1 and 7.2.

7.1.2 Global and Local Search

In computational intelligence, local and global search includes breadth-first search, depth-first search, depth-limited search and bidirectional search, etc. [5,14].

7.1.2.1 Breadth-First Search (BFS)

Breadth-first search goes through the tree by levels, traverses all of the nodes on the top level first, then on the second level, and so on. This strategy has the benefit of being completed and optimal as long as the shallowest solution is the best one. However, the way that the breadth-first search is achieved by keeping all of the leaf nodes in memory, which requires a prohibitive amount of memory when searching for any nodes more than a very small tree.

Mathematically, let $G = (V, E)$ be a graph with n vertices $|V| = n$; $N(v)$ is the set of neighbors of v. Let $\sigma = (v_1, \dots, v_n)$ be an enumeration of the vertices of V. The enumeration σ is said to be a BFS ordering if, for all $1 < i \leq n$, v_i is the vertex $w \in V \setminus \{v_1, \dots, v_i - 1\}$ such that $\nu_{(v_1,\dots,v_{i-1})}(w)$ is the minimal. BFS therefore is iterative.

Breadth-first search is useful when

- The space is not a problem;
- Finding the solution contains the fewest arc;
- A few solutions may exist, and at least one has a short path length;
- Infinite paths may exist, because it explores all of the search space, even with the paths.

7.1.2.2 Depth-First Search (DFS)

Depth-first search goes through the tree by branches, along all the ways down to the leaf nodes at bottom of the tree before trying the next branch over. This strategy requires much less memory than breadth-first search, since it only needs to store a single path from the root of the tree down to the leaf node. However, it is potentially incomplete, since it will keep going down one branch until it finds an end, and it is non-optimal, if there is a solution at the fourth level in the first branch tried and a solution at the second level in the next one, the solution at the fourth level will be returned.

In depth-first search, the frontier acts like a last-in first-out stack. The elements are added to the stack one at a time. The one selected and taken off the frontier at any time is the last element that was added. Implementing the frontier as a stack results in paths being pursued in a depth-first manner - searching one path to its completion before trying an alternative path.

Because depth-first search is sensitive to the order in which the neighbors are added to the frontier, caution must be taken sensibly. This ordering can be fulfilled statically and dynamically where the ordering of the neighbors depends on the goal.

Again, let $G = (V, E)$ be a graph with n vertices; $\sigma = (v_1, \dots, v_n)$ is an enumeration of the vertices of V. The enumeration σ is said to be a DFS ordering if, for all $1 < i \leq n$, v_i is the vertex $w \in V \setminus \{v_1, \dots, v_i - 1\}$ such that $\nu_{(v_1,\dots,v_{i-1})}(w)$ is the maximal.

Depth-first search is appropriate when,

- The space is restricted;

- Many solutions exist, particularly for the case where nearly all paths lead to a solution;
- The order of neighbors of a node is added to the stack so that solutions are found on the first try.

Depth-first search is the basis for a number of other algorithms, such as iterative deepening. This algorithm does not specify the order in which the neighbors are added to the stack that represents the frontier. The efficiency of the algorithm is sensitive to the ordering.

Depth-limited search essentially conducts a depth-first search with a cutoff at a specified depth limit. When the search hits a node at that depth, it stops going down the branch and moves over to the next one. This avoids the potential issues with depth-first search of going down one branch indefinitely. However, depth-limited search is incomplete - if there is a solution only at a level deeper than the limit, it will not be found.

Iterative deepening search commits repeated depth-limited searches starting with a limit of zero and incrementing once each time. As a result it has the space-saving benefits of depth-first search but is also complete and optimal because it will visit all the nodes on the same level first before continuing to the next level in the next round when the depth is incremented.

7.1.2.3 Informed Search

The objective of a heuristic is to produce a solution in a reasonable time frame that is good enough for solving the problem at hand. This solution may not be the best of all the actual solutions to this problem, or it may simply approximate the exact solution. But it is still valuable because finding it does not require much time.

Informed search greatly reduces the amount of time by making intelligent choices for the nodes that are selected for expansion. This implies there exist optional ways of evaluating the likelihood of a given node which is on the solution path.

7.1.2.4 Best-First Search

Suppose that one has an evaluation function $h(n)$ defined at each node n that estimates the cost of reaching the goal from this node. A search that chooses the node on the agenda for which this function is the minimum is called a greedy search. Generally, its performance is not better than the breadth-first search.

The algorithm that chooses the node on the agenda for which function is the minimal is called A* search. It is considered important because heuristic search means uncertainty. The best-first search is to find the best and stop while A* search is to find the best one and then delete the branch, the memory-bounded greedy search is to keep finding the best one till it could not find another one.

The A* algorithm combines features of uniform cost search and pure heuristic search to efficiently compute optimal solutions. A* algorithm is a best-first search algorithm in which the cost associated with a node is $f(n) = g(n) + h(n)$, where

$g(n)$ is the cost of the path from the initial state to node n and $h(n)$ is the heuristic estimate, the cost or a path from node n to a goal. Thus, $f(n)$ estimates the lowest total cost of any solution path going through node n. At each point, a node with lowest f value is chosen for expansion. Ties among nodes of equal f value should be broken in favor of nodes with lower h values. The algorithm terminates when a goal is chosen for expansion.

For the search problem like puzzles, A* algorithm can find optimal solutions to this type of problems. In addition, A* algorithm makes the most efficient use of the given heuristic function in the following sense: among all shortest path algorithms using the given heuristic function $h(n)$, A* algorithm expands the fewest number of nodes.

7.1.2.5 Hill-Climbing Search

Local search is a heuristic method for solving computationally hard optimization problems. Local search is used on solving problems that are formulated as finding a solution maximizing a criterion among a number of candidate solutions. Local search algorithms move from solution to solution in the search space by applying local changes until a solution deemed optimal solution is found or a time bound is elapsed.

Hill climbing search, as we climb a mountain to find the best one, has a rest then tries in different ways simulated annealing search taken the surroundings into consideration. The core of this algorithm is a mathematical optimization which belongs to the family of local search. The iterative algorithm starts with an arbitrary solution and then finds a better solution by incrementally changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.

Mathematically, hill climbing search maximizes (or minimizes) a target function $f(\mathbf{x})$, where $\mathbf{x}$ is a vector of continuous or discrete values. If $y = f(\mathbf{x})$ is a surface, it may have various scenarios such as one maximum, multiple local maxima, or a ridge. Hill-climbing search algorithms include simple hill-climbing search, stochastic hill-climbing search and random-restart hill climbing.

Hill climbing algorithm is good for finding a local optimum but it is not guaranteed to find the best possible solution, namely, the global optimum out of all possible solutions within the search space. Local optima are guaranteed by using restarts only, i.e., repeated local search, or more complex schemes based on iterations like iterated local search, on memory like reactive search optimization, or memory-less stochastic modifications like simulated annealing.

7.1.2.6 Genetic Search

Genetic algorithm (GA) searches surroundings, parental and sibling levels, finds the best way to grow. GA algorithm is a search heuristic that mimics the process of natural selection. This heuristic is routinely used to generate useful solutions to optimize search problems.

GA algorithm belongs to the larger class of evolutionary algorithms (EA) which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover [8,9]. The steps for a GA algorithm are:

Step 1. *Initialization*: Create an initial population; e.g., we assume four strings (S_i, $i = 1, 2, \cdots, N$, $N = 4$) consisting '0' and '1' have been created like the DNA in a gene: $S_1 = (01101)_2 = (13)_{10}$, $S_2 = (11000)_2 = (24)_{10}$, $S_3 = (01000)_2 = (8)_{10}$, $S_4 = (10011)_2 = (19)_{10}$, where subscript '2' refers to a binary number while "10" for decimal.
Step 2. *Evaluation*: Evaluate each member of the population, and calculate a "fitness" for the individual; e.g., we assume the fitness function is $f(S) = S^2$, therefore $f(S_i) = S_i^2$, $i = 1, 2, \cdots, N$. Hence $S_1^2 = 169$, $S_2^2 = 576$, $S_3^2 = 64$ and $S_4^2 = 361$.
Step 3. *Selection*: Constantly improve populations' overall fitness. The probabilities of the given strings are $p(S_i) = \frac{f(S_i)}{\Sigma_{i=1}^{N=4} f(S_i)}$; thus, $p(S_1) = 0.14$, $p(S_2) = 0.49$, $p(S_3) = 0.06$ and $p(S_4) = 0.31$.

We therefore create four intervals to be selected: $[0, S_1]$, $(S_1, S_1 + S_2]$, $(S_1 + S_2, S_1 + S_2 + S_3]$, $(S_1 + S_2 + S_3, S_1 + S_2 + S_3 + S_4]$; namely, $[0, 0.14]$, $(0.14, 0.63]$, $(0.63, 0.69]$, $(0.69, 1.00]$. We hence generate four random numbers: $r_1 = 0.450126$, $r_2 = 0.110347$, $r_3 = 0.572496$, and $r_4 = 0.98503$. Correspondingly, the string S_1 has been randomly selected for once, S_2 twice, S_3 null, S_4 once. Therefore, we generate new strings $S_1' = (11000)_2 = (24)_{10}$, $S_2' = (01101)_2 = (13)_{10}$, $S_3' = (11000)_2 = (24)_{10}$, and $S_4' = (10011)_2 = (19)_{10}$.
Step 4. *Crossover*: Create new individuals by combining aspects of the selected individuals. Using the same example, we operate crossover on the last two binary digits of the strings S_1' and S_2' as well as S_3' and S_4'. We get:

$S_1'' = (11001)_2 = (25)_{10}$. $S_2'' = (01100)_2 = (12)_{10}$. $S_3'' = (11011)_2 = (27)_{10}$. $S_4'' = (10000)_2 = (16)_{10}$.
Step 5. Mutation: Add randomness into populations' genetics. We assume there is no mutation in this example.
Step 6. Repeat: Start again from Step 2 until a termination condition is reached.

We get 4G strings as:

$S_1 = (11111)_2 = (31)_{10}$, $S_2 = (11100)_2 = (28)_{10}$, $S_3 = (11000)_2 = (24)_{10}$, $S_4 = (10000)_2 = (16)_{10}$.

Because $S_1 = (11111)_2 = (31)_{10}$ reaches the biggest number of the five binary digits; namely, the highest ranking fitness has reached (one of the termination conditions), the iteration is terminated.

The termination conditions usually include:

- A solution is found that satisfies the minimum criteria.
- A fixed number of generations reached.
- An allocated budget (e.g., computational time, etc.) reached.
- The highest ranking solution's fitness has reached.
- A plateau no longer produces better results.
- Combinations of the above.

We therefore can summarize GA as Algorithm (4).

```
Result: Search output based on GA Algorithm
initializes $P(0)$; /* $y = P(x)$ is the fitness function*/;
$n \leftarrow 0$; /* $n$ is the generation number; $N$ is the max number*/;
/* $M$ is the max number of individuals*/;
while $n \leq N$ do
    for $i = 0; i < M; i++$ do
        Evaluate Fitness of $P(i)$;
    end
    for $i = 0; i < M; i++$ do
        Select Operation to $P(i)$;
    end
    for $i = 0; i < M; i++$ do
        Crossover Operation to $P(i)$;
    end
    for $i = 0; i < M; i++$ do
        Mutation operation to $P(i)$;
    end
    for $i = 0; i < M; i++$ do
        $P(i+1) \leftarrow P(i)$;
    end
    $n \leftarrow n + 1$;
end
```

Algorithm 4: GA for search

7.1.2.7 Online Search

Online search algorithm creates a map and finds a goal if the best one exists. Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function [4]. SA algorithm hunts the local best through neighbor selection randomly by examining their states, energy, and acceptance probability, after several iterations, finally terminates with the global optimal solution. Simulated annealing (SA) is for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. It is often used when the search space is discrete. Simulated annealing may be more efficient than exhaustive enumeration provided that the goal is merely to find an acceptable solution in a fixed amount of time rather than the best possible solution.

Although hill climbing algorithm is surprisingly effective at finding a good solution which has a tendency to get stuck in local optima, the simulated annealing algorithm is excellent at avoiding this problem and is much better on average at finding an approximate global optimum. When we measure the information retrieval results, we need to answer the questions,

- Computing completeness: whether it could guarantee to find one or not?
- Computing optimality: whether the solution is the better one?
- Computing complexity: how is the time complexity? How much memory we need?

In order to get the search evaluations, the training set, test set, and ground truth are needed. In information retrieval, we always need ground truth which is the dataset [12] manually labeled based on the facts.

7.2 Event Mining and Reasoning

The main goal of event mining is to provide automatic manner for surveillance event that is used to respond the real-time observation of people and vehicles in surveillance [6]. In surveillance, searching for an important clue in order to solve a problem is just like finding a needle in a haystack [14].

7.2.1 Event Mining

The typical categories of event mining include,

- Mining from event stream
- Mining from sequence events
- Event graph mining.

7.2.2 Event Reasoning

7.2.2.1 Forward Chaining

Forward chaining is one of the two primary methods of reasoning when using an inference engine. Forward chaining is a popular implementation strategy for expert systems in computational intelligence.

Forward chaining starts with the available data and uses inference rules to extract more data until a goal is reached. An inference engine using forward chaining searches the inference rules until it finds one where the antecedent is known to be true. When such a rule is found, the engine can infer the new information from the given data. Inference engines will iterate through this process until a goal is reached. The name "forward chaining" comes from the fact that the computer starts with the data and reasons its way to the answer, as opposed to backward chaining.

7.2.2.2 Backward Chaining

Backward chaining (or backward reasoning) is an inference method that is described as working backward from the goal(s). It is used in automated theorem provers, inference engines, proof assistants and other applications.

Backward chaining is one of the two most frequently used methods of reasoning with inference rules and logical implications which usually employs a depth-first search strategy.

Backward chaining starts with a list of goals and works backward from the consequent to the antecedent to see if there is data available that will support any of these consequents. An inference engine using backward chaining would search the rules until it finds one which has a consequent that matches a desired goal. If the antecedent of that rule is not known to be true, then it is added to the list of goals in order for one's goal to be confirmed; one must also provide data that confirms this new rule.

7.2.2.3 Probabilistic Reasoning

A dynamic Bayesian network (DBN) is a Bayesian network which relates to each other [4]. DBNs have shown the potential for a wide range of data mining. For example, in speech recognition, digital forensics, protein sequencing, and bioinformatics, DBNs have shown to produce equivalent solutions to hidden Markov model (HMM) and Kalman filters [3].

Dempster–Shafer (DS) theory refers to the original conception of the theory of Dempster and Shafer. DS theory allows one to combine evidences from multiple sources and reach at a degree of belief represented by a function. In particular, different rules for combining evidences are often with a view to handle conflicts in evidence better.

Kalman filtering [13] also known as linear quadratic estimation (LQE) is an algorithm that uses a series of measurements observed over time, contains noises (random variations) and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone. More formally, Kalman filter operates recursively on streams of noisy input data to produce a statistically optimal estimate of the underlying system state.

Kalman filtering works in a two-step process. In the prediction step, the filter produces estimates of the current state variables along with their uncertainties. Once the outcome of the next measurement is observed, these estimates are updated using a weighted average, with more weight being given to estimates higher certainty. Because of the recursive nature of algorithms, it can run in real time using only the present input measurements and the previously calculated state as its uncertainty matrix.

7.3 Event Exploration and Discovery

7.3.1 Event Discovery

Event discovery derives knowledge from the existing knowledge for obtaining the unknown acknowledge; the input includes event databases (eBase), and event entities in text, event graphs in contextual information, event web [10, 16] as the event carrier as well as the output has metadata, tags, rules, ontological results of events, etc. [7, 15].

7.3.2 Event Exploration

Event exploration [16] is to enrich the intellectual capital in knowledge of event, the main tasks of event exploration for a computer include:

- To know what it already knows.
- To grow what it knows
- To share what it knows
- To save what it knows

A computer gains knowledge from the training dataset, as a machine, a computer explores further unknown information from what it already knows, grows and enriches the knowledge, circulates the knowledge to the community, and saves the knowledge for further exploration. As the intelligence of our human beings, computers grow its knowledge in the cognitive ways using artificial intelligence.

In summery, the life cycle of events includes event detection and recognition, then archives them in an event database for search, retrieval, mining, and reasoning, finally for discovery and exploration. The keypoint of event life cycle is to generate new events based on existing ones by using various event operations and repeat the same life routine of an event. The newly generated events could join the life cycle and be applied to generate other events; thereafter, the event database will be updated [16]. The flow could be shown in Fig. 7.1.

7.4 Questions

Question 1. What is the concept event? What are the relationships between events?
Question 2. What are the operations between events?
Question 3. What is the life cycle of an event?
Question 4. What are the limitations of genetic algorithm (GA)?

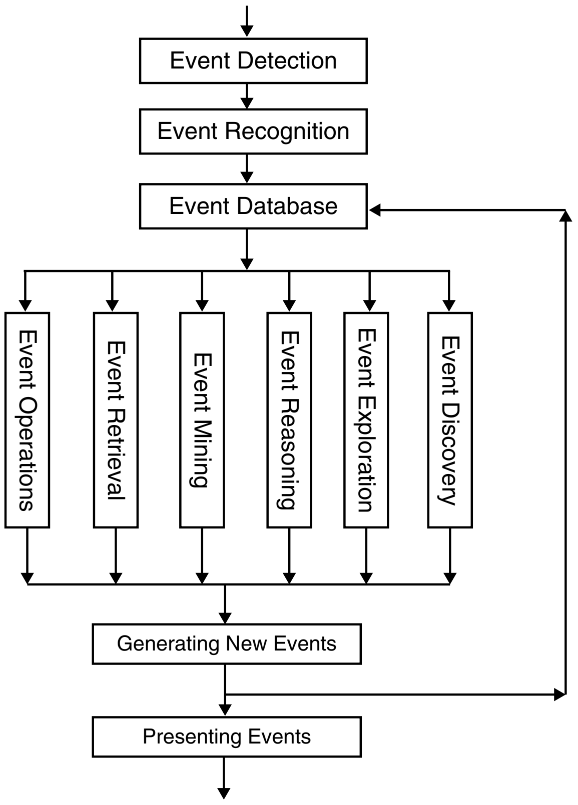


Fig. 7.1 The life cycle of events

References

1. Baeza-Yates R, Ribeiro-Neto B (1999) Modern information retrieval. ACM Press, New York
2. Bimbo A (1999) Visual information retrieval. Morgan Kaufmann Publishers, San Francisco
3. Bui H, Venkatesh S, West W (2001) Tracking and surveillance in wide-area spatial environments using the abstract Hidden Markov model. Pattern Recognit 15(1):177–195
4. Duda R, Hart P, Stork D (2001) Pattern classification. Wiley, New York
5. Ferland K (2009) Discrete mathematics. Houghton Mifflin Company, Boston
6. Hong X, Huang Y, Ma W, Varadarajan S, Miller P, Liu W, Zhou H (2016) Evidential event inference in transport video surveillance. Comput Vis Image Underst 144:276–297
7. Huang M, Liu Z (2012) Layered event ontology model. In: International conference on fuzzy systems and knowledge discovery, pp 948–951
8. Iosifidis A, Tefas A (2012) View-invariant action recognition based on artificial neural networks. IEEE Trans Neural Netw Learn Syst 23(3):412–424
9. Kasabov N (1996) Foundations of neural networks, fuzzy systems, and knowledge engineering. The MIT Press, Boston
10. Kumar S, Ragu S, Kumar S (2011) Embedded video surveillance with real time monitoring on web. Int J Math Trends Technol 2(1):46–49
11. Manning C, Raghavan P, Schutze H (1999) Introduction to information retrieval. Cambridge University Press, Cambridge

12. Oh S, Hoogs A, Perera A, Cuntoor N, Chen C, Lee JT (2011) A large-scale benchmark dataset for event recognition in surveillance video. In: IEEE conference on computer vision and pattern recognition, pp 3153–3160
13. Robertson NM, Reid ID (2011) Automatic reasoning about causal events in surveillance video. EURASIP J Image Video Process 2011:1–19
14. Russell S, Norvig P (2002) Artificial intelligence: a modern approach. Prentice Hall, New Jersey
15. SanMiguel JC, Martinez JM, Garcia A (2009) An ontology for event detection and its application in surveillance video. In: IEEE AVSS, pp 220–225
16. Yan W, Kieran D, Rafatirad S, Jain R (2011) A comprehensive study of visual event computing. Springer Multimed Tools Appl 55(3):443–481

8 Surveillance Alarm Making

8.1 Alarm Setting

In a surveillance system, alarming conditions usually have been well set at the time of sensor deployment. Once the conditions are satisfied, the alarming system will be activated. A plurality of communication channels will be employed to deliver the messages simultaneously. The alarming usually includes three types, i.e., rule-based alarming, probability-based alarming, and system-based alarming.

8.1.1 Rule-Based Alarming

If the triggering conditions such as sensitive area and restrained time are satisfied, then alarming function will be activated. Logically, we describe the alarming procedure as,

IF $<$ *condition* $>$ **THEN** $<$ *action* $>$ **END**

The rule-based alarming systems manage the alarming process by creating rules or triggering conditions which automatically work for alarm processing including delivering alarms to multiple communication channels and store the alarms. The systems provide a centralized location for organizing and viewing the alarms through networks or the Internet.

The rule-based alarming systems are important in monitoring for early detected hazards. The regions where the alarming systems are set and the thresholds applied are dependent on the specific needs of the surveillance system. In addition, we set specified conditions and actions to be performed at the first second when an alarm is activated.

W. Q. Yan, *Introduction to Intelligent Surveillance*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-10713-0_8

The rule-based alarming systems allow us to configure a rule onsite in alarm processing, ranking, and storing [10]. The systems provide tools and back-end services to effectively manage the alarm information.

A rule consists of two parts, namely a set of conditions and a set of actions. If all the conditions specified in a rule are satisfied, then the set of actions in the rule will be activated. The rules are employed in the order of procedure; the topmost rule has the highest precedence in the ranking list.

When more than one rule is specified, the incoming alarm is matched with the rules beginning with the topmost enabled rule in the list. If the incoming alarms match with any rules, then only corresponding actions are committed, and there will be no further processes of remaining rules.

8.1.2 Probability-Based Alarming

It is desired to check alarms from the view of risks, error avoidance, etc. When alarming conditions are set, and if we take the risk avoiding into consideration [16], we hope to make a decision with less errors [9]; hence, the best choice is to select the less errors [8]. Mathematically,

$$P(error|x) = \min(P(\omega_1|x), P(\omega_2|x)) \tag{8.1}$$

$$P(error|x) = \begin{cases} P(\omega_1|x), & P(\omega_1|x) \le P(\omega_2|x) \\ P(\omega_2|x), & P(\omega_2|x) < P(\omega_1|x) \end{cases} \tag{8.2}$$

where $P(\omega_i|x)$ is the error probability of the alarm x given the choice ω_i. We use Bayes' theorem for the conditional probability:

$$P(error|x) = \begin{cases} P(\omega_1|x), & P(x|\omega_1) \cdot P(\omega_1) \le P(x|\omega_2) \cdot P(\omega_2) \\ P(\omega_2|x), & P(x|\omega_2) \cdot P(\omega_2) < P(x|\omega_1) \cdot P(\omega_1) \end{cases} \tag{8.3}$$

If we treat the error as the difference between two conditional probabilities, we select positive difference for the right alarms:

$$g(x) = P(\omega_1|x) - P(\omega_2|x) \tag{8.4}$$

$$P(error|x) = \begin{cases} P(\omega_1|x), & g(x) \le 0 \\ P(\omega_2|x), & g(x) > 0 \end{cases} \tag{8.5}$$

For a monotone function, its logarithmic function is also monotonic.

$$g'(x) = \ln P(\omega_1|x) - \ln P(\omega_2|x) \tag{8.6}$$

If we use Bayes' theorem for the conditional probability, then

$$g'(x) = \ln[P(x|\omega_1) \cdot P(\omega_1)] - \ln[P(x|\omega_2) \cdot P(\omega_2)] \tag{8.7}$$

$$g'(x) = \ln \frac{P(x|\omega_1)}{P(x|\omega_2)} + \ln \frac{P(\omega_1)}{P(\omega_2)} \tag{8.8}$$

$$P(error|x) = \begin{cases} P(\omega_1|x), \; g'(x) \leq 0 \\ P(\omega_2|x), \; g'(x) > 0 \end{cases} \tag{8.9}$$

If we take risks into consideration, we select the case with less risk

$$R(error|x) = \begin{cases} R(\alpha_1|x), \; R(\alpha_1|x) \leq R(\alpha_2|x) \\ R(\alpha_2|x), \; R(\alpha_1|x) > R(\alpha_2|x) \end{cases} \tag{8.10}$$

where $R(\cdot)$ is the risk function.

If the risk is allowed to be expressed by using a linear combination of probabilities,

$$\begin{cases} R(\alpha_1|x) = \lambda_{11} \cdot P(\omega_1|x) + \lambda_{12} \cdot P(\omega_2|x) \\ R(\alpha_2|x) = \lambda_{21} \cdot P(\omega_1|x) + \lambda_{22} \cdot P(\omega_2|x) \end{cases} \tag{8.11}$$

Then,

$$R(error|x) = \begin{cases} R(\alpha_1|x), \; (\lambda_{11} - \lambda_{21}) \cdot P(\omega_1|x) \leq (\lambda_{22} - \lambda_{12}) \cdot P(\omega_2|x) \\ R(\alpha_2|x), \; (\lambda_{11} - \lambda_{21}) \cdot P(\omega_1|x) > (\lambda_{22} - \lambda_{12}) \cdot P(\omega_2|x) \end{cases} \tag{8.12}$$

Again, we use the Bayes' theorem [14]

$$R(error|x) = \begin{cases} R(\alpha_1|x), \; (\lambda_{11} - \lambda_{21}) \cdot P(x|\omega_1) \cdot P(\omega_1) \leq (\lambda_{22} - \lambda_{12}) \cdot P(x|\omega_2) \cdot P(\omega_2) \\ R(\alpha_2|x), \; (\lambda_{11} - \lambda_{21}) \cdot P(x|\omega_1) \cdot P(\omega_1) > (\lambda_{22} - \lambda_{12}) \cdot P(x|\omega_2) \cdot P(\omega_2) \end{cases} \tag{8.13}$$

$$R(error|x) = \begin{cases} R(\alpha_1|x), \; \frac{P(x|\omega_1)}{P(x|\omega_2)} \leq \frac{\lambda_{22}-\lambda_{12}}{\lambda_{11}-\lambda_{21}} \cdot \frac{P(\omega_2)}{P(\omega_1)} \\ R(\alpha_2|x), \; \frac{P(x|\omega_1)}{P(x|\omega_2)} > \frac{\lambda_{22}-\lambda_{12}}{\lambda_{11}-\lambda_{21}} \cdot \frac{P(\omega_2)}{P(\omega_1)} \end{cases} \tag{8.14}$$

Hence, alarm making from the rule-based aspect is to reduce errors and avoid risks in computational way.

Alarm making from the aspect of maximum a posteriori (MAP)-Markov random field (MRF), a risk in Bayes estimation is defined as,

$$R(f^*) = \int_{f \in \mathscr{F}} C(f^*, f) P(f|d) df \tag{8.15}$$

where d is the observation, $P(f|d)$ is the posterior distribution, $C(f^*, f) = \|f^* - f\|^2$ is a $\delta(f^*, f)$ cost function,

$$\delta(f^*, f) = \begin{cases} 0 \; f^* = f \\ 1 \; f^* \neq f \end{cases} \tag{8.16}$$

Therefore, if,

$$\frac{\partial R(f^*)}{\partial f^*} = 0$$

then, we have,

$$f^* = \underset{f \in \mathscr{F}}{\arg\max} P(f|d)$$

8.1.3 System-Based Alarming

System-based alarm is also called an event-driven one which adopts expert systems as the domain knowledge [3]. Therefore, whether an alarm is triggered or not is completely subject to the associated events with the current one. If one or many events as the triggering conditions are satisfied, the alarm will be activated [6,11].

This system-based alarming is derivative from event detection and recognition, has been applied to intelligent surveillance. MATLAB has the event-driven blockset as a toolbox to simulate the event-triggering processes. The blockset makes alarm setting, programming, and activation much easier; it also provides such an interface for sensor management and signal input. The automated control is associated with MATLAB image and vision toolbox, and any special events detected by MATLAB will be utilized for alarm making.

8.2 Decision Making

8.2.1 Simple Decision Making

In computational intelligence, a decision has the following properties:

- **Orderability**:
  If a decision is successive, namely $a \succ b$, meanwhile c is after b, $b \succ c$, then $a \succ c$ is explained as c is behind a.
- **Transitivity**:
  If it is true that decision a is transited to decision b, i.e., $a \to b$ and $b \to c$, then a is able to be transited to decision c, i.e., $a \to c$.
- **Continuity**:
  If a decision is continuous at x_0 that means the decision is continuous from left and right sides, i.e., $\lim_{x \to x_0} f(x) = f(x_0)$, then $\lim_{x \to x_0^+} f(x) = f(x_0^+) = \lim_{x \to x_0^-} f(x) = f(x_0^-) = f(x_0)$.
- **Substitutability**:
  If decision a substitutes decision b, namely $a \Leftrightarrow b$ and $b \Leftrightarrow c$, then a is able to be applied to replace c, $a \Leftrightarrow c$.

- **Monotonicity**:
  If decision a is more critical than that of b, $a \geq b$, then it will take greater effects than that of $b, f(a) \geq f(b)$. $f(\cdot)$ is a function defined on the decision domain.
- **Decomposability**:
  If decision x is within a predefined domain, it must be decomposed or factored by subdecisions. $\forall x \in \Omega, f(x) = h(\alpha) \cdot g(\beta)$, α and β are subdecisions, and $f(\cdot)$, $h(\cdot)$, and $g(\cdot)$ are the functions defined on the decision domain.

Decision is made based on a relationship network. A decision network (also called an influence diagram) is a compact graphical and mathematical representation of a decision situation. An influence diagram includes uncertainty node (oval), decision node(rectangle), and value node (octagon) [7].

An influence diagram sets evidence variables for the current state and possible value of the decision node. That means to set the decision node to that value, export the posterior probabilities for the nodes and export the result for the action.

Decision networks extend belief networks which include decision variables and utilities. In particular, a decision network is a directed acyclic graph.

8.2.1.1 Expert System

In artificial intelligence, an expert system is a computer system that emulates the decision-making ability of a human expert. Expert systems are designed to solve complex problems by reasoning about knowledge, represented primarily as *If* < *condition* > *Then* < *action* > rule rather than through conventional procedural code.

An expert system is divided into two subsystems: inference engine and knowledge base which represents facts and rules. The inference engine applies the rules to the known facts so as to deduce new facts.

In early expert systems, these facts were represented primarily as flat assertions of variables. Later, the knowledge base took on more structure and utilized concepts from object-oriented programming (OOP). Instances and assertions were in lieu of values of object instances.

The inference engine is an automated reasoning system that evaluates the current state of the knowledge base, applies relevant rules, and then asserts new knowledge into the knowledge base [12]. The inference engine may also include capabilities for explanation so that it can explain a user the chain of reasoning used to arrive at a particular conclusion by tracing back over the firing rules that resulted in the assertion.

There are primarily two modes for an inference engine: *forward chaining* and *backward chaining*. The different approaches are dictated by whether the inference engine is being driven by the antecedent or consequent of rule. In forward chaining, an antecedent fires and asserts the consequent.

For an expert system, the team consists of experts and engineers. The fundamental procedure of an expert system is listed as:

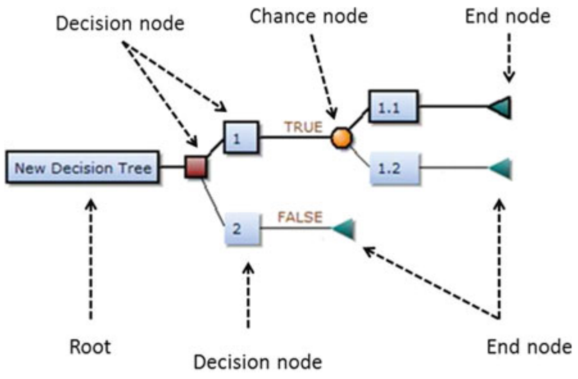


Fig. 8.1 Diagram of a decision tree

Step 1. *Acquire domain knowledge*. The knowledge is from two aspects: experts with domain knowledge and software engineers. Software engineers digitalize human experience from domain experts to understand the requirement of software design and implementation.

Step 2. *Create a causal model*. The team consisting of experts and software engineers initialize the project, set the initial parameters of the system, and start the design and development.

Step 3. *Simplify a qualitative decision model*. Based on domain knowledge and the causal model, a decision model is created to simulate the process of decision making from observations.

Step 4. *Assign probabilities*. The relevant probabilities are given from the observations.

Step 5. *Verify and refine the model*. The model will be verified and refined according to the real requirements, the errors and risks will be reduced to the minimal level.

Step 6. *Perform sensitivity analysis*. Test the designed model and make sure a small change could not affect the system dramatically.

8.2.1.2 Decision Tree

Decision tree [1] represents all Boolean functions which split the records of decision making based on the attributes that optimize current criterion. A decision tree needs to answer the questions: How to split the records? How to specify the attribute test condition? How to determine the best split? When to stop splitting?

In a decision tree, decision nodes are presented as squares, chance nodes are symboled by circles, and end nodes are marked by triangles as shown in Fig. 8.1.

In decision tree, we split the records for decision making based on the attribute test. Multiway split is used by many partitions. Binary split divides values into two subsets. Stop expanding a node when all the records belong to the same class or when all the records have similar attributes [1].

A decision tree is a flowchart-like structure in which internal node represents a "test" on an attribute, each branch represents outcome of the test, and each leaf node represents a class label; a decision will be taken after computing all attributes. The paths from the root to leaves represent classification routines.

In decision making, a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool where the expected values of competing alternatives are calculated. A decision tree consists of three types of nodes: decision nodes, chance nodes, and end nodes [1].

8.2.1.3 Deep Neural Decision Forests and Decision Networks

Mathematically, a tree is modeled as [15],

$$\mathbf{P}_T(y|x,\theta,\pi) = \sum_{l\in\mathscr{L}} \pi_{ly}\mu_l(x|\theta) \tag{8.17}$$

$$\mu_l(x|\theta) = \prod_{n\in\mathscr{N}} d_n^{(l\swarrow n)}(x;\theta)\cdot \bar{d}_n^{(n\searrow l)}(x;\theta) \tag{8.18}$$

$$d_n^{(l\swarrow n)}(x;\theta) = 1 - \bar{d}_n^{(n\searrow l)}(x;\theta) \tag{8.19}$$

where $d_n^{(l\swarrow n)}(x;\theta)$ is a sigmoid function, $l \swarrow n$ is the left subtree, and $n \searrow l$ is the right subtree of node n. $\mathscr{N}$ and $\mathscr{L}$ are internal nodes and the terminal nodes of the tree, respectively. π_{ly} is the probability of a sample reaching leaf l to take on class y. $\mu_l(x|\theta)$ is the probability that sample x will reach leaf l, $\sum_{l\in\mathscr{L}}\mu_l(x|\theta) = 1$, $\forall x \in \mathscr{X}$, θ is a parameter.

A deep neural decision forest (dNRF) $\mathbf{P}_{\mathscr{F}}(y|x) = \frac{1}{k}\sum_{h=1}^{k}\mathbf{P}_{T_h}\mathscr{T}(y|x)$ is an ensemble of decision trees $\mathscr{F} = \{T_1, T_2, \ldots, T_k\}$ with average. The log loss of a tree is,

$$\mathbf{R}(\theta,\pi,\mathscr{T}) = -\frac{1}{|\mathscr{L}|}\sum_{(x,y)\in\mathscr{L}} \log(\mathbf{R}_T(y|x,\theta,\pi)) \tag{8.20}$$

where $\mathscr{L} \subset \mathscr{X} \times \mathscr{Y} = \{(x, y)\}$. We minimize $\mathbf{R}(\theta,\pi,\mathscr{T})$ as:

$$\theta^{(i+1)} = \theta^{(i)} - \eta\cdot\frac{\partial\mathbf{R}(\theta^{(i)},\pi;\mathscr{B})}{\theta} \tag{8.21}$$

where $\eta > 0$ is a learning rate, $\mathscr{B} \subset \mathscr{T}$. π_{ly} could be updated as

$$\pi_{ly}^{(i+1)} = \frac{1}{Z_l^{(i)}}\sum_{(x,y)\subset\mathscr{T}} \frac{\mathbf{1}_{y=y'}\pi_{ly}^{(i)}\mu_l(x|\theta)}{P_T(y|x,\theta,\pi^{(i)})} \tag{8.22}$$

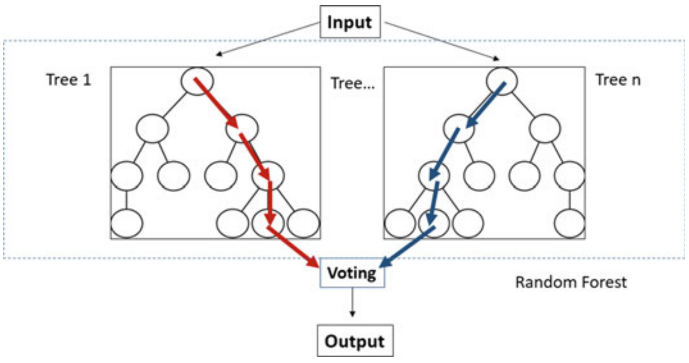


Fig. 8.2 Random forest for decision making

Therefore, $\min(\mathbf{R}(\theta, \pi, \mathscr{T}))$ could be implemented by using the mini-batch consisting of the optimized π and θ. A deep neural decision forests (dNDF) can be implemented by using typically available fully connected (or inner product) and sigmoid layers in DNN frameworks,

In decision making, random forest could help us to make correct decision based on voting theory of pattern classification. When we have multiple decision trees available, we therefore construct a random forest for decision making. In the forest, we have multiple trees to reflect the decisions, and we therefore need to fuse the decisions together, shown in Fig. 8.2. When the mixture of multiple decisions such as voting, boosting, etc., are taken, we think the correctness of decision making could be achieved.

Random forest [5] is an ensemble learning method for classification that operates by constructing a multitude of decision trees at training time and output by individual trees. The method combines the "bagging" idea and the random selection of features in order to construct a collection of decision trees with controlled variance.

After each tree is built, all of the data is run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data.

A decision diagram or a decision network is a compact graphical and mathematical representation of a decision situation. It is a generalization of a Bayesian network, in which not only probabilistic inference problems but also decision-making problems are modeled. The steps of a decision network are:

Step 1. Set evidence variables for the current state
Step 2. Set possible value of the decision node
Step 3. Set the decision node to that value
Step 4. Calculate the posterior probabilities for the nodes
Step 5. Output the result for the action.

In a decision network, each decision node is assigned a variable and possible value for current state. After this initialization, the posteriors for the nodes will be calculated based on priority and evidence. With the posteriors, the effective decision of each node will be made for practical applications.

8.2.2 Complex Decision Making

Markov decision processes (MDPs) provide a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker [16]. MDPs are useful for studying a wide range of optimization problems through dynamic programming and reinforcement learning which are used in a wide area of disciplines including robotics, automated control, economics, and manufacturing. A Markov decision process (MDP) model contains:

- A set of possible world states S
- A set of possible actions A
- A real-valued reward function $R(s)$
- A transition model $T(s, \pi, s^*)$.

A Bellman equation in Eq. (8.23) is a necessary condition for optimality associated with dynamic programming which writes the value of a decision problem at a certain point in terms of the payoff from some initial choices and the value of the remaining decision problem resulting from those initial choices. This breaks a dynamic optimization problem into simpler subproblems:

$$U(s) = R(s) + \gamma \cdot \max_{\pi} \sum_{s^*} T(s, \pi, s^*) \cdot U(s^*) \tag{8.23}$$

where Eq. (8.24) shows the Bellman updates,

$$U_{i+1}(s) \leftarrow R(s) + \gamma \cdot \max_{\pi} \sum_{s^*} T(s, \pi, s^*) \cdot U_i(s^*), i = 1, 2, \ldots \tag{8.24}$$

The convergence of these updates is guaranteed by Eq. (8.25),

$$\|U_{i+1}(s) - U_i(s)\| < \varepsilon \frac{1-\gamma}{\gamma}, \varepsilon > 0, 1 \geq \gamma > 0 \tag{8.25}$$

This ensures that $U_{i+1} \approx U$, namely

$$\|U_{i+1} - U\| < \varepsilon, \varepsilon > 0 \tag{8.26}$$

8.2.3 Alarming Channels

Once a surveillance alarm is given, it should be dispatched by multiple channels [2]. Typically, we apply the methods available below for message delivering:

- **Email**. An email is an official record to track back; emails could be checked offline but not timely. An email system could be linked to a broadcast system such as Twitter, Facebook, or a Web site.
- **Mobile SMS/MMS**. A mobile is a "Swiss Army knife" with multiple functionalities in communications; a mobile could be operated at anywhere in anytime if the ring could be rung.
- **Call**. Phone call is the shortest and most convenient way to communicate timely and fast. Most countries allow some special numbers to be dialed free for public services.
- **Speakers**. Loud speakers make sure everybody could be informed directly and indirectly in public area timely. It also helps to remind each other in alarming environment.
- **Siren**. Siren and flashing lights are for specific facilities who could not access the normal messaging channels such as disabled people or people enclosed in special space.

Technically, we implement the alarm delivering through [4],

- Using the SQL commands gets those alarm events from the event database (eBase)
- Using email system sends emails or SMS texts to mobiles
- Sending an email automatically uses PHP, Python, Ruby or other programming language.

For example, PHP language usually could be employed to send emails using the following way:

```
<?php
$to = "recipient@example.com";
$subject = "Hi!";
$body = "Hi,\ n \ n How are you?";
if (mail($to, $subject, $body)) {
echo("< p >Message successfully sent!</p>");
} else {
echo("< p >Message delivery failed...< /p >");
}
?>
```

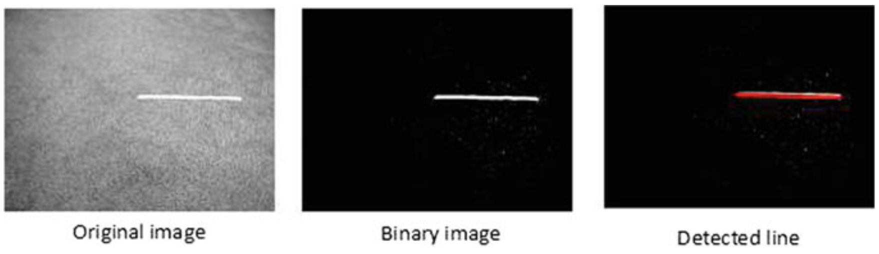


Fig. 8.3 Detected line from a surveillance image

8.3 Alarming for a Traffic Ticketing System

In this section, we introduce a new implementation that detects traffic ticketing events from videos and triggers actions when a car passes through the solid white stop line located opposite the traffic light at the moment when it turns red.

The first step is to detect the solid stop lines at the intersection. The location of stop line determines the event trigger condition. Detection of lines from an image is a well-known problem, it can be effectively solved using Hough transform (HT) [13]. Due to the robustness of HT algorithm, it is the most convenient and powerful method for straight line extraction. The equation for HT is defined as Eq. (8.27):

$$\rho = x \cdot cos(\theta) + y \cdot sin(\theta) \tag{8.27}$$

where (x, y) coordinates will be transferred to space. Hence, to find a line in (x, y) space has been converted to search points in (ρ, θ) space. Then, a vote stage is operated to rank peaks in the Hough space where the peaks in Hough space will denote the lines.

After obtained the location of the stop line in the footage, foreground detection will be conducted automatically because the video may contain slight shaking from the camera and other moving objects such as pedestrians.

In the traffic ticketing management, the next crucial step is to search for the vehicle owner by using the registration plate number in a database.

The vehicle entity may consist of many attributes such as plate number, model number, year built, color, and status. The primary key of this table is its plate number. The registration information records how many vehicles are owned by a driver. The relationship between the owner entity and the vehicle entity is one-to-many, which means the owner can have more than one car. There are three tables: driver table, vehicle table, and registration table.

Figure 8.3 shows the detected line from a surveillance image. Figure 8.4 illustrates the line for surveillance alarming. Hough transform correctly identifies the stop line before the moving vehicle detection. The location of the stop line is stored and used later for event alarming. The condition for triggering the alarm is when a car passes the stop line [11].

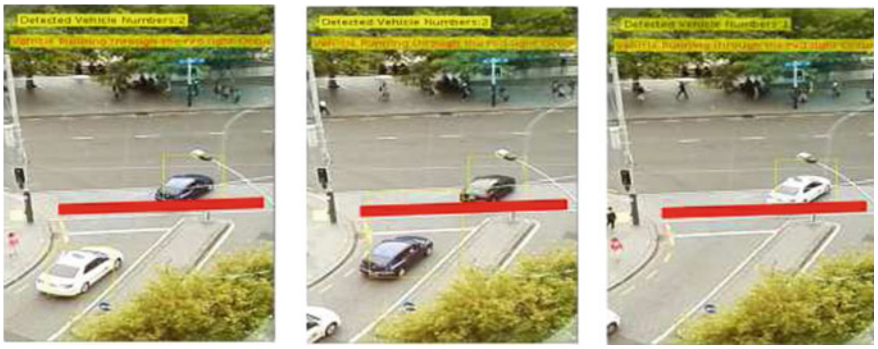


Fig. 8.4 Line for surveillance alarm making

8.4 Incident Responses

Once a disaster occurred, we need to respond it timely [17]. The professional steps include:

1. **Preparation**: Good strategies and tactics will lead to good practices; training well is one of the effective ways for preparation. The well-designed processes and operations are double-checked at the preparation time, and well preparation will lead to operation in logical sequence; avoid rush. For example, we use "drop," "cover," and "hold" in earthquake practice. The drill is: drop to the floor, get under cover, and hold on to furniture.
2. **Identification**: Once an alarm is given, the source should be tracked and the alarm could be identified. The dispensable examination will make the further work much on focus. Usually, the alarms are unique and should be reported in real situation.
3. **Containment**: This action makes sure that the alarm process is under control and committed on track. The dynamic monitoring to the alarm should be taken, any changes of the situation should be well known, and the relevant operations and processes should be well designed before this containment.
4. **Eradication**: An alarm or risk is permanently removed and taken out after the incident. This makes the alarming process clear, especially when dealing with multiple alarms. The best way to deal with alarms is one by one, in case one alarm mixes and triggers another.
5. **Recovery**: The process after the alarm making, the scene should be recovered, and the layout should be the same as before the alarms. The effective commitment will assist the next round of alarm identification and incident response which starts from a very clear point.
6. **Lessons learned**: Learning from the incident or failure is helpful to avoid further losing or making better success. The experience accumulated in the alarm

processing should be introduced to colleagues and makes the peers aware what the standard way the alarm processing is. The effective process will lead to the success of the team in alarming process and incident response.

8.5 Questions

Question (1) What is a decision tree? How to use decision trees to make surveillance alarms?
Question (2) What is a random forest? How to deal with the alarms using random forest?
Question (3) What is the difference between a decision tree and a decision diagram?
Question (4) What is an expert system? How to use it for making surveillance alarms?
Question (5) Why policy is important in surveillance alarming and incident response?

References

1. Alpaydin E (2010) Introduction to machine learning, 2nd edn. The MIT Press, Massachusetts
2. Alm H, Osvalder AL (2012) The alarm system and a possible way forward. Work 41(1):2840–2844
3. Artikis A, Baber C, Bizarro P, Canudas-de-Wit C, Etzion O, Fournier F, Schuster A (2014) Scalable proactive event-driven decision making. IEEE Technol Soc Mag 33(3):35–41
4. Bandini S, Bogni D, Manzoni S (2002) Alarm correlation in traffic monitoring and control systems: a knowledge-based approach. In: ECAI, pp 638–642
5. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
6. Cong Y, Yuan J, Liu J (2011) Sparse reconstruction cost for abnormal event detection. In: IEEE CVPR, pp 3449–3456
7. Filicheva S, Zaikin A, Kanakov O (2016) Dynamical decision making in a genetic perceptron. Phys D Nonlinear Phenom 318:112–115
8. Huanxiong X, Shuwen S, Yiwu Y, Wenzeng G, Shanshan Z, Jie W (2011) Design and realization of fire alarm by determining probability based on multi-sensor integrated. Comput Meas Control 2:42
9. Hubballi N, Suryanarayanan V (2014) False alarm minimization techniques in signature-based intrusion detection systems: a survey. Comput Commun 49:1–17
10. Hu J, Yi Y (2016) A two-level intelligent alarm management framework for process safety. Saf Sci 82:432–444
11. Jiang F, Wu Y, Katsaggelos AK (2007) Abnormal event detection from surveillance video by dynamic hierarchical clustering. In: IEEE ICIP, pp V-145
12. Kasabov N (1996) Foundations of neural networks, fuzzy systems, and knowledge engineering. The MIT Press, Boston
13. Klette R (2014) Concise computer vision. Springer, London
14. Kohavi R (1996) Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In: IEEE KDD, pp 202–207

15. Kontschieder P, Fiterau M, Criminisi A, Rota Bulo S (2015) Deep neural decision forests. In: IEEE ICCV, pp 1467–1475
16. Lomi V, Tonetto D, Vangelista L (2003) False alarm probability-based estimation of multipath channel length. IEEE Trans Commun 51(9):1432–1434
17. Sun E, Zhang X, Li Z (2012) The internet of things (IOT) and cloud computing (CC) based tailings dam monitoring and pre-alarm system in mines. Saf Sci 50(4):811–815

Surveillance Computing 9

In 1975, Dr Gordon Moore, co-founder of the company Intel, rectified his observations and reiterated doubling every 2 years in the number of components per integrated circuit and predicted this rate of growth would continue for at least another decade. The prediction, later called Moore's law, has spurred the computer chips industry in decades.

9.1 Challenges of Modern Computing

Since 2005, we have experienced the bottleneck of computing, Moore's law seems coming to an end, chips speed stopped increasing, and transistors could not be squeezed into the small space anymore; hence, computer CPU power could not increase too much. Therefore, supercomputing and high-performance computing are needed indeed [36,46,46].

Since the wake of supercomputing era in 1960s, the supercomputing technology has an enormous development ever since. Within almost two decades, supercomputers have gone up to 200,000 times faster and have been used in many areas of development that requires millions of processors.

9.1.1 Parallel Computing

Parallel computing is a form of computation in which a breadth of calculations is fulfilled simultaneously, operating on the principle that large tasks often are divided into small ones, which are then solved concurrently ("in parallel"). There are different forms of parallel computing: bit-level, instruction-level, and task parallelism. Parallelism has been put into practice for decades in high-performance computing [36]. As power consumption has become a concern in recent years, parallel computing

W. Q. Yan, *Introduction to Intelligent Surveillance*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-10713-0_9

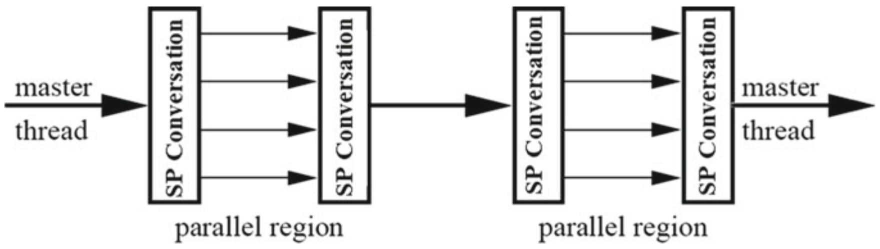


Fig. 9.1 Serial-parallel programming

has become the dominant paradigm in computer architecture, mainly in the form of multicore processing.

Parallel computers are roughly classified into the levels at which the hardware supports multicore and multiprocessor having multiple processing elements within a single machine, while clusters and grids are in the use of multiple computers to work on the same task. Specialized parallel architectures are utilized alongside traditional processors for accelerating specific tasks [30,40].

Traditionally, computer software has been written in serial way. The instructions are executed on a central processing unit of one computer. One instruction may be executed at a time only after previous one has been finished; the next is to be executed. Parallel computing, on the other hand, uses multiple processing elements simultaneously. This is accomplished by breaking a task into independent parts so that each processing element executes its part of the algorithm simultaneously with the others. The processing elements are diverse and include resources such as a single computer with multiple processors, several networked computers, specialized hardware, or any combinations of them.

Parallel programs are more difficult to be written than sequential ones since concurrency introduces new classes for potential software. Communication and synchronization between the different subtasks are typically to get good parallel program performance. Figure 9.1 shows a serial and parallel model for solving the parallel programming problem.

Concurrent programming languages, libraries, APIs, and parallel programming have been created for programming in parallel which utilize the memory architecture: shared memory, distributed memory, or shared distributed memory [45].

OpenMP is the most broadly used shared memory APIs, whereas Message Passing Interface (MPI) is adapted, one part of a program promises to deliver a required datum to another part of a program. In the runtime library of OpenMP, the "vcomp.lib" and "vcompd.lib" provide functions of multithread-based dynamic link for C/C++ programming. A typical example of the OpenMP routine is,

```
# include “omp.h”
void main(){
# pragma omp parallel{ /* create threads */
int ID = omp_get_thread_num();
printf(" hello(%d) , ID);
printf(" world(%d) \n, ID);
}
}
```

The output is,

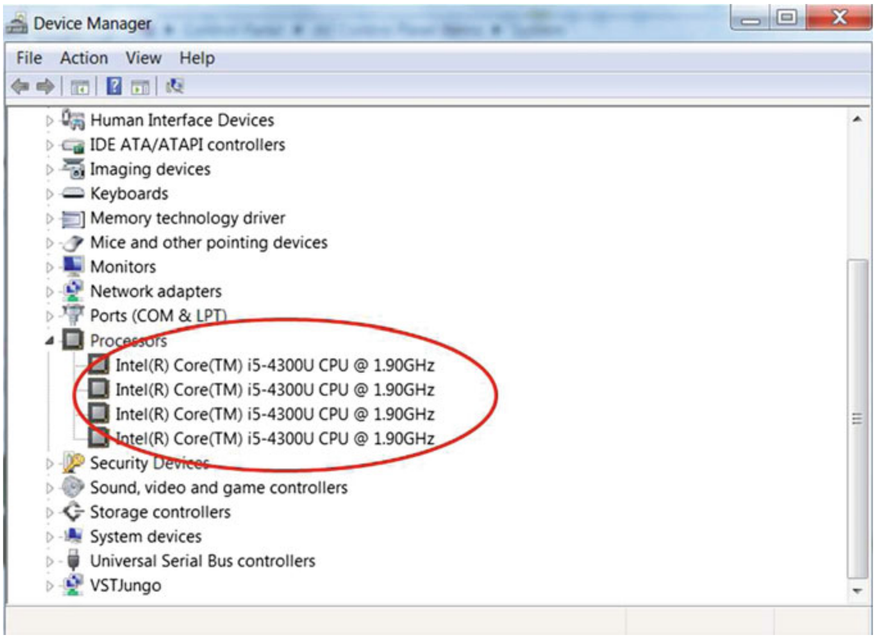
"hello(1) hello(0) world(1) world(0)"
where *omp_get_thread_num*$(\cdot)$ is the function for creating threads, the relevant functions also include: *omp_get_max_thread*$(\cdot)$, *omp_get_num_thread*$(\cdot)$, *omp_set_num_thread*$(\cdot)$, etc.

9.1.2 Multicore Processing

A multicore processor is a single computing component with two or more independent central processing units (called "cores"), which read and execute program instructions. The CPU instructions are ordinary and basic ones such as "add," "move," and "branch," but the multiple cores run the instructions at the same time and increase overall speed for programs amenable to parallel computing. Manufacturers typically integrate the cores onto a single integrated circuit die (known as a chip multiprocessor or CMP) or onto multiple dies in a single-chip package.

Processors were originally developed with only one core. A multicore processor implements multiprocessing in a single physical package. Designers may couple cores in a multicore device tightly or loosely. For example, cores may or may not share caches, and may implement message passing or shared memory inter-core communication methods. Common network topologies to interconnect cores include bus, ring, two-dimensional mesh, and crossbar. Homogeneous multicore systems include only identical cores; cores are not identical. Just as with single-processor systems, cores in multicore systems may implement architectures such as vector processing or multithreading.

Multicore processors have been applied to networks, digital signal processing (DSP), and graphics [25,40]. The improvement in performance gained by using a multicore processor depends vary on software algorithms and the implementation. In particular, possible gains are limited by the fraction of software that is run in parallel. In the best case, so-called embarrassingly parallel problems may implement speedup factors near the number of cores; or even more if the problem is split up enough to fit within each core's cache, avoiding use of much slower main system memory. Most applications, however, are not accelerated so much unless programmers invest a prohibitive amount of effort in refactoring the whole problem. Figure 9.2 shows the configuration of a multicore system.



Human Interface Devices
IDE ATA/ATAPI controllers
Imaging devices
Keyboards
Memory technology driver
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
Processors
Intel(R) Core(TM) i5-4300U CPU @ 1.90GHz
Intel(R) Core(TM) i5-4300U CPU @ 1.90GHz
Intel(R) Core(TM) i5-4300U CPU @ 1.90GHz
Intel(R) Core(TM) i5-4300U CPU @ 1.90GHz
Security Devices
Sound, video and game controllers
Storage controllers
System devices
Universal Serial Bus controllers
VSTJungo

Fig. 9.2 Specifications of a multicore computer system

9.1.3 Cluster Computing

Cluster computing addresses the latest results in the fields that support high-performance distributed computing (HPDC). In HPDC environments, parallel and distributed computing techniques are applied to the solution of computationally intensive applications across networks of computers [57].

In a nutshell, network clustering connects independent computers together in coordinated fashion. Because clustering is a term used broadly, the hardware configuration of clusters varies substantially depending on the networking technologies chosen and the purpose.

9.1.4 Supercomputing

Supercomputing is historically achieved by vector computers, now are parallel or parallel vector. A supercomputer is a computational machine at the frontline of contemporary processing capacity, particularly the calculations happen at the speed of nanoseconds. While the supercomputers of the 1970s adopted only a few processors, in the 1990s, machines with thousands of processors began to appear, by the end of the twentieth century, massively parallel supercomputers with thousands of "off-the-shelf" processors were the normal. Since 2013, China's Tianhe-II supercomputer has been the fastest one in the world. In 2016, China's Sunway TaihuLight installed by China National Supercomputing Center was ranked as the world's fastest supercomputer on a Top500 list. In 2018, IBM Summit from Oak Ridge, USA is taking its leading position (122.3 PFLOPS).

The systems with massive numbers of processors generally take one of two paths. In one approach (e.g., in distributed computing), a large number of discrete computers (e.g., laptops) distributed across a network (e.g., the Internet) devote some or all of their time to solve a real problem; each individual computer (client) receives and completes small tasks, and reports the results to a central server which integrates the tasks from all the clients into the overall solution [40]. In another approach, a large number of dedicated processors are placed in close proximity to each other (e.g., in a computer cluster); this saves considerable time moving data around and makes it possible for the processors to work together (rather than on separate tasks), for example, in mesh and hypercube architectures [40].

Supercomputers play an important role in the field of computational science and are used for a great deal of computationally intensive tasks in various fields, including quantum mechanics, weather forecasting, climate research, oil and gas exploration, molecular modeling, and physical simulations.

Supercomputing covers a number of activities and highlights:

- *Domain-specific tertiary-level qualifications*: the domains are related to computer science, geosciences, physics, chemistry, mathematics, and engineering.
- *Parallel programming*: this programming includes distributed-memory programming and shared-variable programming.
- *High-performance computing*: the computing covers hybridization, performance analysis, optimization, and scaling [36].
- *Scientific computing*: typically scientific computing comprehends to numerical libraries and application-specific packages.
- *Data management*: data from various area with big volume need to be managed with database design, data grid, distributed computing, and metadata [8,10,16].

9.1.5 High-Performance Computing

High-performance computing (HPC) aims at solving scientific problems via supercomputers and fast networks as well as visualization [36]. HPC takes use of parallel processing for running advanced applications efficiently, reliably, and quickly [30]. The term HPC is occasionally used as a synonym for supercomputing, though technically a supercomputer is a system that performs at or near the currently highest operational rate for computers. Generally speaking, HPC solves problems via supercomputers plus fast networks and visualization.

9.1.6 Multithread Computing

A thread is a line or flow of control through a program. There are three models for creating threads: many to one, one to one, or many to many; this depends on how many processors will be applied to a program. Threads have been built into programming language Java. In Java, mutexes, condition variables, and thread join are used for the purpose of multithread synchronization in programming. Meanwhile, parallel platform OpenMP adopt directives, library routines, and environment variable to carry out the same work [36].

Multithreading is the ability of a program to manage its use by more than one user at a time or multiple requests by the same user without having multiple copies of the programming in the computer. Central processing units (CPU) have hardware support to efficiently execute multiple threads [40]. These are distinguished from multiprocessing systems (such as multicore systems) in that the threads have to share the resources of a single core: computing units, CPU caches, and translation lookaside buffer (TLB). Multithreading aims to increase utilization of a single core by using thread-level as well as instruction-level parallelism. As the two techniques are complementary, they are sometimes combined in systems with multithreading CPUs with multithreading cores. The advantages include:

- If a thread gets a vast of cache misses, the other thread(s) can continue, taking advantage of the unused computing resources, which thus can lead to faster overall execution, as these resources would have been idle if only a single thread was executed.
- If a thread cannot use all the computing resources of the CPU because instructions depend on each other's result, running another thread can avoid leaving these idle.
- If several threads work on the same set of data, they can actually share their cache, which leads to better cache usage or synchronization [40].

Meanwhile, the comments on multithreading also include:

- Multiple threads can interfere with each other when sharing hardware resources such as caches of translation look aside buffers [40].
- Execution time of a single thread could not be improved but can be degraded, even when only one thread is being executed. This is due to slower frequencies and additional pipeline stages that are necessary to accommodate thread-switching hardware.
- Hardware support for multithreading is more visible to software, thus requiring more changes to both applications and operating systems than multiprocessing.

In parallel computing, Fork–Join model is a way of setting up and executing parallel programs so that execution branches are off in parallel at designated points in the program, to "Join" (refers to merge) at a subsequent point and resume sequential execution. Parallel sections may fork recursively until a certain task granularity is reached. Fork–Join model can be considered as a parallel version of the divide and conquer paradigm shown in Fig. 9.1.

Fork–Join model is the main model of parallel execution in OpenMP framework; an OpenMP program begins with an initial thread. When any thread encounters a parallel construct using the parallel keyword, a team of threads are forked. An implicit barrier is used at the end of parallel construct. Any number of parallel constructs can be specified in a single program, parallel constructs may be nested. Depending on the implementation, a nested parallel structure may yield a team of threads or may be completed by a single thread [40].

9.1.7 Graphics Processing Unit (GPU)

Graphics processing unit (GPU), also occasionally called visual processing unit (VPU), is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. Modern GPUs are very efficient at manipulating computer graphics and image processing, the highly parallel structure makes them more effective than general purpose CPUs for algorithms where processing large blocks of data is operated in parallel. In a personal computer, a GPU can be presented on a video or graphics card, or it can be on the motherboard or in certain CPUs [49].

The term GPU was popularized first by Nvidia in 1999 which is a single-chip processor with integrated transforms, lighting, triangle setup, clipping, and rendering engines.

GPU programming is assisted by the libraries such as GPU-accelerated BLAS library, GPU-accelerated FFT library, GPU-accelerated sparse matrix library, and GPU-accelerated RNG library. Typically, the Basic Linear Algebra Subprograms (BLAS) library supports vector-vector operations (Level 1), matrix-vector operations (Level 2), and matrix-matrix operations (Level 3). The library has been applied to system solving, QR decomposition, SVD decomposition, eigenvalues, inverse, least squares, Markov chain Monte Carlo (MCMC), genetic algorithm (GA), etc.

With the emergence of deep learning, the importance of GPUs has increased. While training deep learning neural networks, GPUs can be more than 250 times faster than CPUs.

9.2 OpenCL Computing

9.2.1 OpenCL Programming

When we use Fork–Join model for arithmetic calculation, the matrix multiplication $(a_{ij})_{M\times N} = (B_{ij})_{M\times K} \cdot (C_{ij})_{K\times N}$ traditionally should be,

$$c_{i,j} = \sum_{k=1}^{K} a_{ik} \cdot b_{kj};\ i = 1, 2 \ldots, M;\ j = 1, 2 \ldots, N. \tag{9.1}$$

If we adopt the parallel computation, the multiplication will be,

$$c_{i,j} = \sum_{k=1}^{K} a_{ik} \cdot b_{kj}; j = 1, 2 \ldots, N; \forall i. \tag{9.2}$$

The difference between Eqs. (9.1) and (9.2) is that the latter uses parallel "for" loop; the elements of row i will be given at the "simultaneously" as the multiplication output.

In scalar multiplication, multithread ID will be applied and used to the calculation. Previously it was,

$$c_i = a_i \cdot b_i; i = 1, 2 \ldots, N. \tag{9.3}$$

Using parallel programming, now it is,

$$c_{id} = a_{id} \cdot b_{id}; id = 1, 2 \ldots, N_{id}. \tag{9.4}$$

The difference is that the multithread *id* has been applied to the scalar multiplication in Eq. (9.5) instead of i in Eq. (9.3). The *id* was allocated by the parallel platform automatically, that means the computation is parallel. Analogously, the famous plus operation "$c(i) + +;$" in C++ is,

$$c(id) + +; \tag{9.5}$$

where *id* is the number of the thread or device number in parallel platform. Based on the new definitions of matrix multiplication and scalar multiplication including the famous "++" count increasing operation, the traditional source code built on serial programming in linear algebra [30,36], system solving, differential equations, numerical analysis, probability and statistics, image/video and digital signal processing should be rewritten for catering the computational needs in parallel computing [25,49].

For instance, C/C++ compiler of CUDA [33] Toolkit provides the libraries for GPU-accelerated parallel computations such as Basic Linear Algebra Subprograms (BLAS) library, fast Fourier transform (FFT) library, sparse matrix library, and random number generator (RNG) library.

Among them, BLAS library has three levels: vector-vector operations, matrix-vector operations, and matrix-matrix operations. The functions cover the computations like solving a linear system [30,36],

$$A \cdot x = b \tag{9.6}$$

where A is a $N \times N$ matrix; b is a N dimension vector. The solution is,

$$x = A^{-1} \cdot b \tag{9.7}$$

A matrix A could be decomposed by using QR decomposition,

$$A = Q \cdot R \tag{9.8}$$

where A is a $N \times N$ matrix, Q is an orthogonal matrix, its columns are orthogonal unit vectors meaning $Q^T \cdot Q = I, Q^T = Q^{-1}$, and R is an upper triangular matrix or right triangular matrix.

Singular matrix decomposition (SVD) refers to

$$A = U \Lambda V^* \tag{9.9}$$

where Λ is a $m \times n$ diagonal matrix with nonnegative real numbers on the diagonal, U is an $m \times m$, and V^* is the conjugate transpose of the $n \times n$ unitary matrix V, $V^{-1} = V^T$.

Matrix eigenvalues λ_i are found by,

$$f(\lambda) = \det(A - I \cdot \lambda) = 0 \tag{9.10}$$

Namely,

$$X^{-1}AX = diag\{\lambda_1, \lambda_2, \ldots, \lambda_n\} \tag{9.11}$$

where X is eigenvector matrix.

The least squares are used for regression, for linear regression given the data points (x_i, y_i), $i = 1, 2, \ldots, N$, we assume the fitting linear equation is,

$$f(x) = ax + b \tag{9.12}$$

$$\varepsilon = \sum_{i=1}^{N} (y_i - f(x_i))^2 = \sum_{i=1}^{N} (f_i - ax_i - b)^2 \tag{9.13}$$

Hence, the linear system for the linear regression is,

$$\begin{cases} \frac{\partial \varepsilon}{\partial a} = 0 \\ \frac{\partial \varepsilon}{\partial b} = 0 \end{cases} \tag{9.14}$$

Solving this linear system [30,36], we can get a and b.

The Basic Linear Algebra Subprograms (BLAS) are a specified set of low-level subroutines that perform common linear algebra operations such as copying, vector scaling, vector dot products, linear combinations, and matrix multiplication. They are still used as a building block in higher-level mathematical programming languages and libraries, including the systems like Matlab, GNU Octave, Mathematica, NumPy, and the **R**.

9.2.2 Examples of OpenCL Programming

Tone mapping is a technique used in image processing and computer graphics to map one set of colors to another to approximate the appearance of high-dynamic-range images in a medium that has a more limited dynamic range. Tone mapping addresses the problem of strong contrast reduction from the scene radiance to the displayable range while preserving the image details and color appearance.

Tone mapping demonstrates how to use high-dynamic-range (HDR) rendering with tone mapping effect in OpenCL; Fig. 9.3 illustrates straightforward linear tone mapping and advanced tree-component curve-based tone mapping technique applied to HDR image.

God rays in atmospheric optics are rays of sunlight that appear to radiate from the point in the sky where the sun is located. These rays, which stream through gaps in clouds (particularly stratocumulus) or between other objects, are columns of sunlit air separated by darker cloud-shadowed regions. Despite seeming to converge at a point, the rays are in fact near-parallel shafts of sunlight, and their apparent convergence is a perspective effect.

As shown in Fig. 9.4, the God rays sample demonstrates how to use high-dynamic-range (HDR) rendering with God rays (crepuscular rays) effect in OpenCL. This

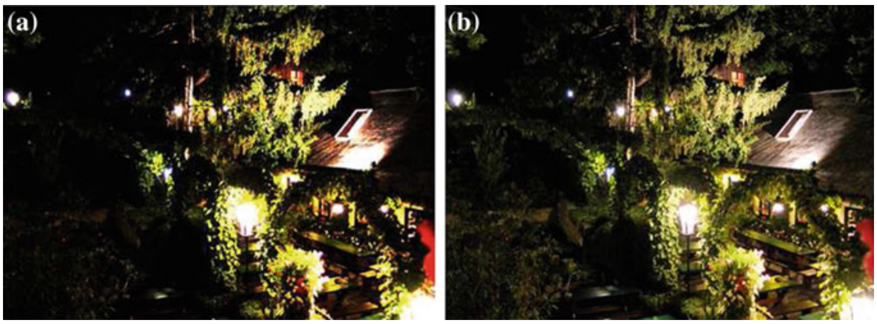


Fig. 9.3 The example of tone mapping, **a** before the tone mapping, **b** after the tone mapping

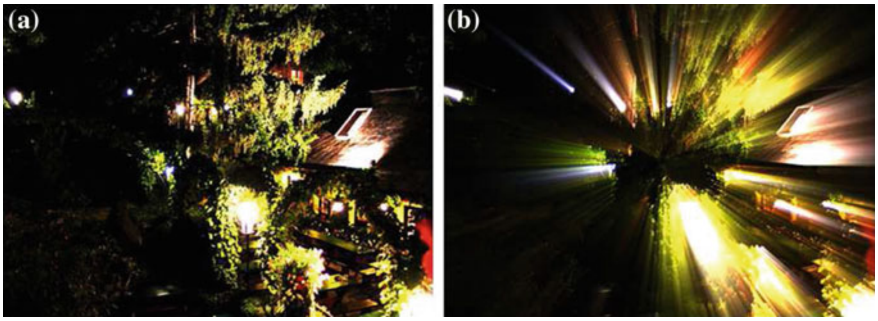


Fig. 9.4 The example of God ray, **a** before the processing, **b** after the processing

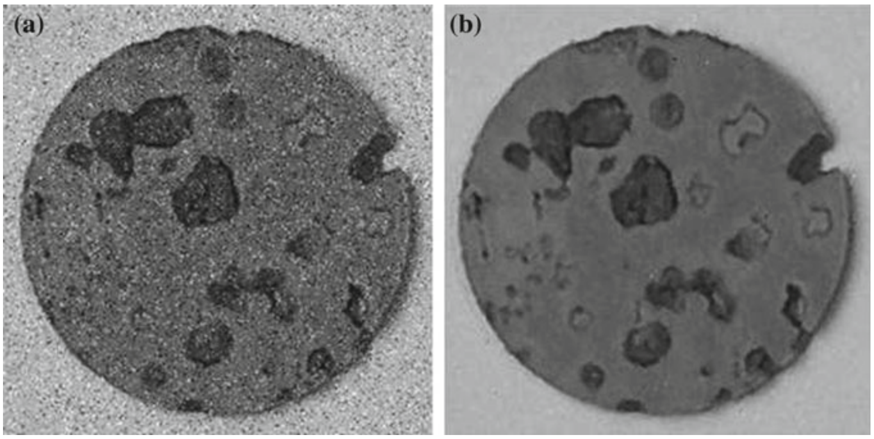


Fig. 9.5 The example of median filter, **a** before filtering, **b** after filtering

implementation optimizes rendering passes by sharing intermediate data between pixels during pixel processing, improves the method performance, and reduces data loads.

Median filtering is one kind of smoothing technique, as linear Gaussian filtering. All smoothing techniques are effective at removing noises in smooth patches or smooth regions of a signal, but adversely affect edges. Edges are of critical importance to the visual appearance of images. For the small to moderate levels of (Gaussian) noise, the median filter is demonstrably better than Gaussian blur at removing noise while preserving edges for a given, fixed window size. However, its performance is not that much better than Gaussian blur for high levels of noise, whereas, for speckle noise and salt-and-pepper noise, namely, impulsive noise, it is particularly effective.

In Fig. 9.5, median filter sample demonstrates how to use median filter in OpenCL. This implementation optimizes filtration process using implicit single instruction multiple data (SIMD).

9.3 MATLAB Parallel Computing

MATLAB provides the functionalities of parallel computing and built-in multithreading automatically enabled in core MATLAB since R2008a. Concretely, MATLAB provides parallel computing toolbox such as optimization toolbox and statistics toolbox.

MATLAB provided parallel computing tools controlled by its users for a variety of applications; it has the ability to leverage CPUs and GPUs to step applications further.

Specifically, MATLAB has the parallel for-loops (parfor) for running task-parallel algorithms on multiple processors and support for CUDA enabled NVIDIA GPUs

[33]. It also has the ability to run eight workers locally on a multicore desktop, computer cluster, and grid support with MATLAB Distributed Computing Server as well as the capabilities to run interactive and batch execution of parallel applications, especially for large dataset handling [16,17] and data-parallel algorithms [6].

MATLAB implemented parallel computing through:

- *Worker*: an independent MATLAB session that runs code distributed by the client [40].
- *Client*: the MATLAB session with which that distributes jobs to workers.
- *Parfor*: a parallel computing toolbox that distributes independent code segments to workers.
- *Random stream*: a pseudorandom number generator and the sequence of values it generates.
- *Reproducible computation*: a computation that can be exactly replicated even in the presence of random numbers.

A segment of source code for MATLAB programming is listed below as an example:

```
numberofRuns = 2000;
......
tic
parfor  i = 1 : numberofRuns
dataParfor(1, :) = runSimBioModel();
end
elapsedTimeParallel = toc;
```

9.4 Mobile Computing for Surveillance

Mobile phones have become an integral part of our lives. Nowadays, they come integrated with multimedia devices such as a camera, speaker, radio, and a microphone [38]. While primarily facilitating teleconversations, it also offers additional services such as textual communication, games, audio/video playing, radio, image/video capturing and transmission, alarm, calculator, and calendar [25,49].

More recently, mobile phones are working like personal computers. The powerful advancement of mobiles is its ability to receive signals from Wi-Fi and therefore link to this world. The more developed Apps based on the operating systems like Android and iOS are accelerating the popularization of mobile phones.

Because of broad applications of 3G/4G/5G/6G communications, a mobile could be used for voice communication, text at anywhere in any time; its screen is up with inertia resolution and is able to show real-time videos via broadband communication [49]. All smartphones at present can even access YouTube and run software such as Skype.

If surveillance Apps based on mobile platforms are developed, that means we could view surveillance data through this portable device [6]; if we could control the surveillance device, that will be much valuable.

Multimedia simplification [53] is a transcoding technology that could be applied to multimedia messaging system (MMS) of mobile for audio and video transmission. In video transmission, we only send those important frames or part of video streams to a receiver that will greatly save the receiver's time and communication resources [49].

When mobile computing meets deep learning [21], compressing networks [37] have been recommended as the solutions to reduce the workload caused by deep learning and neural networks.

Neural networks are typically overparameterized, there is significant redundancy for deep learning models. Network pruning has been used both to reduce network complexity and to reduce overfitting. This potentially makes deep neural networks more energy efficient to run on mobile.

A deep neural network is pruned by removing the redundant connections, keeping only the most informative connections. The weights are quantized so that multiple connections share the same weight, thus only the codebook (effective weights) and the indices need to be stored. Huffman coding is employed to take advantage of the biased distribution of effective weights.

9.5 Cloud Computing for Surveillance

Cloud computing is a broad expression that includes delivering hosted services such as computation, storage, and IO/network on the Internet [47,48]. The advantages of using cloud computing make organizations pay particular attention to it because it is on demand, self-service, location independent, elastic, and accessible by network from everywhere [50]. Cloud computing has the features,

- Significant savings in hardware infrastructure
- Significant savings in technical resources required to maintain the system
- Green solution based on the shared infrastructure at the hosting company
- Easy to install
- Easy to modify after the installation.

Cloud environment consists of three core components:

- Software as a service (**SaaS**): in SaaS, the whole software or application runs on physical server and becomes available to consumers on the Internet.
- Platform as a service (**PaaS**): PaaS is software and product development tools as well as programming languages which are leased by providers so that clients can build and deploy their own applications for special use [40].
- Infrastructure as a service (**IaaS**): IaaS delivers computing services, storage, and network, typically in the form of VMs running on hosts.

Cloud computing is closely related to virtualization of applications [43,44]. When application virtualization clients log onto a server, the server can issue different clients their own exclusion permissions according to the corporate directories. In this way, the permissions to download software by clients can be restricted. This also supports the function of centralized management and deployment of a single image. Once the software has been matched and used, the software preferences and profiles would be saved in the clients' cache memory to ensure that the clients are able to access those software when they are offline. Any service patches and updates of the software are applied to the application virtualization server image, when the clients match the software next time, they could choose to update the software and get the newer version. If clients are not sure about the newer version of the software compatibility to their operating systems [40], clients could revert back to the older version, but the condition is that the older version is still retained on the application virtualization server. In this way, there is a list of available software for the client, who exists as a graphical user interface (GUI)-based window on the clients' computers, and the clients are able to match and run any software from the list anytime [40].

In the second type of application virtualization, the software is loaded as an image from the application virtualization server remotely, the software runs in the application virtualization servers. This kind of application virtualization is well known as secure local virtual desktop images. The most significant advantage of the second type of the application virtualization is that it does not matter what operating system is in the clients' machines for executing the software as they are being run in the server [40]. Another advantage about this type of application virtualization is that it is more suitable for mobile devices, like mobile phones and iPads, as these mobile devices do not have enough processing power to run processor applications, but a powerful server does indeed.

The third type of application virtualization is presentation virtualization, which represents a category of virtualization technology that abstracts the processing of an application from the delivery of its graphics and I/O. With the application installed in a single location, presentation virtualization enables its use by multiple users. Each user connects to an individual session that stops a server supporting presentation virtualization. Within that session, there are the applications that have been provisioned for the user. In the presentation virtualization, the applications are being run on a remote computer; however, the user interface (UI) is transmitted over the network to the thin machine from the server.

Cloud computing is a storage-oriented system that is growing rapidly in recent years [61]. Video and image storage shows its challenges with the substantial requirement of infrastructure because visual surveillance needs storage facilities like big data that may be costly to any users. Also, once the storage disks are full or damaged, the huge data will be in risk, and thus the backup is absolutely needed. With timely data backup, users can easily access the data in anytime without worrying about cloud facilities.

VSaaS is primarily driven by numerous pivotal factors such as dynamic technology, cybersecurity, and remote access. A cloud-based visual surveillance system (CVSS) allows any users to benefit from upfront capital costs. The surveillance

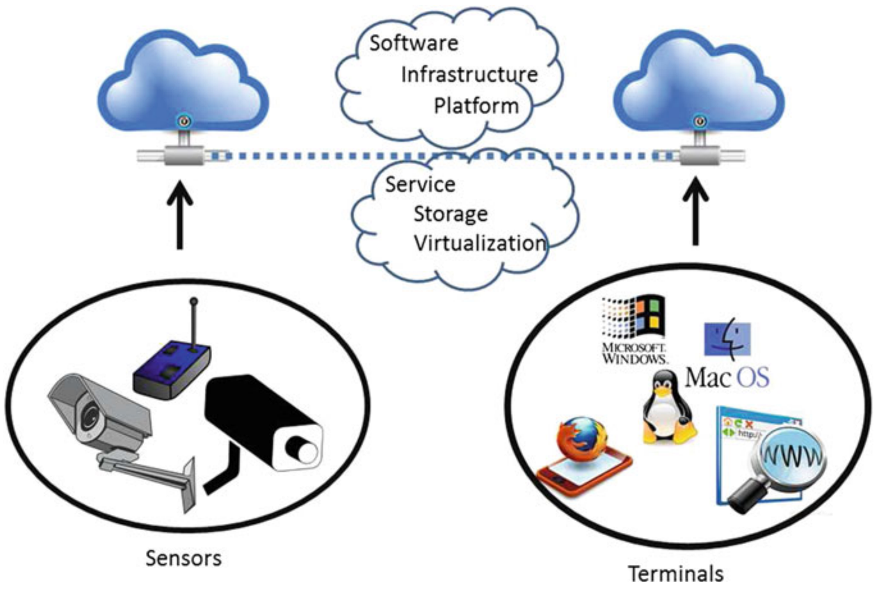


Fig. 9.6 Cloud computing for surveillance

system therefore lessens the resources and human workload. As visual surveillance requires sufficient space to store the big surveillance data, we think the first research problem is how to dynamically allocate enough space to deposit these videos and images from surveillance sensors. We believe pushing notification is an important feature of a cloud-based system.

If surveillance systems are based on cloud computing [2,4,5,11,41], a browser is secure enough for surveillance users to access the relevant surveillance content [35,39], security staff could use the browser to view alarms and respond incidents that will greatly reduce human workload [13,14,60] shown in Fig. 9.6. Traditional software for surveillance such as human face recognition [55], pedestrian, and vehicle detection [22] could be embedded into surveillance systems as a cloud-based service of software, the captured video footages, detected objects, and recognized events could be archived into the cloud-based databases [16,19,31,32,54]. The huge space for surveillance could be dynamically allocated on private cloud and hybrid cloud [28,42,58,59]. Intelligent surveillance could be as a service of cloud computing [23,24,26,29,34,56]. The push messaging service could be applied to mobile phones and other terminal devices which is dramatically different from the previous surveillance systems [1,3,7,9,11,12,38].

9.6 Questions

Question 1. What is the Fork–Join model? What are its advantages and disadvantages?
Question 2. Please list the differences between the serial programming and paralleling programming.
Question 3. Which toolbox of MATLAB provides the features of parallel computation?

References

1. Alamri A, Hossain MS, Almogren A, Hassan MM, Alnafjan K, Zakariah M, Alghamdi A (2015) QoS-adaptive service configuration framework for cloud-assisted video surveillance systems. Springer Multimed Tools Appl 75(21):13333–13348
2. Chen TS, Lin MF, Chieuh TC, Chang CH, Tai WH (2015) An intelligent surveillance video analysis service in cloud environment. In: ICCST, pp 1–6
3. Chen WT, Chen PY, Lee WS, Huang CF (2008) Design and implementation of a real time video surveillance system with wireless sensor networks. In: Vehicular technology conference, pp 218–222
4. Chen X, Xu JB, Guo WQ (2013) The research about video surveillance platform based on cloud computing. In: International conference on machine learning and cybernetics, vol 2, pp 979–983
5. Chen YL, Chen TS, Yin LC, Huang TW, Wang SY, Chieuh TC (2014) City eyes: an unified computational framework for intelligent video surveillance in cloud environment. In: IEEE international conference on internet of things (iThings), green computing and communications (GreenCom), IEEE and cyber, physical and social computing (CPSCom), pp 324–327
6. Davenport TH, Barth P, Bean R (2012) How big data is different. MIT Sloan Manag Rev 54(1):43–46
7. Dunkel D (2012) The "wonderful world" of cloud surveillance. SDM 42(6):50
8. Franks B (2012) Taming the big data tidal wave: finding opportunities in huge data streams with advanced analytics. Wiley, Hoboken
9. Frank H (2011) Cloud computing for syndromic surveillance. Emerg Health Threat J 4:71–71
10. Gandomi A, Haider M (2015) Beyond the hype: big data concepts, methods, and analytics. Int J Inf Manag 35(2):137–144
11. Hassan MM, Hossain MA, Abdullah-Al-Wadud M, Al-Mudaihesh T, Alyahya S, Alghamdi A (2015) A scalable and elastic cloud-assisted publish/subscribe model for IPTV video surveillance system. Clust Comput 18(4):1539–1548
12. Hossain MA (2014) Framework for a cloud-based multimedia surveillance system. Int J Distrib Sens Netw 2014(3):1–11
13. Hossain MA (2013) Analyzing the suitability of cloud-based multimedia surveillance systems. In: High performance computing and communications and IEEE international conference on embedded and ubiquitous computing, pp 644–650
14. Hossain MA, Song B (2016) Efficient resource management for cloud-enabled video surveillance over next generation network. J Mob Netw Appl 21(5):806–821

15. Hossain MS, Hassan MM, Al Qurishi M, Alghamdi A (2012) Resource allocation for service composition in cloud-based video surveillance platform. In: Multimedia and expo workshops, pp 408–412
16. Hu H, Wen Y, Chua T-S, Li X (2014) Toward scalable systems for big data analytics: a technology tutorial. IEEE Access 2:652–687
17. Huang T (2014) Surveillance video: the biggest big data. Comput Now 7(2) (Online Publication)
18. Jia Z, Wang H, Caballero RE, Xiong Z, Zhao J, Finn A (2012) A two-step approach to see-through bad weather for surveillance video quality enhancement. Mach Vis Appl 23(6):1059–1082
19. Karimaa A (2011) Video surveillance in the cloud: dependability analysis. In: International conference on dependability, pp 92–95
20. Kilts S (2007) Advanced FPGA design. Wiley, Hoboken
21. Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90
22. Li Q, Zhang T, Yu Y (2011) Using cloud computing to process intensive floating car data for urban traffic surveillance. Int J Geogr Inf Sci 25(8):1303–1322
23. Limna T, Tandayya P (2012) Design for a flexible video surveillance as a service. In: Image and signal processing, pp 197–201
24. Limna T, Tandayya P (2016) A flexible and scalable component-based system architecture for video surveillance as a service, running on infrastructure as a service. Springer Multimed Tools Appl 75(4):1765–1791
25. Meyer-Baese U (2007) Digital signal processing with field programmable gate arrays. Springer, New York
26. Neal D, Rahman SM (2012) Video surveillance in the cloud-computing. In: International conference on electrical and computer engineering, pp 58–61
27. Paul AK, Park JS (2013) Multiclass object recognition using smart phone and cloud computing for augmented reality and video surveillance applications. In: Informatics, electronics & vision, pp 1–6
28. Peng-Jung W, Yung-Cheng K (2014) Computing resource minimization with content-aware workload estimation in cloud-based surveillance systems. In: IEEE international conference on acoustics, speech and signal processing, pp 5050–5053
29. Prati A, Vezzani R, Fornaciari M, Cucchiara R (2013) Intelligent video surveillance as a service. In: Intelligent multimedia surveillance, pp 1–16
30. Rauber T, Runger G (2010) Parallel programming: for multicore and cluster systems. Springer, Berlin
31. Renkis Martin (2013) Bandwidth, storage, speed for cloud surveillance. Secur Syst News 16(5):16
32. Rodriguez-Silva DA, Adkinson-Orellana L, Gonz'lez-Castano FJ, Armino-Franco I, Gonz'lez-Martinez D (2012) Video surveillance based on cloud storage. In: Cloud Computing, pp 991–992
33. Sanders J, Kandrot E (2011) CUDA by examples: an introduction to general-purpose GPU programming. Addison-Wesley, Upper Saddle River
34. Sharma CM, Kumar H (2014) Architectural framework for implementing visual surveillance as a service. In: Computing for sustainable global development, pp 296–301
35. Shiwen Z, Yaping L, Qin L (2014) Secure and efficient video surveillance in cloud computing. In: International conference on mobile Ad Hoc and sensor systems, pp 222–226
36. Shonkwiler R, Lefton L (2006) An introduction to parallel and vector scientific computing. Cambridge University Press, New York
37. Han S, Mao H, Dally B (2016) Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding, In: ICLR
38. Song B, Hassan MM, Tian Y, Hossain MS, Alamri A (2015) Remote display solution for video surveillance in multimedia cloud. Springer Multimed Tools Appl 75(21):13375–13396

39. Song B, Tian Y, Zhou B (2014) Design and evaluation of remote video surveillance system on private cloud. In: Biometrics and security technologies, pp 256–262
40. Stallings W (2015) Operating systems: internals and design principles. Pearson Education Limited, New Jersey
41. Sunehra D, Bano A (2014) An intelligent surveillance with cloud storage for home security. In: Annual IEEE india conference, pp 1–6
42. Tekeoglu A, Tosun AS (2015) Investigating security and privacy of a cloud-based wireless IP camera: NetCam. In: International conference on computer communication and networks, pp 1–6
43. Tong Y, Yan W, Yu J (2015) Analysis of a secure virtual desktop infrastructure system. Int J Digit Crime Forensics 7(1):69–84
44. Tong Y (2015) Analytics of high secure desktop virtualization network systems. Masters thesis, Auckland University of Technology, New Zealand
45. Valera M, Velastin S (2005) Intelligent distributed surveillance systems: a review. Image Signal Process 152(2):192–204
46. Vecchiola C, Pandey S, Buyya R (2009) High-performance cloud computing: a view of scientific applications. In: International symposium on pervasive systems, algorithms, and networks, pp 4–16
47. Wang Z, Liu S, Fan Q (2013) Cloud-based platform for embedded wireless video surveillance system. In: Computational and information sciences, pp 1335–1338
48. Wenzhe J, Guoqing W, Zhengjun Z, Xiaoxue Y (2013) Dynamic data possession checking for secure cloud storage service. J Netw 8(12):2713–2721
49. Woods J (2012) Multidimentional signal, image, and video processing and coding. Elsevier Inc, Massachusetts
50. Xiong Y, Wan S, She J, Wu M, He Y, Jiang K (2016) An energy-optimization-based method of task scheduling for a cloud video surveillance center. J Netw Comput Appl 59:63–73
51. Xiong Y, Wan SY, He Y, Su D (2014) Design and implementation of a prototype cloud video surveillance system. JACIII 18(1):40–47
52. Xu Z, Mei L, Liu Y, Hu C, Chen L (2016) Semantic enhanced cloud environment for surveillance data management using video structural description. Computing 98(1–2):35–54
53. Yan W, Kankanhalli M (2007) Multimedia simplification for optimized MMS synthesis. ACM Trans Multimed Comput Commun Appl 3(1):5-es
54. Yan W, Kieran D, Rafatirad S, Jain R (2011) A comprehensive study of visual event computing. Springer Multimed Tools Appl 55(3):443–481
55. Yi S et al (2012) The model of face recognition in video surveillance based on cloud computing. In: Advances in computer science and information engineering, pp 105–111
56. Yu-Sheng W, Yue-Shan C, Tong-Ying J, Jing-Shyang Y (2012) An architecture for video surveillance service based on P2P and cloud computing. In: International conference on ubiquitous intelligence and computing, pp 661–666
57. Yuan X, Sun Z, Varol Y, Bebis G (2003) A distributed visual surveillance system. In: IEEE AVSS, pp 199–205
58. Zhang C, Chang EC (2014) Processing of mixed-sensitivity video surveillance streams on hybrid clouds. In: IEEE international conference on cloud computing, pp 9–16
59. Zhao ZF, Cui XJ, Zhang HQ (2012) Cloud storage technology in video surveillance. Adv Mater Res 532:1334–1338
60. Zhou L (2017) Design and implementation of a cloud based intelligent surveillance system. Master thesis, Auckland University of Technology, Auckland, New Zealand
61. Zhou L, Yan W, Shu Y, Yu J (2017) CVSS: a cloud-based visual surveillance system. Int J Digit Crime Forensics 10(1):1–13

Glossary

Data fusion The process of integration of multiple data and knowledge representing the same real-world object into a consistent, accurate, and useful representation.
Ontology The nature of existence as well as the basic categories of objects and their relations.
Metadata Data about data, namely additional information of a given set of data.
Camera calibration Correcting image distortions which are the phenomenon that lines of the image are curved due to the surface irregularities of camera lens when obtaining the images from sensors.
Event In textual topic detection and extraction, an event is something that happened somewhere at a certain time.
Finite State Machine (SVM) A model of behavior composed of a finite number of internal states (or simple states), the transitions between these states, the set of events that occur, and actions performed.
Telic event Telic event is an event that has end point of its time interval; however, atelic event has not end point.
Atomic event Atomic event is an elementary one which cannot be divided into further events and is the simplest type of event.
Composite event Composite event is defined by composition of two or more atomic events, and it is a complex event.
Bayesian network A graphical model for representing conditional independencies between a set of random variables.
Dynamic Bayesian network A Bayesian network represents sequences of variables.
Deep learning Deep learning has powerful ability of nonlinear processing using a cascade of multiple layers for feature transformation and end-to-end learning.

Index