Domingo Mery

Computer Vision for X-Ray Testing

Imaging, Systems, Image Databases, and Algorithms

Springer

Computer Vision for X-Ray Testing

Domingo Mery

Computer Vision for X-Ray Testing

Imaging, Systems, Image Databases, and Algorithms

Springer

Domingo Mery
Pontificia Universidad Católica de Chile
Santiago
Chile

ISBN 978-3-319-20746-9 ISBN 978-3-319-20747-6 (eBook)
DOI 10.1007/978-3-319-20747-6

Library of Congress Control Number: 2015942783

Springer Cham Heidelberg New York Dordrecht London

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media
(www.springer.com)

To Ximena, Anais and Valeria
who show me everyday
the X-rays of love

Foreword

The wavelengths of X-rays are far shorter than those of visible light, and even shorter than those of ultraviolet light. Wilhelm Conrad Röntgen (1845–1923) was awarded the first Nobel prize in Physics in 1901 for his contributions to the detection of electromagnetic radiation, and to the generation of X-rays, which are a form of electromagnetic radiation. Radiographs are produced by having X-rays, emitted from a source, geometrically assumed to be a point in three-dimensional (3D) space, recorded on a screen. This screen might have a slightly curved surface, but we can also see it (via defined mapping) as an image plane.

X-ray technology provides a way to visualize the inside of visually opaque objects. Pixel intensities in recorded radiographs correspond basically to the density of matter, integrated along rays; those readers interested in a more accurate description may wish to look up the interaction of X-rays with matter by way of photo-absorption, Compton scattering, or Rayleigh scattering by reading the first chapter of this book.

X-ray technology aims at minimizing scattering, by having nearly perfect rays pass through the studied object. Thus, we have a very particular imaging modality: objects of study need to fit into a bounded space, defined as being between source and image plane, and pixel intensities have a meaning which differs from our commonly recorded digital images when using optical cameras.

When modeling an X-ray imaging system we can apply much of the projective geometry, mathematics in homogeneous spaces, or analogous parameter notations: we just need to be aware that we are looking "backwards," from the image plane to the source (known as projection center), and no longer from the image plane into the potentially infinite space in front of an optical camera. Thus, it appears that the problem of understanding 3D objects is greatly simplified by simply studying a bounded space: using a finite number of source-plus-screen devices for recording this bounded space; applying photogrammetric methods for understanding multi-view recordings, and applying the proper interpretation (e.g., basically density) to the corresponding pixel values. Thus, this very much follows a common scenario of a computer vision, while also including image preprocessing and segmentation, object detection, and classification. The book addresses all of these subjects in the particular context of X-ray testing based on computer vision.

The briefly sketched similarities between common (i.e., optical-camera-based) computer vision and X-ray testing techniques might be a good motif to generate curiosity among people working in computer vision, in order to understand how their knowledge can contribute to, or benefit from, various methods of X-ray testing.

The book illustrates X-ray testing for an interesting range of applications. It also introduces a publically available software system and an extensive X-ray data base. The book will undoubtedly contribute to the popularity of X-ray testing among those in the computer vision and image analysis community, and may also serve as a textbook or as support material for undertaking related research.

Auckland
April 2015

Reinhard Klette

Preface

This book has been written in many *spatiotemporal coordinates*. For instance, some equations and figures were performed during my Ph.D. at the Technical University of Berlin (1996–2000). During that period, but in Hamburg, I took several X-ray images—that have been used in this book—in YXLON X-ray International Labs. After completing my Ph.D., and during my work in Santiago, Chile as associate researcher at the University of Santiago of Chile (2001–2003) and faculty member at the Catholic University of Chile (2004–to date) I have written more than 40 journal papers on computer vision applied to X-ray testing. During this time, I have developed a Matlab Toolbox that has been used in my research projects and in my classes teaching image processing, pattern recognition and computer vision for graduate and undergraduate students. Over the last few years, my graduate students have taken thousands of X-ray images in our X-ray Testing Lab at the Catholic University of Chile. Moreover, in my sabbatical year at the University of Notre Dame (2014–2015), I had the time and space to teach the computer vision course for students of computer sciences, electrical engineering, and physics, and I have been able to bring together all those related papers, diagrams, and codes in this book.

The present work has been written not only in three different countries (Germany, Chile, and the United States) over the last 15 years, but also in many different small places that provided me with the time and peace to write a paragraph, a caption of a figure, a code, or whatever I could. For example, I remember a Café in Michigan City where I spent various hours last winter writing this book with a delicious cappuccino beside me; or my study room in Fisher Apartments on Notre Dame Campus, looking out the window at a squirrel holding a nut; or on a narrow tray table while taking an Inter-Regio train between Berlin and Hamburg, which was where I drew a diagram using a pen and probably a napkin; and of course, my delightful office at the Catholic University of Chile with its breathtaking view of the Andes Mountains.

This book has been put together on the basis of four main pillars that have been constructed over the last 15 years: the first pillar is the set of journal and conference papers that I have published. The second corresponds to the material used in my classes and the feedback received from students when I have been teaching image processing, pattern recognition, and computer vision. The third pillar is the Matlab Toolbox that I was able to develop during this time, and which has been tested in several experiments, classes, and research projects, among others. The fourth pillar is the thousands of X-ray images that my research group has been taking in recent years at our Lab, and the X-ray images of die castings that I took in Hamburg. Over all this time, I have realized that this amount of work can all be brought together in a book that collects the most important contributions in computer vision used in X-ray testing.

Scope

X-ray imaging has been developed not only for its use in medical imaging for humans, but also for materials or objects, where the aim is to analyze—nondestructively—those inner parts that are undetectable to the naked eye. Thus, X-ray testing is used to determine if a test object deviates from a given set of specifications. Typical applications are analysis of food products, screening of baggage, inspection of automotive parts, and quality control of welds. In order to achieve efficient and effective X-ray testing, automated and semi-automated systems are being developed to execute this task. In this book, we present a general overview of computer vision methodologies that have been used in X-ray testing. In addition, some techniques that have been applied in certain relevant applications are presented: there are also some areas—like casting inspection—where automated systems are very effective, and other application areas—such as baggage screening—where human inspection is still used. There are certain application areas—like welds and cargo inspections—where the process is semi-automatic; and there is some research in areas—including food analysis—where processes are beginning to be characterized by the use of X-ray imaging. In this book, Matlab programs for image analysis and computer vision algorithms are presented with real X-ray images that are available in a public database created for testing and evaluation.

Organization

The book is organized as follows:

Chapter 1 (X-ray Testing): This chapter provides an introduction to the book. It illustrates principles about the physics of X-rays, and describes X-ray testing and imaging systems, while also summarizing the most important issues on computer vision for X-ray testing.

Chapter 2 (Images for X-ray Testing): This chapter presents a description of the 𝔾𝔻𝕏ray database, the dataset of more than 19,400 X-ray images used in this book to illustrate and test several computer vision methods. The database includes five groups of X-ray images: castings, welds, baggage, natural objects and settings.

Chapter 3 (Geometry in X-ray Testing): This chapter presents a mathematical background of the monocular and multiple view geometry that is normally used in X-ray computer vision systems.

Chapter 4 (X-ray Image Processing): This section covers the main techniques of image processing used in X-ray testing, such as image pre-processing, image filtering, edge detection, image segmentation, and image restoration.

Chapter 5 (X-ray Image Representation): This chapter covers several topics that are used to represent an X-ray image (or a specific region of an X-ray image). This representation means that new features are extracted from the original image; this can provide us with more data than the raw information expressed as a matrix of gray values.

Chapter 6 (Classification in X-ray Testing): This section covers known classifiers with several examples that can be easily modified in order to test different classification strategies. Additionally, the chapter covers how to estimate the accuracy of a classifier using hold-out, cross-validation and leave-one-out approaches.

Chapter 7 (Simulation in X-ray Testing): This chapter reviews some basic concepts of the simulation of X-ray images, and presents simple geometric and imaging models that can be used in the simulation.

Chapter 8 (Applications in X-ray Testing): This section describes relevant applications for X-ray testing such as the inspection of castings and welds, baggage screening, quality control of natural products, and inspection of cargos and electronic circuits.

Who Is This Book For

This book covers an introduction to computer vision algorithms that can be used in X-ray testing problems such as defect detection, baggage screening, 3D recognition, quality control of food products, and inspection of cargos and electronic circuits, among others. This work may not be ideal for students of computer science or electrical engineering who want to obtain a deeper knowledge of computer vision (for which purpose there are many wonderful textbooks on image processing, pattern recognition, and computer vision[1]). Rather, it is a good starting point for undergraduate or graduate students who wish to learn basic computer vision and its application in problems of industrial radiology.[2] Thus, the aim of this book is to cover complex topics on computer vision in an easy and accessible way.

[1] See for example [1–8].

[2] Obviously, the algorithms outlined in this book can be used in similar applications such as glass inspection [9] or quality control of food products using optical images [10]—to name but a few.

For instance, we present complex topics (such as support vector machines and SIFT descriptors) in such a straightforward way that any student who does not have much knowledge of these fields, can still understand how they work without having to analyze complicated equations.

Hands on!

In this book there is a Matlab Toolbox called 𝕏vis Toolbox.[3] with around 150 functions for computer vision in X-ray testing. Each function has a 'help' with an example in order to show its use in X-ray testing. Additionally, the book gives several Matlab examples that can be followed by the reader. These examples use 𝕏vis Toolbox. Moreover, there are around 19,400 X-ray images on the 𝔾𝔻𝕏ray database[4] that can be used to test different algorithms and codes. The available examples, toolbox and X-ray images can help people to learn more about computer vision for X-ray testing. The reader can modify the codes and can create his/her own codes in order to develop new functions for X-ray testing. The reader does not need any advance knowledge of Matlab to read and understand this document; however, he/she must have familiarity with basic linear algebra, geometry, and general knowledge of programming. If the reader does not (want to) use Matlab, he/she can also understand the examples from a traditional perspective by way of analyzing the input and the output given in each example. For more online resources, such as papers, figures and slides, the reader can visit the webpage of the present book at the following address: http://dmery.ing.puc.cl/index.php/book/.

Notre Dame and Santiago de Chile Domingo Mery
2015

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
2. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2 edn. John Wiley & Sons, Inc., New York (2001)
3. Faugeras, O., Luong, Q.T., Papadopoulo, T.: The geometry of multiple images: The laws that govern the formation of multiple images of a scene and some of their applications. The MIT Press, Cambridge MA, London (2001)
4. Forsyth, D.A., Ponce, J.: A modern approach. Computer Vision: A Modern Approach (2003)
5. Gonzalez, R., Woods, R.: Digital Image Processing, third edn. Pearson, Prentice Hall (2008)

[3] Available online on http://dmery.ing.puc.cl/index.php/book/xvis/.

[4] Available on-line on http://dmery.ing.puc.cl/index.php/material/gdxray/.

6. Hartley, R.I., Zisserman, A.: Multiple view geometry in computer vision, second edn. Cambridge University Press (2003)
7. Klette, R.: Concise computer vision: an introduction into theory and algorithms. Springer Science & Business Media (2014)
8. Szeliski, R.: Computer vision: Algorithms and applications. Springer-Verlag New York Inc (2011)
9. Carrasco, M., Pizarro, L., Mery, D.: Visual inspection of glass bottlenecks by multipleview analysis. International Journal of Computer Integrated Manufacturing **23**(10):925–941 (2010)
10. Mery, D., Pedreschi, F., Soto, A.: Automated design of a computer vision system for visual food quality evaluation. Food and Bioprocess Technology **6**(8):2093–2108 (2013)

Acknowledgments

This book could not have been written without the steady support and understanding of my wife Ximena and my daughters Anais and Valeria. Consequently, this book is dedicated to my nuclear family.

Thanks to my big family, my mother, my father, my sister, and my brother for the confidence and support.

Deep thanks also go to Reinhard Klette (University of Auckland) who carefully read the manuscript and gave me relevant and detailed suggestions to improve the book. A special debt of gratitude is owed to Peter Kozak for his detailed correction of my sometimes broken English (contaminated with my native Spanish and the German that I learned during my seven years in Germany). And I am very grateful to Simon Rees at Springer's UK and Karthik Raj D.S. Production Editor at Scientific Publishing Services based in Chennai, India for the edition and production of this book.

Finally, I would like to offer my gratitude to all my students, colleagues and co-authors who helped me in numerous discussions and suggestions, particularly the following: Gonzalo Acuña, José Miguel Aguilera, Marco Arias, Miguel Angel Berti, Sandipan Banerjee, Kevin Bowyer, Miguel Carrasco, Max Chacón, Aldo Cipriano, Patricio Cordero, Esteban Cortázar, Dieter Filbert, Rodrigo González, Dennis Hahn, Sergio Hernández, Nancy Hitschfeld, Thomas Jaegger, Valerie Kaftandjian, James Kapaldo, German Larraín, Gabriel Leiva, Hans Lobel, Iván Lillo, Carlos Madrigal, Daniel Maturana, Carlos Mena, Cristobal Moenne, Sebastián Montabone, Germán Mondragón, Diego Patiño, Karim Pichara, Christian Pieringer, Luis Pizarro, Vladimir Riffo, Doris Sáez, Alvaro Soto, Erick Svec, Romeu da Silva, Gabriel Tejeda, René Vidal, Yuning Zhao, Uwe Zscherpel, and Irene Zuccar.

This work was supported by Fondecyt Grant No. 1130934 from CONICYT, Chile, along with all the sabbatical support received from the Catholic University of Chile and the University of Notre Dame.

Contents

About the Author

Domingo Mery was born in Santiago de Chile in 1965. He received his Bachelor in Science (B.Sc.) degree in Electronic Engineering from the National University of Engineering, Peru, in 1989; a Diploma (M.Sc.) degree in Electrical Engineering from the Technical University of Karlsruhe, Germany, in 1992; and a Ph.D. with distinction awarded by the Technical University of Berlin in 2000. He was a Research Scientist at the Institute for Measurement and Automation Technology at the Technical University of Berlin with the collaboration of YXLON X-ray International. He was a recipient of a Scholarship from the Konrad-Adenauer-Foundation for his M.Sc., and was also awarded a Scholarship by the German Academic Exchange Service (DAAD) for his Ph.D. work. He was Associate Research in 2001 at the Department of Computer Engineering at the University of Santiago, Chile. At present he is a Full Professor at the Machine Intelligence Group (GRIMA) of the Department of Computer Sciences at the Pontificia Universidad Católica de Chile. He was Chair of the Computer Sciences Department between 2005 and 2009. His research interests include image processing for fault detection in aluminum castings, X-ray imaging, real-time programming, and computer vision. He is author of more than 60 technical SCI publications, and more than 60 conference papers. He is Local Co-chair of ICCV2015 (to be held in Santiago de Chile). He was program general chair of the PSIVT2007, program chair PSIVT2009 and General Co-chair of PSIVT2011 (Pacific-Rim Symposium on Image and Video Technology), and the 2007 Iberoamerican Congress on Pattern Recognition. During 2014–2015, he spent a sabbatical year at the Computer Vision Research Lab of the University of Notre Dame. At present, he is the Director of Research and Innovation at the School of Engineering of the Catholic University of Chile.

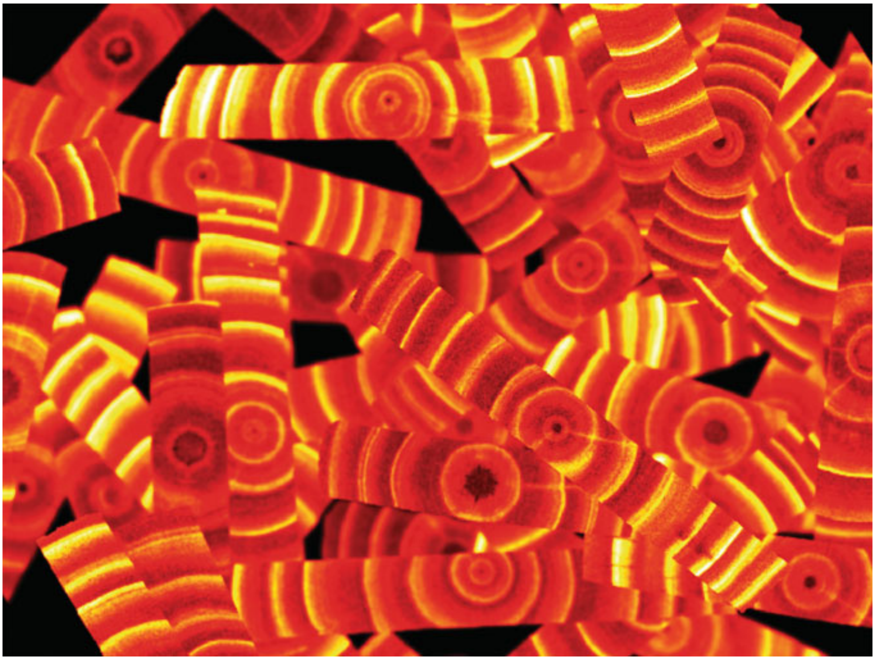
Awards in Nondestructive Testing:

- John Grimwade Medal for publishing the best paper during 2013 in Insight.[5]
- Ron Halmshaw Award for publishing the best paper during 2012 on industrial radiography in Insight (see footnote 5).
- Ron Halmshaw Award for publishing the best paper during 2005 on industrial radiography in Insight (see footnote 5).
- Best Paper Award. Panamerican Conference on Non-destructive Testing (PANNDT 2003).

[5] Insight is the Journal of the British Institute of Non-destructive Testing.

Chapter 1
X-ray Testing

Abstract X-ray testing has been developed for the inspection of materials or objects, where the aim is to analyze—nondestructively—those inner parts that are undetectable to the naked eye. Thus, X-ray testing is used to determine if a test object deviates from a given set of specifications. Typical applications are inspection of automotive parts, quality control of welds, baggage screening, analysis of food products, inspection of cargos, and quality control of electronic circuits. In order to achieve efficient and effective X-ray testing, automated and semiautomated systems based on computer vision algorithms are being developed to execute this task. In this book, we present a general overview of computer vision approaches that have been used in X-ray testing. In this chapter, we offer an introduction to our book by covering relevant issues of X-ray testing.



Cover image: *X-ray images of woods (series* `N0010` *colored with 'hot' colormap).*

D. Mery, *Computer Vision for X-Ray Testing*, DOI 10.1007/978-3-319-20747-6_1

1.1 Introduction

Since Röntgen discovered in 1895 [1] that X-rays can be used to identify inner structures, X-rays have been developed not only for their use in *medical imaging* for human beings, but also in *nondestructive testing* (NDT) for materials or objects, where the aim is to analyze (nondestructively) the inner parts that are undetectable to the naked eye [2]. NDT with X-rays, known as *X-ray testing*, is used in many applications such as inspection of automotive parts, quality control of welds, baggage screening, analysis of food products, inspection of cargos, and quality control of electronic circuits among others. X-ray testing usually involves measurement of specific part features such as integrity or geometric dimensions in order to detect, recognize, or evaluate wanted (or unwanted) inner parts. Thus, X-ray testing is a form of NDT defined as a task that uses X-ray imaging to determine if a *test object* deviates from a given set of specifications, without changing or altering that object in any way.

The most widely used X-ray imaging systems employed in X-ray testing are digital radiography (DR) and computed tomography (CT) imaging.[1] On the one hand, DR emphasizes high throughput. It uses electronic sensors (instead of traditional radiographic film) to obtain a digital X-ray projection of the target object, consequently it is simple and quick. A flat amorphous silicon detector can be used as an image sensor in X-ray testing systems. In such detectors, and using a semiconductor, energy from the X-ray is converted directly into an electrical signal that can be digitalized into an X-ray digital image [4]. On the other hand, CT imaging provides a cross-section image of the target object so that each object is clearly separated from any others; however, CT imaging requires a considerable number of projections to reconstruct an accurate cross-section image, which is time consuming.

In order to achieve efficient and effective X-ray testing, automated and semiautomated systems are being developed to execute this task that can be difficult (e.g., recognition of very small defects), tedious (e.g., inspection of thousand of similar items), and sometimes dangerous (e.g., explosive detection in baggage screening). Compared to manual X-ray testing, automated systems offer the advantages of objectivity and reproducibility for every test. Fundamental disadvantages are, however, the complexity of their configuration, the inflexibility to any change in the evaluation process, and sometimes the inability to analyze intricate images, which is something that people can generally do well. Research and development is, however, ongoing into automated adaptive processes to accommodate modifications.

X-ray testing is one of the more accepted ways for examining an object without destroying it. The purpose of this nondestructive method is to detect or recognize certain parts of interest that are located inside a test object and are thus not detectable to the naked eye. A typical example is the inspection of castings [5]. The material defects occurring in the casting process such as cavity, gas, inclusion, and sponge

[1] Computed tomography is beyond the scope of this book due to space considerations. For NDT applications using CT, the reader is referred to [3].

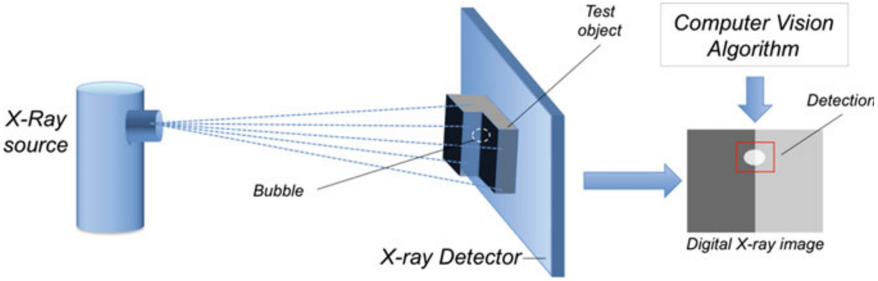


Fig. 1.1 Simple model of an X-ray computer vision system. In this example, a computer vision algorithm is used to detect a defect inside the test object automatically

must be detected to satisfy the security requirements; consequently, it is necessary to check 100 % of the parts.

The principle aspects of an X-ray testing system is illustrated in Fig. 1.1. Typically, it comprises the following steps:

- The test object is located in the desired position.
- The X-ray source generates X-rays which pass through the test object.
- The X-rays are detected and converted (e.g., by a flat panel or by an image intensifier and CCD camera) in order to obtain a digital X-ray image.
- Computer vision algorithms are used to evaluate the X-ray image.

In recent years, flat detectors made of amorphous silicon have been widely used as image sensors in some industrial inspection systems [6, 7]. In these detectors, the energy from the X-ray is converted directly into an electrical signal by a semiconductor (without an image intensifier). However, using flat detectors is not always feasible because of their high cost compared to image intensifiers.

The properties of the X-rays that are used in X-ray testing are summarized in the following: (i) X-rays can penetrate light blocking materials (e.g., metal) depending on a material's thickness; (ii) X-rays can be detected by photographic materials or electronic sensors; (iii) X-rays can spread a straight line; and (iv) X-rays can use many substances to stimulate fluorescence (fluoroscopy).

1.2 History

The discovery of X-rays by Röntgen in November 1895 [1] defines the beginning of the X-ray testing of metallic parts. A couple of days after the discovery of the 'X' radiation, he made radiographs of balance-weights in a closed box and a chamber of a shotgun (see Fig. 1.2). Röntgen observed that using X-rays, one can look not only into the inside of a human body, but also into metallic articles, if the strength and intensity of the X-rays are strong enough [8]. The potential use in the detection of hidden defects within armor-plates and machine parts was already envisioned at Yale University in 1896 [9].

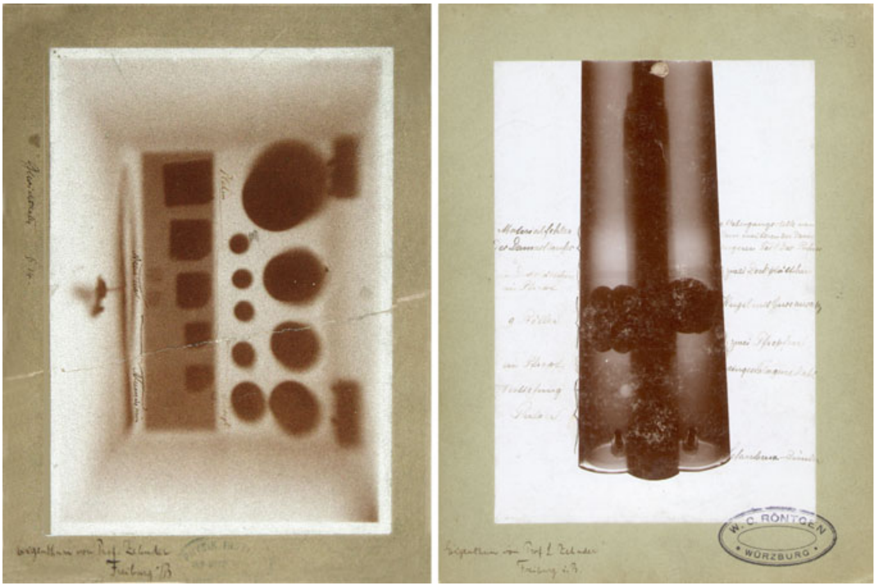


Fig. 1.2 X-ray image of balance-weights in a closed box and a shotgun taken by Wilhelm Conrad Röntgen in the summer of 1896. Courtesy of the Deutsches Röntgen-Museum in Würzburg

The industrial use of X-rays began in Germany only two decades after their discovery. X-ray testing took place at that time with the help of radiographic films [10]. Radioscopy with fluorescent screens was developed only toward the end of the 1930s and at the beginning of the 1940s. In the following years, closed cabinets were already being used for X-ray testing of aluminum castings in the automobile industry [11].

In 1948, the image intensifier was developed, which converts X-rays into a visible light [12]. Image intensifier technology was originally developed as a low-light enhancer for military night-vision devices [13]. The introduction of the image intensifier led to considerable progress in the inspection technique, since otherwise the examiner would have to regard the X-ray image on a fluorescent screen. The brightness of the image was so small that the eyes needed a long time to adapt to the dark. Into the image amplifier, an examiner could always look in the radiograph directly with the help of special optics. Image intensifiers, television equipment, and electrically controlled manipulators were developed further in the 1960s as radioscopic systems, which were widely used for casting and welding inspection in the 70s [9].

Computer tomography (CT) was developed in 1972 [12, 14]. With 2D-CT cross-section pictures of the object computed from its projections. These slices, which represent a reconstruction of the local distribution of the absorption coefficients of the object, are processed in order to find objects of interest in the test object. However, one disadvantage of the procedure is the high time requirement: for the reconstruction of meaningful slice images, both a minimum gate time per object position is necessary for a sufficient signal/noise ratio along with a minimum number of projections. For this reason, the use of computer tomography is so far limited in X-ray testing to the material development and research range, as well as to the examination of particularly important and expensive parts [15]. Later 3D-CT was developed, with which the whole object is reconstructed as voxels. State-of-the art industrial computer tomography used this kind of CT [3].

Approaches to the automatic image evaluation as well as image restoration were already used in the 1980s with the help of the image-processing techniques and CCD cameras [11]. The first fully automatic X-ray testing systems were installed in the industry at the beginning of the 1990s. One example can be found in the quality control of aluminum wheels performed by Alumetall Co. in Nuremberg, in which an automatic casting part recognition is also integrated using bar codes for the adjustment of the image analysis algorithms for different types of wheel [9].

At the end of the 1990s, flat panel detectors from amorphous silicon were industrially used in some test systems [16, 17]. With these detectors, the X-rays are converted by a semiconductor directly into electrical signals (without image intensifier). However, the X-ray testing with flat detectors was not always profitable due to their high costs (in the comparison to the image intensifier).

Today, novel X-ray detectors have been developed based on new semiconductors like CdTe or CZT [18]. They can count photons at high rates by discriminating different energy channels. Thus, image noise can be decreased, contrast can be enhanced, and specific materials can be imaged.

In the last few decades, fully automatic and semiautomatic test systems have been used in many applications as we will cover in Chap. 8.

1.3 Physics of the X-rays

In general, X-rays are from same physical nature as visible light, radiowaves, microwaves, ultraviolet, or infrared. They are all electromagnetic waves, which spread at the speed of light, although with different wavelengths (see Table 1.1).

In the following, the formation of X-rays and their interaction with matter are explained. These principles of physics can be found in many textbooks (see for example [19]).

1.3.1 Formation of X-rays

The formation of X-rays is performed in an X-ray tube in five steps as shown in Fig. 1.3:

1. A high DC voltage U is applied between cathode and anode.
2. The cathode is strongly heated by the voltage U_h, so that the kinetic energy of the heat is transferred to the mobile electrons in the cathode. The electrons are thus in a position to withdraw from the cathode.

Table 1.1 Electromagnetic spectrum [20]

Electromagnetic waves ⟶	Radiowaves	Microwaves	Infrared	Visible light	Ultraviolet	X-rays	Gammarays
Wavelength in (m)	$1 \sim 10^4$	$10^{-3} \sim 1$	$7.7 \cdot 10^{-7} \sim 10^{-3}$	$3.9 \cdot 10^{-7} \sim 7.7 \cdot 10^{-7}$	$10^{-8} \sim 3.9 \cdot 10^{-7}$	$10^{-12} \sim 10^{-8}$	$10^{-14} \sim 10^{-12}$
Frequency in (Hz)	$3 \cdot 10^8 \sim 3 \cdot 10^4$	$3 \cdot 10^{11} \sim 3 \cdot 10^8$	$3.9 \cdot 10^{14} \sim 3 \cdot 10^{11}$	$7.7 \cdot 10^{14} \sim 3.9 \cdot 10^{14}$	$3 \cdot 10^{16} \sim 7.7 \cdot 10^{14}$	$3 \cdot 10^{20} \sim 3 \cdot 10^{16}$	$3 \cdot 10^{22} \sim 3 \cdot 10^{20}$
Energy in (eV)	$1.2 \cdot 10^{-6} \sim 1.2 \cdot 10^{-10}$	$1.2 \cdot 10^{-3} \sim 1.2 \cdot 10^{-6}$	$1.6 \sim 1.2 \cdot 10^{-3}$	$3.2 \sim 1.6$	$1.2 \cdot 10^2 \sim 3.2$	$1.2 \cdot 10^6 \sim 1.2 \cdot 10^2$	$1.2 \cdot 10^8 \sim 1.2 \cdot 10^6$

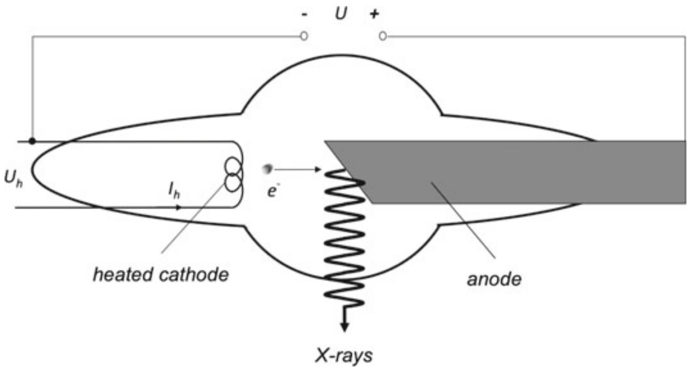


Fig. 1.3 Basic diagram of an X-ray tube

3. The electrons emitted by the hot cathode are accelerated by high-voltage U.
4. These high energy electrons, which are called cathode rays, are incident on the anode.
5. X-rays are produced when electrons of sufficiently high energy incident on the anode are suddenly decelerated.

There is a distinction between discrete and continuous X-rays (commonly known as *Bremsstrahlung*).

Discrete X-rays
These result in transitions of electrons in the inner shells of an atom (see Fig. 1.4a). This happens when a highly accelerated electron e^- ① knocks an electron e_1^- from the atomic shell. Since both electrons leave the atom ②, a hole is formed (where e_1^- was) that is immediately filled by an outer electron (e.g., e_2^-) ③. In an atom, the electrons may be shown only on certain bands with a precisely specified energy level. The deeper the band is in the atom, the greater is the energy of that electron. When jumping from the electron to a lower band (in our example e_2^-) the energy

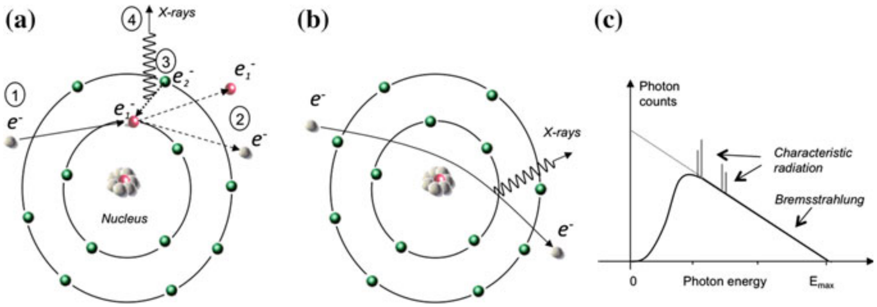


Fig. 1.4 X-ray formation and spectrum. **a** Characteristics X-rays (discrete). **b** Bremsstrahlung (continuous). **c** Spectrum

difference between the two energy levels is emitted as electromagnetic radiation. Energy transitions in the region of the inner electron shells which have high binding energies leads to the emission of X-rays ④. Therefore, the spectrum of this radiation consists of lines at specific wavelengths or energies that are exclusively dependent on the nature of the atom (see Fig. 1.4c). These are called characteristic X-ray lines.

Continuous X-rays (Bremsstrahung)

In addition to the discrete X-rays, there is a continuos radiation called Bremsstrahlung. This occurs when a highly accelerated electron approaches the domain of attraction of the atomic nucleus of the anode and are deflected due to the Coulomb force (Fig. 1.4b). There is no collision between nucleus and electron. Since the electron interacts with the Coulomb force, the direction and velocity of the electron are changed. In this deceleration, the electron loses some or all of the kinetic energy that is emitted in the form of X-rays to the outside. The closer the electron is to the nucleus, the greater is the deceleration and thus the energy of the Bremsstrahlung. As electrons can come close to the nucleus at any distance, this electromagnetic radiation has a continuous spectrum with an upper cutoff frequency $E_{\max}$ (see Fig. 1.4c). The maximum energy is obtained when an electron is completely decelerated, i.e., when the kinetic energy of the electron ($E_{\text{kin}} = e \cdot U$) is converted entirely into photon energy ($E_{\text{photo}} = h \cdot \nu$), where e is the electric charge, U the anode voltage, h the Planck's constant, and ν the frequency of the electromagnetic wave. The smallest possible X-ray wavelength becomes of $E_{\text{kin}} = E_{\text{photo}} (= E_{\max})$ and $c = \lambda\nu$ with:

$$\lambda_{\min} = \frac{h \cdot c}{E_{\max}} = \frac{h \cdot c}{e \cdot U} \tag{1.1}$$

where c is the speed of light in vacuum. Changes to the heating of the cathode I_h (see Fig. 1.3) result in a proportional change of the energy flux density. An increase in the high-voltage U leads to the displacement of the maximum energy flux density to a higher energy.

1.3.2 Scattering and Absorption of X-rays

One aspect particularly important for X-ray testing is the attenuation of the intensity of X-rays when passing through matter. The attenuation is a function of X-ray energy and the material structure of the irradiated material (considerably in terms of density and thickness). The attenuation occurs by two processes: scattering and absorption. The scattering via classical scattering (Rayleigh scattering and Compton effect); and absorption through the photoelectric effect, pair production and partly by the Compton effect. In the following, these are explained as interactions of X-rays with atoms.

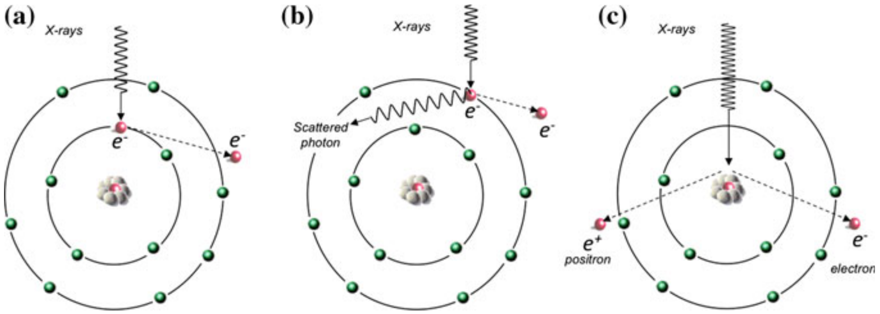


Fig. 1.5 Interaction of X-rays with matter. **a** Photoelectric effect. **b** Compton effect. **c** Pair production

Rayleigh Scattering

In this process, there is a scattering of X-rays from their original path, in which the radiation loses no energy. The lower the energy of the radiation, the more they are deflected from the original path of the rays.

Photo Effect

The photoelectric effect that occurs is likely to happen when the radiation energy just exceeds the binding energy of the electron. In the photoelectric effect, the energy of the incident photon is completely transferred to an electron, and mainly on one of the inner electron shells. The electron takes over the energy that the quantum of radiation it emits as kinetic energy and leaves the atomic union (Fig. 1.5a). This effect increases proportionally to $E^{-3}Z^5$, where E is the energy of the radiation and Z is the atomic number. The photoelectric effect plays a role in the small and medium energies of X-rays.

Compton Effect

In case the radiation energy is very much larger than the binding energy of the atomic electron, the X-ray radiation strikes out the electron from the atom. A portion of the energy of the X-ray radiation is transferred to the electron and converted into kinetic energy. The radiation is scattered and loses energy (see. Fig. 1.5b). This results in a scattering due to the change of direction of the photons at the same time and absorption due to the energy loss. This effect is proportional to the atomic number of the atom Z and inversely proportional to the energy of the radiation to E.

Pair Production

In case the radiation energy is greater than 1.02 MeV and passes it straight into the proximity of the nucleus, the radiation can be turned into matter, producing an electron e^- and e^+ positron (see Fig. 1.5c), whose masses are $m_{e^-} = m_{e^+} =$

511 keV$/c^2$. The pair production is more frequent, the greater the quantum energy and the higher the atomic number of the irradiated material. In cases, where X-rays come from X-ray tubes there is no pair production, as the energy is always in the keV range.

Absorption and scattering can be described mathematically by the X-ray absorption law, which characterizes the intensity distribution of X-rays through matter:

$$\varphi(x) = \varphi_0 e^{-\mu x} \tag{1.2}$$

with φ_0 incident energy flux density, μ absorption coefficient, x thickness of the irradiated matter and φ energy flux density after passage through matter with the thickness of x (see Fig. 1.6a). The absorption coefficient μ depends on the incident photon energy and the density and atomic number of the irradiated material. It is composed of the coefficients of the classical dispersion σ_R, the photoelectric effect τ, the Compton effect σ_C, and the pair production χ:

$$\mu = \sigma_R + \tau + \sigma_C + \chi \tag{1.3}$$

Because of the continuous distribution of the energy of the Bremsstrahlung (see Fig. 1.4c), X-rays contain photons of different energies. In practice, therefore, the course of the absorption curve can only be determined empirically. In the case of aluminum, the course of the absorption coefficient in Fig. 1.7.

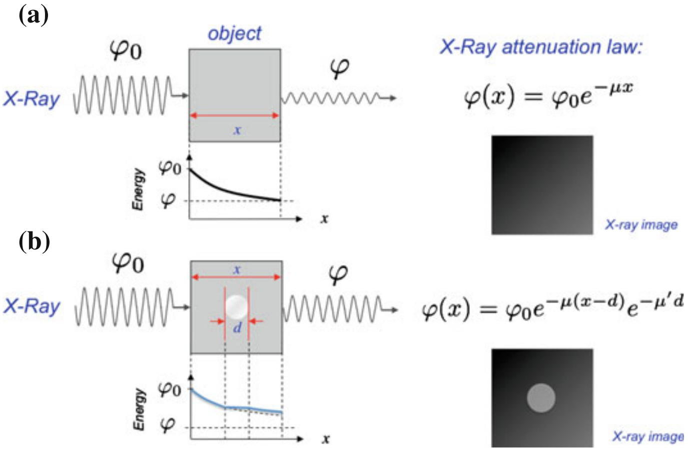


Fig. 1.6 X-ray image formation according to absorption law: **a** X-ray image of a homogenous object, and **b** X-ray image of an object with two different materials

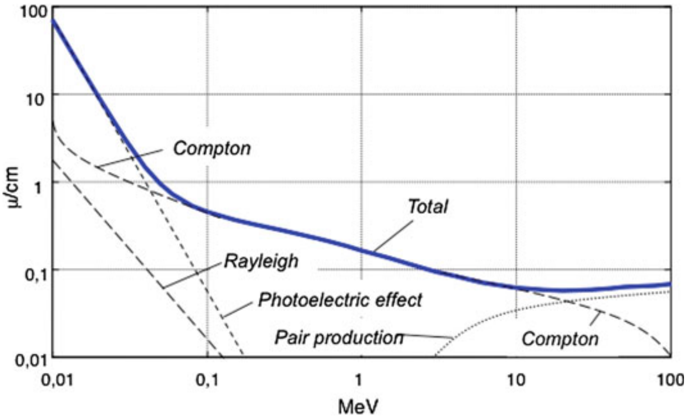


Fig. 1.7 Absorption coefficient for aluminum [20]

1.4 X-ray Testing System

The essential components of an automatic X-ray testing system (see Fig. 1.8), such as X-ray source, manipulator, image acquisition system are explained below.

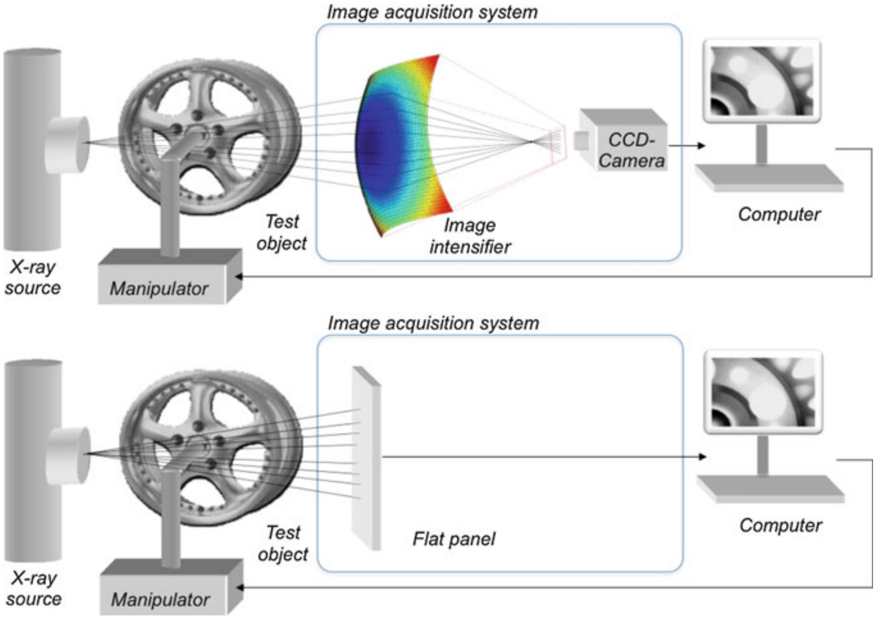


Fig. 1.8 X-ray testing systems. There are two kinds of image acquisition system: based on image intensifiers (*top*) and based on flat panels (*bottom*). In this example, an aluminum wheel is inspected using a manipulator

1.4.1 X-ray Source

There are six requirements for an X-ray source [21]:

1. Adjustable quantum energy.
2. Possible large adjustable dose rate.
3. Intensity of the radiation as uniform as possible in the field of the object to be irradiated.
4. Smallest possible intensity of radiation outside the area to be irradiated.
5. Acceptable price.
6. Long life with constancy of features.

In this section, we describe the essential components of an X-ray source that fulfill the conditions mentioned. An explanation of the formation of X-rays can be found in Sect. 1.3.1.

Hot Cathode

The cathode is made of a filament, from which the electrons emerge through the thermoelectric effect in the vacuum of the X-ray tube. Usually, tungsten (W), also known as wolfram, is used because of its high melting point (about 3380 °C). An influence of the dose rate (independent of the quantum energy of the X-rays) is achieved by controlling the electron emission over the heating current (Figs. 1.3 and 1.9). The quantum energy is adjusted by the high voltage between electrodes. Using an aperture that surrounds the filament, a thin, sharply defined electron beam is generated.

Anode

At the anode surface, the kinetic energy of the cathode beam is converted 99 % into heat and only 1 % into the desired X-rays. To reduce the geometric blur of the imaging process a small focal spot is required. In the focal spot of an X-ray tube, however, so much heat is created that the anode material may melt if the heat is not dissipated quickly and effectively. In order to increase the performance of an X-ray source and at the same time to reduce the focal spot, the anodes are constructed as follows:

Anode Material

The surface layer should be made of materials with a high melting point, high atomic number, and high thermal conductivity. The element tungsten (W) best meets the three criteria. In order to reduce the roughening during the operation, as well as to avoid cracking, it is alloyed with rhenium (Re).

Line Focus

To reduce the optical focus, the electron beam strikes the anode surface in the focal spot inclined by about $\alpha = 7^\circ \sim 20^\circ$ from the vertical axis.

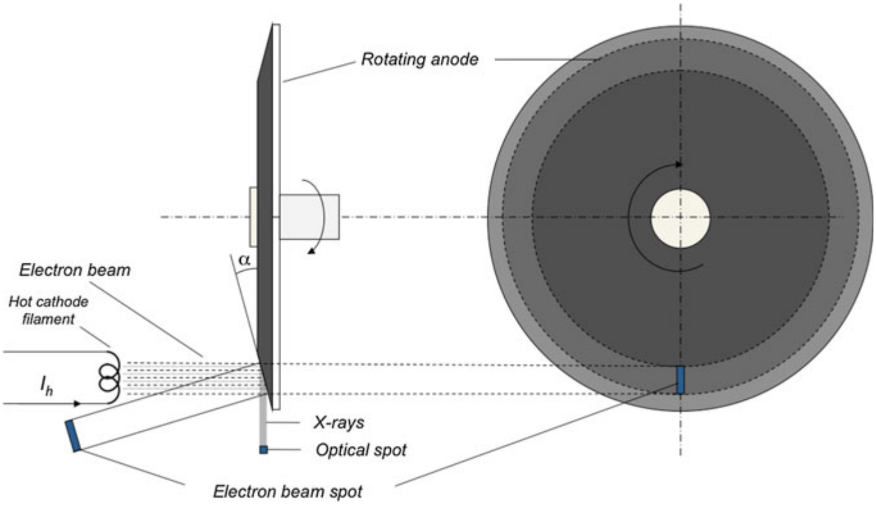


Fig. 1.9 Basic structure of an anode

Rotating Anode

By rotating the anode, the applied heat can be distributed over an entire ring without changing the size of the optical spot (see Fig. 1.9). The distribution of the high thermal load is better the larger the diameter of the ring and the higher the rotation speed.

Envelope

Given that between the electrodes the voltage is high voltage, anode and cathode must be electrically isolated from each other. In addition, the tube envelope forms the vacuum vessel and the mechanical attachment of the tube components. Up until now, glass has been used for this purpose. However, in recent years envelopes made of metal and ceramics have been used.

1.4.2 Manipulator

A manipulator is a device that can be handled with the test objects in the desired manner without the operator using his/her hands to touch [22]. In an X-ray computer vision system, the task of the manipulator is the handling of the test objects. Due to the possibilities of movement, degrees of freedom of the manipulator, the test object can be brought into the desired position. For a manual inspection, the axes of a manipulator are moved by means of one or more joysticks. When an automatic inspection of this task is undertaken, it is handled by a programmable logic controller (PLC) or an industrial computer.

A manipulator consists of sliding elements and rotary elements with which a translation or rotation of the object test can be performed. Previously, the manipulator moved the test object trough the X-ray beam [23]. This solution resulted in a complicated mechanical construction with a high mechanical load, wear and increased maintenance. Today it is possible to move the X-ray tube and the detector that is rigidly connected to it by a *C-arm manipulator*. These manipulators are much easier to control, and are faster and cheaper [16, 17]. An example of such a manipulator is described in Sect. 3.3.4 (see Fig. 3.15).

1.4.3 Image Intensifier

The X-ray image intensifier has two functions: (i) possible lossless conversion of X-ray projection information into a visible image and (ii) its brightness gain [13]. On the basis of the structure of an X-ray image intensifier shown in Fig. 1.10, the operation is explained. The X-ray radiation enters through an input screen into a vacuum tube. As the radiolucent input screen has to withstand the atmospheric pressure, it should not be too thin. Here metals are used with low atomic numbers that are transparent to X-rays, in which the absorption and scattering are relatively small. Thereafter, the radiation incident on the X-ray fluoroscopy screen, in which the conversion of X-radiation into visible light takes place. The X-rays are absorbed and about 2000 photons per X-ray quantum are triggered. The light strikes the photocathode and sets photoelectrons. These electrons are accelerated by approximately 25 kV, which are represented with reduced electron optics on an output phosphor screen. The output image of the image intensifier is then captured by a CCD camera.

The disadvantage of the image intensifier is the geometric distortion due to the curvature of the input screen; details for this can be found in Sect. 3.3.2.

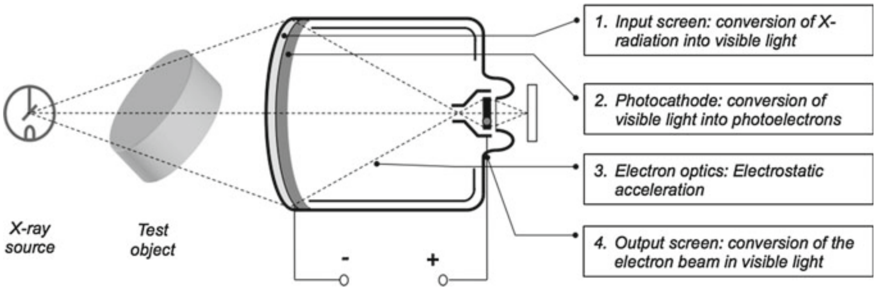

1. Input screen: conversion of X-radiation into visible light
2. Photocathode: conversion of visible light into photoelectrons
3. Electron optics: Electrostatic acceleration
4. Output screen: conversion of the electron beam in visible light
X-ray source
Test object
-
+

Fig. 1.10 Schematic illustration of the operation of an image intensifier

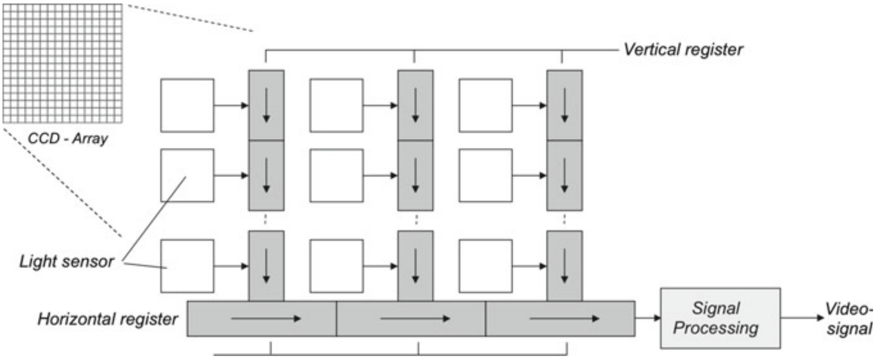


Fig. 1.11 Operation of a CCD-Array

1.4.4 CCD Camera

CCD cameras use solid-state imaging sensors based on CCD (charge-coupled device) arrays. In these imaging sensors, the active detector surface is divided into individual pixels in the CCD-sensor, while incident light is converted and transported into an electrical charge. The principle of the charge transport is based on the charge transfer that takes place in the shift registers (Fig. 1.11).

The CCD cameras are characterized by very good image geometry, high light-sensitivity, and several mega pixels for conventional cameras. In modern days, there are HDTV (High Definition Television) cameras up to 2,200,000 pixels. Furthermore, a CCD camera can achieve a resolution of 24 megapixels and the exposure time can be in a range between seconds and 1/4,000 s.

Due to the low sensitivity of the CCD-image sensor for direct X-ray radiation, the radiation must be converted into visible light. In an X-ray testing system with CCD camera, this conversion happens in the image intensifier (see Sect. 1.4.3).

1.4.5 Flat Panel

A second possible image acquisition system is the flat panel detector based on amorphous silicon (a-Si), in which the X-ray, without going through an image intensifier with CCD camera, is converted from a semiconductor directly into electrical signals (see Fig. 1.12). In this technology a thin view of a-Si is deposited on a glass plate as a support. As in a CCD-chip, a pixel array with switching elements is generated in the silicon layer so that the charge which is stored in the individual pixels can be read out serially and electronically [14].

The advantages of this detector are: larger image receiving surface, no geometric distortion, a high gray-level resolution (12 $\sim$ 16 Bit/Pixel), that is very light and small. Due to the high gray-level resolution and greater imaging surface less test positions are required for the inspection. The low weight allows for easier and faster mechanics [16, 17]. An flat detector is shown in Fig. 1.12.

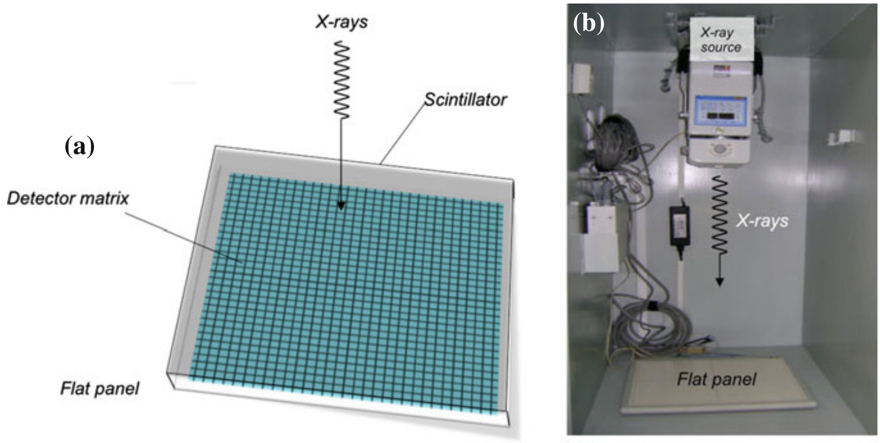


Fig. 1.12 Flatpanel: **a** Basic structure [24] und **b** Example: Canon, model CXDI-50G (resolution: 2208 × 2688 pixels and 4,096 grayscale image). In this example, the X-ray emitter tube is Poskom, model PXM-20BT

1.4.6 Computer

In the context of X-ray testing, a computer system is typically used for the following tasks:

1. To control the image acquisition system.
2. To store acquired X-ray images.
3. To run computer vision algorithms that evaluate X-ray images.
4. To compute statistical analysis.
5. To display results.
6. To control the X-ray source.
7. To control the manipulator.

1.5 X-ray Imaging

In this section we present image formation, acquisition, and visualization.

1.5.1 X-ray Image Formation

In X-ray testing, X-ray radiation is passed through the test object, and a *detector* captures an X-ray image corresponding to the radiation intensity attenuated

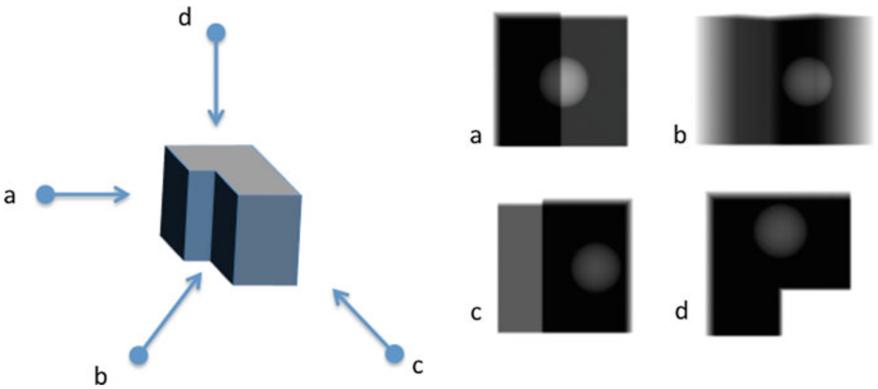


Fig. 1.13 Simulation of an X-ray image of object of Fig. 1.1 from four different points of view. Each *arrow* represents the orientation of the X-ray projection where the beginning corresponds to the X-ray source

by the object.[2] According to the principle of *photoelectric absorption* (1.2): $\varphi = \varphi_0 \exp(-\mu x)$, where the transmitted intensity φ depends on the incident radiation intensity φ_0, the thickness x of the test object, and the energy dependent linear absorption coefficient μ associated with the material, as illustrated in Fig. 1.6.

In a photographic image, the surface of the object is registered. On the contrary, in an X-ray image, the inside of the object is captured. In order to illustrate the formation, we simulate the X-ray image of the object of Fig. 1.1 in several positions (in this example we use the approach outlined in Chap. 7). In this case, we have a homogenous test object with a spherical cavity inside. The result is shown in Fig. 1.13. In this example, we can observe, on the one hand, the absorption phenomenon. The thicker the object the more attenuated the X-rays. In our visualization, bright gray values are used for high output energy (low attenuation), and dark gray values for low output energy (high attenuation). On the other hand, we can see the phenomenon of the summation of shadows, i.e., the output intensity of an image point corresponds to the summation of all the attenuations the X-ray encountered.

It is worth noting that if X-ray radiation passes through n different materials, with absorption coefficients μ_i and thickness x_i, for $i = 1, \ldots, n$, the transmitted intensity φ can be expressed as

$$\varphi = \varphi_0 \exp\left(-\sum_{i=1}^{n} \mu_i x_i\right). \tag{1.4}$$

[2] As explained in Sect. 1.3, X-rays can be *absorbed* or *scattered* by the test object. In this book, we present only the first interaction because scattering is not commonly used for X-ray testing applications covered in this book. For an interesting application based on the X-ray scattering effect, the reader is referred to [25].

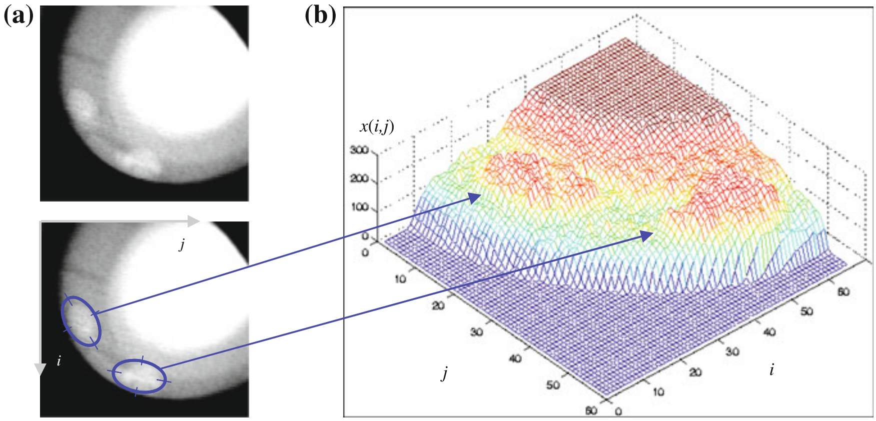


Fig. 1.14 Image formation process: **a** X-ray image of a wheel with two defects, **b** 3D plot of the gray values of the image

This explains the image generation of regions that are present within the test object, as shown in Figs. 1.6 and 1.13, where a gas bubble is clearly detectable. The contrast in the X-ray image between a flaw and a defect-free area of the object test is distinctive. In such X-ray images, we can see that the defects, like voids, cracks, or bubbles, show up as bright features. The reason is that the absorption in these areas is shorter. Hence, according to the principle of differential absorption, the detection of flaws can be achieved automatically using image-processing techniques that are able to identify unexpected regions in a digital X-ray image. A real example is shown in Fig. 1.14 which clearly depicts two defects.

Another example is illustrated in Fig. 1.15a, where a backpack containing a knives and a handgun is shown. However, X-ray images sometimes contain overlapped objects, making it extremely difficult to distinguish them properly, as shown in Fig. 1.15b where a handgun (superimposed onto a laptop) is almost impossible to detect.

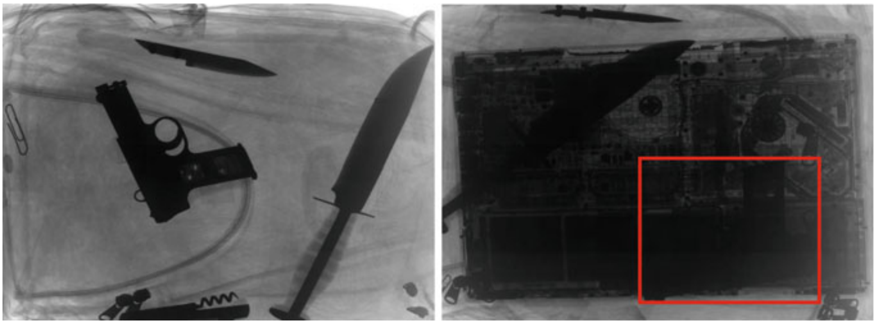


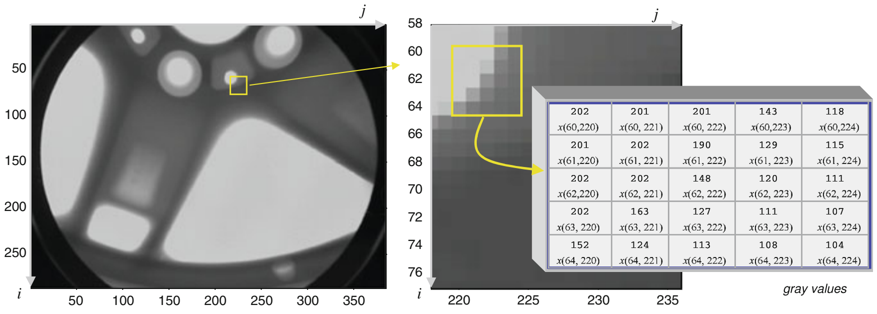
Fig. 1.15 X-ray images of a backpack. *Left* It is easy to recognize a handgun (and two knives). *Right* It is extremely difficult to detect the handgun (see *red rectangle*)

1.5.2 Image Acquisition

In X-ray examination, X-ray radiation is passed through the material under test, and a detector senses the radiation intensity attenuated by the material(s) of the test object. The spacial distribution of the attenuation coefficients of the elements of the object test define the X-ray information that is acquired by the sensor.

The X-ray image is usually captured with a CCD camera (see Sect. 1.4.4) or a flat panel (see Sect. 1.4.5). The digitalized image is stored in a matrix. An example of a digitized X-ray image is illustrated in Fig. 1.16. The size of the image matrix corresponds to the resolution of the image. In this example, the size is 286 × 384 picture elements, or *pixels*. Each pixel has a *gray value* associated. This value is between 0 and 255 for a scale of $2^8 = 256$ gray levels. Here, '0' means 100 % black and a value of '255' corresponds to 100 % white, as illustrated in Fig. 1.17. Typically, the digitized X-ray image is stored in a 2D matrix, e.g., $\mathbf{X}$, and its pixels are arranged in a grid manner. Thus, element $x(i, j)$ denotes the gray value of the ith row of the jth column, pixel (i, j), as shown in the matrix of Fig. 1.16.

The eye is only capable of resolving around 40 gray levels [26]; however, for computer vision applications, gray-level resolution must be a minimum of 256 levels. In some applications, $2^{16} = 65{,}536$ gray levels are used [17], which allows one to evaluate both very dark and very bright regions in the same image.



202 $x(60,220)$	201 $x(60, 221)$	201 $x(60, 222)$	143 $x(60,223)$	118 $x(60,224)$
201 $x(61,220)$	202 $x(61, 221)$	190 $x(61, 222)$	129 $x(61, 223)$	115 $x(61, 224)$
202 $x(62,220)$	202 $x(62, 221)$	148 $x(62, 222)$	120 $x(62, 223)$	111 $x(62, 224)$
202 $x(63, 220)$	163 $x(63, 221)$	127 $x(63, 222)$	111 $x(63, 223)$	107 $x(63, 224)$
152 $x(64, 220)$	124 $x(64, 221)$	113 $x(64, 222)$	108 $x(64, 223)$	104 $x(64, 224)$

Fig. 1.16 Digital X-ray image

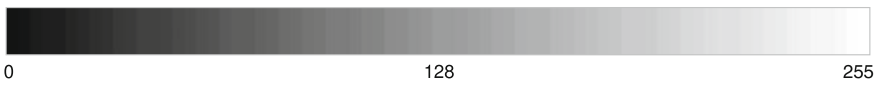


Fig. 1.17 256 gray-level scale

1.5.3 X-ray Image Visualization

In many X-ray testing applications, it is necessary to display X-ray images. For example, when we present a result based on an X-ray image, or when a human evaluation of an X-ray image is required (e.g., baggage screening). In those cases, it is useful to have a suitable visualization of X-ray images.

A simple way to visualize an X-ray image is using a grayscale as shown in Fig. 1.16 that uses the grayscale of Fig. 1.17. Conventionally, X-ray images have been 'black and white' because of the gray nature of the radiographies and fluorescent screens. Usually, a common human eye can distinguish less than 50 gray values [26], however, a trained human eye is able to recognize up to 100 gray values [27].

Nowadays, it is possible to assign colors to grayscale images. With today's computing technology, especially with the ongoing advancements in displays, there is no reason to think that X-ray images must be visualized in grayscale only. In the seventeenth century, Newton said *indeed rays, properly expressed, are not colored* [28]. He was referring to light rays. Now, one can say that X-rays, *properly expressed*, are not gray because they are not visible! We can just find a suitable way to visualize them. Thus, we can use the power of human vision that can distinguish thousands of colors [27].

In order to improve the visualization of an X-ray image, *pseudo coloring* can be used. In pseudo coloring, a gray value is converted into a color value. That is, we need a map function that relates the gray value x with a color value $(R(x), G(x), B(x))$ for red, green and blue respectively if we use a RGB-based color map [29]. Some examples of the color maps are illustrated in Fig. 1.18 in which the transformations $(R(x), G(x), B(x))$ are shown for 'jet', 'hsv', 'parula', 'hot', 'rainbow', and 'sinmap' [27, 29, 30]. An example of a pseudocolored X-ray image is illustrated in Fig. 1.19.

The mentioned transformations correspond to linear mappings that can be loaded from a look-up table. In addition, there are some interesting algebraic or trigonometric transformations that can be used in pseudocoloring [31]. One of them is the 'sin transformation' generally defined as:

$$\begin{aligned} R(x) &= \mid a_R + k_R \sin(\cos(\omega_R x) + \theta_R) \mid \\ G(x) &= \mid a_G + k_G \sin(\cos(\omega_G x) + \theta_G) \mid \\ B(x) &= \mid a_B + k_B \sin(\cos(\omega_B x) + \theta_B) \mid \end{aligned} \tag{1.5}$$

where ω_C, θ_C, k_C and a_C are frequency, phase, amplitude and offset for channel $C = R, G, B$. This color map is implemented in function Xsincolormap (see Appendix B) of 𝕏vis Toolbox.[3] An example of a pseudocolored X-ray image is illustrated in Fig. 1.19 for 'rainbow' and 'sinmap'.

[3] Matlab toolbox used in this book (see Sect. 1.6.4).

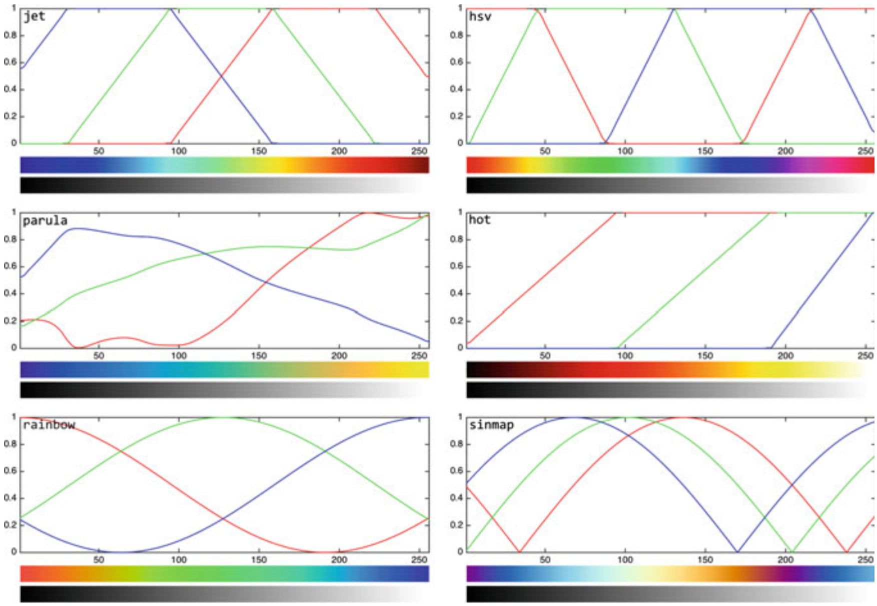


Fig. 1.18 Color maps used in pseudo coloring

Matlab Example 1.1 In Fig. 1.19, we have an X-ray image of a pen case. In this example, we show different visualizations of a small region of this image, namely the pencil sharpener. The example shows the classical grayscale representation, pseudocolors, and a 3D representation:

Listing 1.1 : X-ray image representation.

```
% ImageVisualization.m
close all
I  = Xloadimg('B',2,4);                          % input image
figure(1); imshow(I); title('X-ray image')
hold on
i1 = 250; i2 = 399; j1 = 340; j2 = 529;
plot([j1 j2 j2 j1 j1],[i1 i1 i2 i2 i1],'r');    % crop
J  = I(i1:i2,j1:j2);                             % cropped image
Xcolorimg(J,[],0);                               % color representation
```

The output of this code is in Fig. 1.19. We can see the use of a color map for pseudocolor representations. A very interesting visualization is the 3D representation, where the z-axis corresponds to the gray value, in which the screw of the sharpener is clearly distinguishable. The output of this example is obtained using Xcolorimg (see Appendix B) of 𝕏vis Toolbox. The reader can experiment a different visualization using command Xdyncolor (see Appendix B) of 𝕏vis Toolbox, where a video of an X-ray image is presented. In this video, each frame is displayed using a different colormap that slightly varies from frame to frame. □

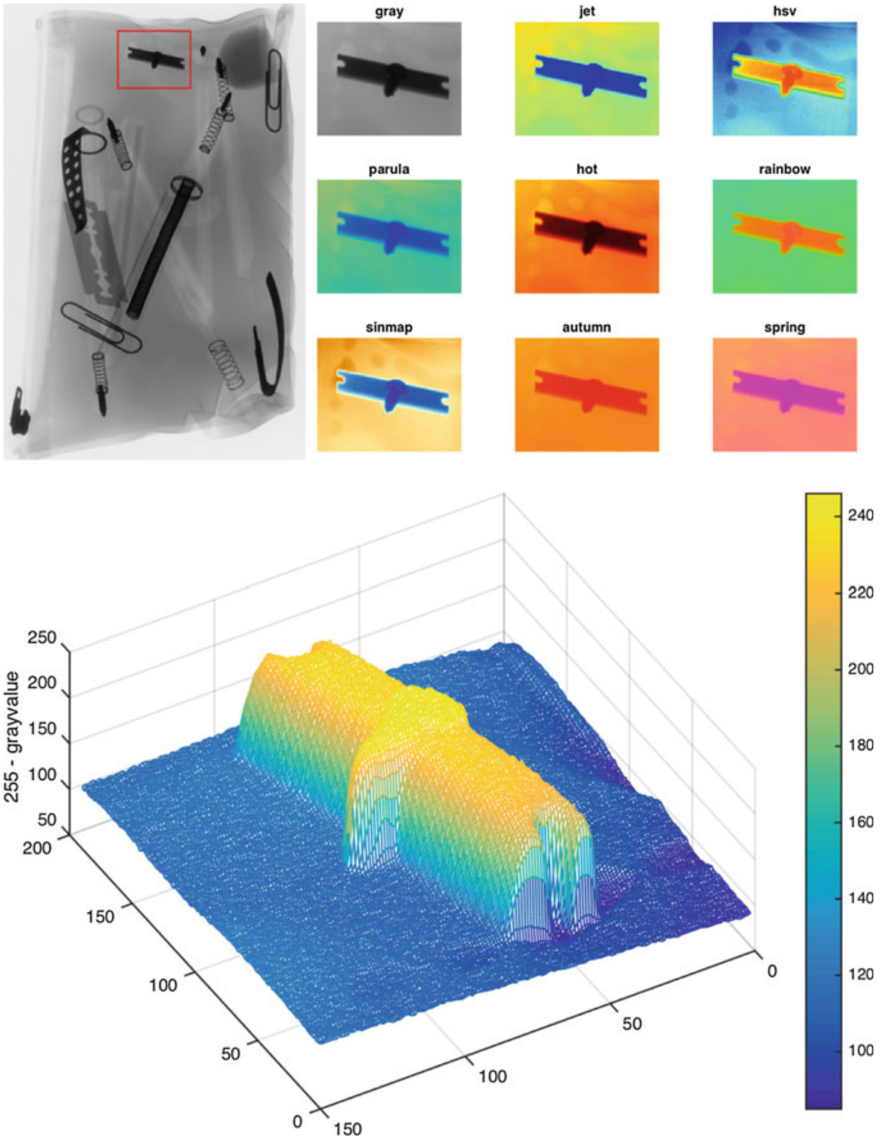


Fig. 1.19 Different visualizations of an X-ray image (→ Example 1.1)

1.5.4 Dual Energy

Coefficient μ in (1.2) can be modeled as $\mu/\rho = \alpha(Z, E)$, where ρ is the density of the material, and $\alpha(Z, E)$ is the mass attenuation coefficient that depends on the atomic number of the material Z, and the energy E of the X-ray photons. The absorption coefficient varies with energy (or wavelength) according to [32]:

$$\frac{\mu}{\rho} = k\lambda^3 Z^3 \tag{1.6}$$

where k is a constant. Values for $\alpha(Z, E)$ are already measured and available in several tables (see [33]). In order to identify the material composition—typically for explosives or drug detection—the atomic number Z cannot be estimated using only one image, as a thin material with a high atomic number can have the same absorption as a thick material with a low atomic number [25]. For this purpose, a *dual-energy* system is used [34], where the object is irradiated with a high energy level E_1 and a low level energy E_2. In the first case, the absorbed energy depends mainly on the density of the material. In the second case, however, the absorbed energy depends primarily on the effective atomic number and the thickness of the material [35].

Using dual energy, it is possible to calculate the ratio

$$R = \log(\varphi_2/\varphi_0)/\log(\varphi_1/\varphi_0), \tag{1.7}$$

where φ_1 and φ_2 are the transmitted intensities φ obtained by (1.2) using energies E_1 and E_2, respectively. Thus, from

$$R = \alpha(Z, E_2)/\alpha(Z, E_1), \tag{1.8}$$

the term $-\rho z$ is canceled out, Z can be directly found using the known measurements $\alpha(Z, E)$ [36]. From both images, a new image is generated using a *fusion model*, usually a look-up table that produces pseudo color information [37, 38], as shown in Fig. 1.20.

Matlab Example 1.2 In Fig. 1.20, we have two X-ray images acquired from the same object at the same position but with different energies: the first one was taken at 5 mA and 70 kV and the second one at 5 mA and 100 kV. For an image generation of dual energy, we can use the following Matlab code:

Listing 1.2 : Dual energy.

```
% DualEnergy.m
close all
X1  = Xloadimg('B',60,1,0);             % low energy image
X2  = Xloadimg('B',60,2,0);             % high energy image
x   = Xloaddata('B',60,'DualEnergyLUT'); % LUT
map = parula;                            % color map
Y   = Xdualenergy(X2,X1,x.LUT,map,1);   % conversion
```

The output of this code is in Fig. 1.20. We can see the use of a color map for pseudo-color representations. The output image is a grayscale image, however, the each gray value is displayed according to a 256 colors palette as shown in right bar. In this example, we use Xdualenergy (see Appendix B) of 𝕏vis Toolbox. □

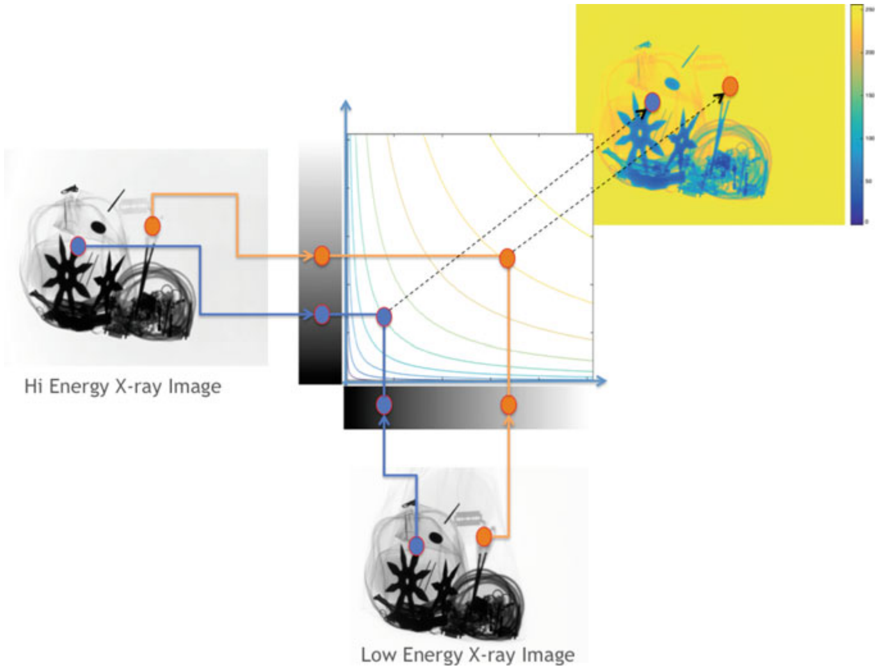


Fig. 1.20 Generation of a pseudocolor image using dual energy. In this example, the colors correspond to different materials (→ Example 1.2)

1.6 Computer Vision

Computer Vision is the science and technology of giving computers the ability to 'see' and 'understand' images taken by one or more cameras. The goal of computer vision is to study and develop algorithms for interpreting the visual world captured in images or videos. Typical topics of computer vision are: detection and recognition, automated visual inspection, image stitching, image processing and analysis (enhancement, filtering, morphological operations, edge detection and segmentation), video processing (optical flow and tracking), recognition of patterns, feature extraction and selection, local descriptors and classification algorithms, and finally, geometric vision topics such as projective geometry, camera geometric model, camera calibration, stereovision, and 3D reconstruction [29, 39–45].

In order to give an introduction to the topics of computer vision that have been used in X-ray testing and will be covered in this book, we follow Fig. 1.21 which illustrates an extended version of our simple model presented in Fig. 1.1.

In this general schema, X-ray images of a test object can be generated at different positions and different energy levels. Depending on the application, each block of this diagram can be (or not be) used. For example, there are applications such as weld inspection that use a segmentation of a single monoenergetic X-ray image

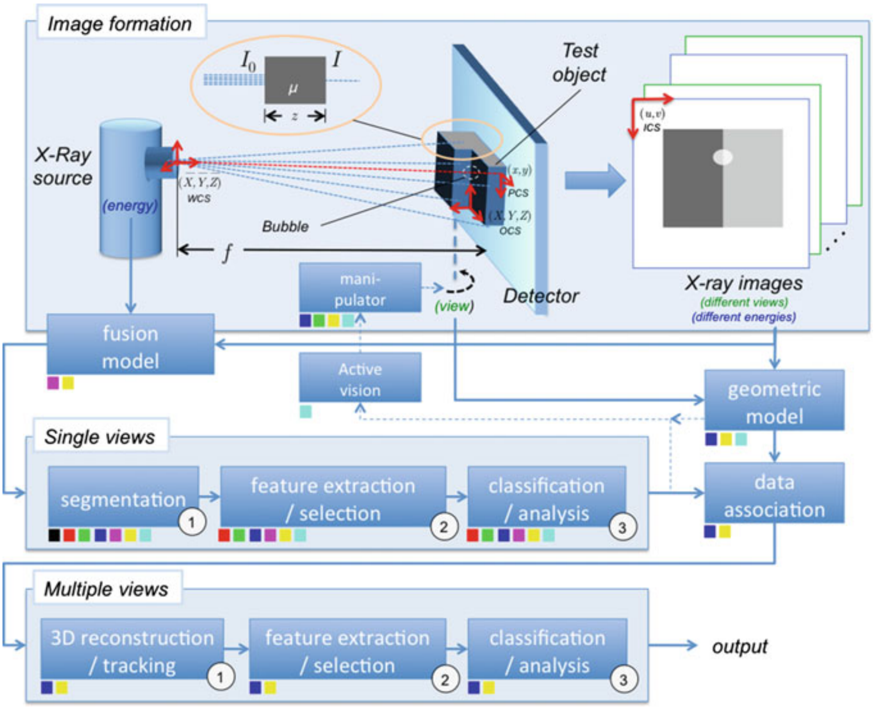


Fig. 1.21 General schema for X-ray testing using computer vision (see text)

(black square), sometimes with pattern recognition approaches (red squares); applications like casting inspection that use monoenergetic multiple views where the decision is taken analyzing individual views (green squares) or corresponding multiple views (blue squares); applications including baggage screening that use dual energy of single views (magenta squares) and multiple views (yellow squares); and finally, applications for cargo inspections that employ active vision where a next-best view is set according to the information of a single view (cyan squares). In each case, the blocks without the corresponding color square are not used.

1.6.1 Geometric Model

The X-ray image of a test object corresponds to a projection in perspective, where a 3D point of the test object is viewed as a pixel in the digital X-ray image, as illustrated in Fig. 1.21. A geometric model that describes this projection can be highly useful for 3D reconstruction and for data association between different views of the same object. Thus, 3D features or multiple view 2D features can be used to improve the diagnosis performed by using a single view.

As we will learn in Chap. 3, for the geometric model, four coordinate systems are used (see Fig. 1.21):

- OCS (X, Y, Z): Object Coordinate System, where a 3D point is defined using coordinates attached to the test object.
- WCS $(\bar{X}, \bar{Y}, \bar{Z})$: World Coordinate System, where the origin corresponds to the optical center (X-ray source) and the $\bar{Z}$ axis is perpendicular to the projection plane of the detector.
- PCS (x, y): Projection Coordinate System, where the 3D point is projected into the projection plane $\bar{Z} = f$, and the origin is the intersection of this plane with $\bar{Z}$ axis.
- ICS (u, v): Image Coordinate System, where a projected point is viewed in the image. In this case, (x, y)-axes are set to be parallel to (u, v)-axes.

The geometric model OCS → ICS, i.e., transformation $\mathbf{P} : (X, Y, Z) \rightarrow (u, v)$, can be expressed in homogeneous coordinates as [46]:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{1.9}$$

where λ is a scale factor and $\mathbf{P}$ is a 3×4 matrix modeled as three transformations:

(i) OCS → WCS, i.e., transformation $\mathbf{T}_1 : (X, Y, Z) \rightarrow (\bar{X}, \bar{Y}, \bar{Z})$, using a 3D rotation matrix $\mathbf{R}$, and 3D translation vector $\mathbf{t}$;
(ii) WCS → (PCS), i.e., transformation $\mathbf{T}_2 : (\bar{X}, \bar{Y}, \bar{Z}) \rightarrow (x, y)$, using a perspective projection matrix that depends on focal distance f; and
(iii) PCS → ICS, i.e., transformation $\mathbf{T}_3 : (x, y) \rightarrow (u, v)$, using scales factor α_x and α_y, and 2D translation vector (u_0, v_0).

The three transformations OCS → WCS → PCS → ICS are expressed as

$$\mathbf{P} = \underbrace{\begin{bmatrix} \alpha_x & 0 & u_0 \\ 0 & \alpha_x & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{T}_3} \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{T}_2} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}}_{\mathbf{T}_1} \tag{1.10}$$

The parameters included in matrix $\mathbf{P}$ can be estimated using a calibration approach [42].

In order to obtain multiple views of the object, n different projections of the test object can be achieved by rotating and translating it (for this task a manipulator can be used). For the pth projection, for $p = 1 \dots n$, the geometric model $\mathbf{P}_p$ used in (1.9) is computed from (1.10) including 3D rotation matrix $\mathbf{R}_p$ and 3D translation $\mathbf{t}_p$. Matrices $\mathbf{P}_p$ can be estimated using a calibration object projected in the n different positions [46] or using a bundle adjustment algorithm where the geometric model is obtained from the n X-ray images of the test object [47].

1.6.2 Single View Analysis

A computer vision system for single view analysis, as shown in Fig. 1.21, consists typically of the following steps: an X-ray image of the test object is taken and stored on a computer. The digital image is improved in order to enhance the details. The X-ray image of the parts of interest is found and isolated from the background of the scene. Significant features of the segmented parts are extracted. Selected features are classified or analyzed in order to determine if the test object deviates from a given set of specifications. Using a supervised pattern recognition methodology, the selection of the features and the training of the classifier are performed using representative images that are to be labeled by experts [44]. In this book, we will cover several techniques of image processing (Chap. 4), image representation (Chap. 5), and classification (Chap. 6) that have been in X-ray testing.

For the segmentation task, two general approaches can be used: a traditional image segmentation or a *sliding-window* approach. In the first case, image-processing algorithms are used (e.g., histograms, edge detection, morphological operations, filtering, etc. [29]). Nevertheless, inherent limitations of traditional segmentation algorithms for complex tasks and increasing computational power have fostered the emergence of an alternative approach based on the so-called *sliding-window* paradigm. Sliding-window approaches have established themselves as state-of-art in computer vision problems where a visually complex object must be separated from the background (see, for example, successful applications in face detection [48] and human detection [49]). In the sliding-window approach, a detection window is moved over an input image in both horizontal and vertical directions, and for each localization of the detection window, a classifier decides to which class the corresponding portion of the image belongs according to its features. Here, a set of candidate image areas are selected and all of them are fed to the subsequent parts of the image analysis algorithm. This resembles a brute force approach where the algorithm explores a large set of possible segmentations, and at the end the most suitable is selected by the classification steps. An example for weld inspection using sliding-windows can be found in Chap. 8.

1.6.3 Multiple View Analysis

It is well known that *A picture is worth a thousand words*, however, this is not always true if we have an *intricate* image as illustrated in Fig. 1.15b. In certain X-ray applications, e.g., baggage inspection, there are usually *intricate* X-ray images due to overlapping parts inside the test object, where each pixel corresponds to the attenuation of multiple parts, as expressed in (1.4).

In some cases, *active vision* can be used in order to adequate the viewpoint of the test object to obtain more suitable X-ray images to analyze. Therefore, an algorithm is designed for guiding the manipulator of the X-ray imaging system to poses where the detection performance should be higher [50] (see Fig. 1.21).

In other cases, multiple view analysis can be a powerful option for examining complex objects where uncertainty can lead to misinterpretation. Multiple view analysis offers advantages not only in 3D interpretation. Two or more images of the same object taken from different points of view can be used to confirm and improve the diagnosis undertaken by analyzing only one image. In the computer vision community, there are many important contributions in multiple view analysis (e.g., object class detection [51], motion segmentation [52], simultaneous localization and mapping (SLAM) [53], 3D reconstruction [54], people tracking [55], breast cancer detection [56], and quality control [57]). In these fields, the use of multiple view information yields a significant improvement in performance.

Multiple view analysis in X-ray testing can be used to achieve two main goals: (i) analysis of 2D corresponding features across the multiple views, and (ii) analysis of 3D features obtained from a 3D reconstruction approach. In both cases, the attempt is made to gain relevant information about the test object. For instance, in order to validate a single view detection—filtering out false alarms—2D corresponding features can be analyzed [58]. On the other hand, if the geometric dimension of a inner part must be measured a 3D reconstruction needs to be performed [59].

As illustrated in Fig. 1.21, the input of the multiple view analysis is the *associated data*, i.e., corresponding points (or patches) across the multiple views. To this end, associated 2D cues are found using geometric constraints (e.g., epipolar geometry and multifocal tensors [42, 60]), and local scale-invariant descriptors across multiple views (e.g., like SIFT [61]).

Finally, 2D or 3D features of the associated data can be extracted and selected, and a classifier can be trained using the same pattern recognition methodology explained in Sect. 1.6.2.

Depending on the application, the output could be a measurement (e.g., the volume of the inspected inner part is 3.4 cm^3), a class (e.g., the test object is defective), or an interpretation (e.g., the baggage should be inspected by a human operator given that uncertainty is high).

1.6.4 𝕏VIS *Toolbox*

In this book, we use many commands of 𝕏VIS Toolbox, i.e., a Matlab Toolbox that we developed for X-ray testing with computer vision.[4] 𝕏VIS Toolbox contains approximately 150 functions for image processing, feature extraction, feature transformation, feature analysis, feature selection, data selection and generation, classification, clustering, performance evaluation, multiple view analysis, image sequence processing, and tracking with geometrical constraints.

[4] 𝕏VIS Toolbox is available on: http://dmery.ing.puc.cl/index.php/book/.

Commands of 𝕏vis Toolbox starts with letter ‘X’. For example Ximmedian (see Appendix B) corresponds to the implemented function of 𝕏vis Toolbox for median filtering.

Each function of 𝕏vis Toolbox has a ‘help’ with one or more examples. For example, this is the help for Ximmedian (see Appendix B):

Listing 1.3 : Help of command Ximmedian of 𝕏vis Toolbox.

```
% J = Ximmedian(I,k)
%
% Toolbox Xvis: Median filtering of image I.
%
%    Input data:
%       I grayvalue image.
%
%    Output:
%       J: filtered image using median mask of k x k pixels.
%
%    Example:
%       I = imread('circuit.tif');
%       figure(1)
%       imshow(I,[]); title('original image')
%       J = Ximmedian(I,7);
%       figure(2)
%       imshow(J,[]); title('transformed image')
```

In addition, as we will see in this book, 𝕏vis Toolbox includes a general framework that designs a computer vision system automatically in a few lines of code, or using two powerful graphic user interfaces one for feature extraction (Fig. 5.28) and one for feature and classifier selection (Fig. 6.20). The interface automatically finds the features and the classifiers for a given visual task avoiding the classical trial and error framework commonly used by human designers.

A quick reference for 𝕏vis Toolbox can be found in Appendix B.

1.6.5 𝔾𝔻𝕏ray Database

We developed an X-ray database that contains more than 19,400 X-ray images.[5] The database is described in detail in Chap. 2 and Appendix A. The database includes five groups of X-ray images: castings, welds, baggage, natural objects, and settings. Each group has several series, and each series several X-ray images.

Most of the series are annotated or labeled. In those cases, the coordinates of the bounding boxes of the objects of interest or the labels of the images are available in standard text files. The size of 𝔾𝔻𝕏ray is 3.5 GB.

[5] 𝔾𝔻𝕏ray is available on: http://dmery.ing.puc.cl/index.php/material/gdxray/.

1.7 Summary

In this book, we present a general overview of computer vision approaches that have been used in X-ray testing. In this chapter, we gave an introduction to our book by covering relevant issues of X-ray testing.

X-ray testing has been developed for the inspection of materials or objects, where the aim is to analyze—nondestructively—those inner parts that are undetectable to the naked eye. Thus, X-ray testing is used to determine if a test object deviates from a given set of specifications.

Typical applications are:

- Inspection of automotive parts
- Quality control of welds
- Baggage screening
- Analysis of food products
- Inspection of cargo
- Quality control of electronic circuits

In order to achieve efficient and effective X-ray testing, automated and semiautomated systems based on computer vision algorithms are being developed to execute this task.

We gave an introduction to some physic and geometric principles related to computer vision. Following which, an overview of single and multiple view analysis was presented.

References

1. Röntgen, W.: Eine neue Art von Strahlen: I Mitteilung. In: Sitzungsbericht der Würzburger Physikal.-Medicin. Gesellschaft. Verlag und Druck der Stahel'schen K. Hof- und Universitäts-Buch- und Kunsthandlung. Würzburg (1895)
2. Hellier, C.: Handbook of Nondestructive Evaluation, 2nd edn. McGraw Hill, New York (2013)
3. Goebbels, J.: Handbook of technical diagnostics, chap. Computed Tomography. Springer, Berlin (2013)
4. Rowlands, J.: The physics of computed radiography. Phys. Med. Biol. **47**(23), R123 (2002)
5. Mery, D., Jaeger, T., Filbert, D.: A review of methods for automated recognition of casting defects. Insight **44**(7), 428–436 (2002)
6. Hanke, R., Fuchs, T., Uhlmann, N.: X-ray based methods for non-destructive testing and material characterization. Nucl. Instrum. Methods Phys. Res. Sect. A: Accel. Spectrom. Detect. Assoc. Equip. **591**(1), 14–18 (2008)
7. Purschke, M.: IQI-sensitivity and applications of flat panel detectors and X-ray image intensifiers—a comparison. Insight **44**(10), 628–630 (2002)
8. Lossau, N.: Röntgen: Eine Entdeckung verändert unser Leben, 1st edn. Köln, vgs (1995)
9. Richter, H.U.: Chronik der Zerstörungsfreien Materialprüfung, 1st edn. Deutsche Gesellschaft für Zerstörungsfreie Prüfung DGZfP, Verlag für Schweißen und verwendete Verfahren, DVS-Verlag GmbH, Berlin, (1999)
10. Schaefer, M.: 100 Jahre Röntgenprüftechnik - Prüfsysteme früher und heute. In: DGZfP Jahrestagung, pp. 13–26. Deutsche Gesellschaft für Zerstörungsfreie Prüfung e.V., Aachen (1995)

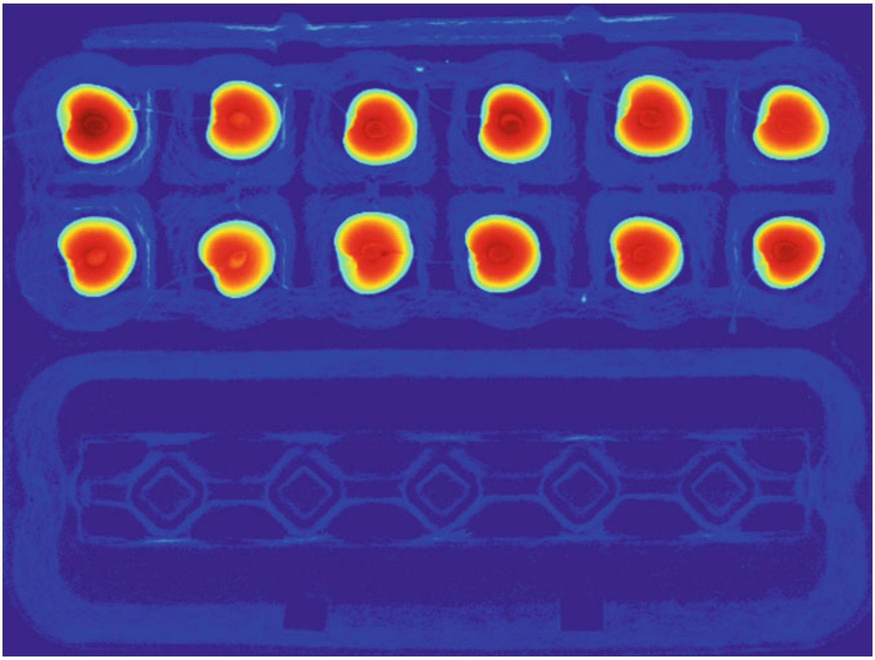
11. Purschke, M.: Radioskopie – Die Prüftechnik der Zukunft? In: DGZfP Jahrestagung, vol. Berichtsband 68.1, pp. 77–84. Deutsche Gesellschaft für Zerstörungsfreie Prüfung e.V., Celle (1999)
12. Völkel: Grundlagen für den Prüfer mit Röntgen- und Gammastrahlung (Durchstrahlungsprüfung). Amt für Standarisierung, Meßwesen und Warenprüfung, Fachgebiet Zerstörungsfreie Werkstoffprüfung (1989)
13. Heinzerling, J.: Bildverstärker-Fernseh-Kette. In: Ewen, K. (ed.) Moderne Bildgebung: Physik, Gerätetechnik, Bildbearbeitung und -kommunikation, Strahlenschutz, Qualitätskontrolle, pp. 115–126. Georg Thieme Verlag, Stuttgart (1998)
14. Bunke, J.: Computertomographie. In: Ewen, K. (ed.) Moderne Bildgebung: Physik, Gerätetechnik, Bildbearbeitung und -kommunikation, Strahlenschutz, Qualitätskontrolle, pp. 153–170. Georg Thieme Verlag, Stuttgart (1998)
15. Jaeger, T.: Optimierungsansätze zur Lösung des *limited data problem* in der Computertomographie. Verlag Dr. Köster, Berlin (1997)
16. Bavendiek, K., Krause, A., Beyer, A.: Durchsatzerhöhung in der industriellen Röntgenprüfung – Eine Kombination aus innovativem Prüfablauf und optimierter Bildauswertung. In: DGZfP Jahrestagung, vol. Berichtsband 63.1, pp. 301–306. Deutsche Gesellschaft für Zerstörungsfreie Prüfung e.V., Bamberg (1998)
17. Jaeger, T., Heike, U., Bavendiek, K.: Experiences with an amorphous silicon array detector in an ADR application. In: International Computerized Tomography for Industrial Applications and Image Processing in Radiology, DGZfP Proceedings BB 67-CD, pp. 111–114. Berlin (1999)
18. Szeles, C., Soldner, S.A., Vydrin, S., Graves, J., Bale, D.S.: CdZnTe semiconductor detectors for spectroscopic X-ray imaging. IEEE Trans. Nucl. Sci. **55**(1), 572–582 (2008)
19. Als-Neielsen, J., McMorrow, D.: Elements of Modern X-ray Physics, 2nd edn. Wiley, West Sussex (2011)
20. Kuchling, H.: Taschenbuch der Physik, 12th edn. Harri Deutsch, Thun-Frankfurt, Main (1989)
21. Heinzerling, J.: Röntgenstrahler. In: Ewen, K. (ed.) Moderne Bildgebung: Physik, Gerätetechnik, Bildbearbeitung und -kommunikation, Strahlenschutz, Qualitätskontrolle, pp. 77–85. Georg Thieme Verlag, Stuttgart (1998)
22. Schwieger, R.: Stillegung, sicherer Einschluß und Abbau kerntechnischer Anlagen. Institut für Werkstoffkunde, Universität Hannover, Technischer Bericht (1999)
23. Kosanetzky, J.M., Krüger, R.: Philips MU231: Räderprüfanlage. Technischer Bericht, Philips Industrial X-ray GmbH, Hamburg (1997)
24. Horbaschek, H.: Technologie und Einsatz von Festkörperdetektoren in der Röntgentechnik (1998). Vortrag der Firma Siemens Pforchheim in der 9. Sitzung des Unterausschusses Bildverarbeitung in der Durchstrhlungprüfung (UA BDS) der Deutschen Gesellschaft für Zerstörungsfreie Prüfung e.V. (DGZfP), Ahrensburg
25. Wells, K., Bradley, D.: A review of X-ray explosives detection techniques for checked baggage. Appl. Radiat. Isot. **70**(8), 1729–1746 (2012)
26. Castleman, K.: Digital Image Processing. Prentice-Hall, Englewood Cliffs (1996)
27. Neri, E., Caramella, D., Bartolozzi, C.: Image processing in radiology. In: Baert, A.L., Knauth, M., Sartor, K. (eds.) Medical Radiology. Diagnostic Imaging. Springer, Berlin (2008)
28. Agoston, G.A.: The concept of color. Color Theory and Its Application in Art and Design, pp. 5–10. Springer, New York (1987)
29. Gonzalez, R., Woods, R.: Digital Image Processing, 3rd edn. Pearson Prentice Hall, Upper Saddle River (2008)
30. MathWorks: Image Processing Toolbox for Use with MATLAB: User's Guide. The MathWorks Inc. (2014)
31. Abidi, B.R., Zheng, Y., Gribok, A.V., Abidi, M.A.: Improving weapon detection in single energy X-ray images through pseudocoloring. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. **36**(6), 784–796 (2006)
32. Cullity, B.D., Stock, S.R.: Elements of X-ray Diffraction. Pearson, New Jersey (2001)

33. Hubbell, J., Seltzer, S.: Tables of X-ray mass attenuation coefficients and mass energy-absorption coefficients from 1 keV to 20 MeV for elements Z = 1 to 92 and 48 additional substances of dosimetric interest. http://www.nist.gov/pml/data/xraycoef/index.cfm (1996)
34. Rebuffel, V., Dinten, J.M.: Dual-energy X-ray imaging: benefits and limits. Insight-Non-Destr. Test. Cond. Monit. **49**(10), 589–594 (2007)
35. Singh, S., Singh, M.: Explosives detection systems (EDS) for aviation security. Signal Process. **83**(1), 31–55 (2003)
36. Heitz, G., Chechik, G.: Object separation in X-ray image sets. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2010), pp. 2093–2100 (2010)
37. Franzel, T., Schmidt, U., Roth, S.: Object detection in multi-view x-ray images. Pattern Recognit. pp. 144–154 (2012)
38. Baştan, M., Yousefi, M.R., Breuel, T.M.: Visual words on baggage X-ray images. Computer Analysis of Images and Patterns, pp. 360–368. Springer, Berlin (2011)
39. Klette, R.: Concise Computer Vision: An Introduction into Theory and Algorithms. Springer Science & Business Media. Springer, New York (2014)
40. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer, New York (2011)
41. Forsyth, D.A., Ponce, J.: Computer Vision: A Modern Approach. Prentice Hall, Upper Saddle River (2003)
42. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2003)
43. Faugeras, O., Luong, Q.T., Papadopoulo, T.: The Geometry of Multiple Images: The Laws that Govern the Formation of Multiple Images of a Scene and Some of Their Applications. The MIT Press, Cambridge (2001)
44. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley, New York (2001)
45. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Berlin (2006)
46. Mery, D.: Explicit geometric model of a radioscopic imaging system. NDT & E Int. **36**(8), 587–599 (2003)
47. Mery, D.: Automated detection in complex objects using a tracking algorithm in multiple X-ray views. In: Proceedings of the 8th IEEE Workshop on Object Tracking and Classification Beyond the Visible Spectrum (OTCBVS 2011), in Conjunction with CVPR 2011, pp. 41–48. Colorado Springs (2011)
48. Viola, P., Jones, M.: Robust real-time object detection. Int. J. Comput. Vis. **57**(2), 137–154 (2004)
49. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. Conf. Comput. Vis. Pattern Recognit. (CVPR2005) **1**, 886–893 (2005)
50. Riffo, V., Mery, D.: Active X-ray testing of complex objects. Insight **54**(1), 28–35 (2012)
51. Su, H., Sun, M., Fei-Fei, L., Savarese, S.: Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In: International Conference on Computer Vision (ICCV2009) (2009)
52. Zografos, V., Nordberg, K., Ellis, L.: Sparse motion segmentation using multiple six-point consistencies. In: Proceedings of the Asian Conference on Computer Vision (ACCV2010) (2010)
53. Konolige, K., Agrawal, M.: FrameSLAM: from bundle adjustment to realtime visual mapping. IEEE Trans. Robot. **24**(5), 1066–1077 (2008)
54. Agarwal, S., Snavely, N., Simon, I., Seitz, S., Szeliski, R.: Building Rome in a day. In: IEEE 12th International Conference on Computer Vision (ICCV2009), pp. 72–79 (2009)
55. Eshel, R., Moses, Y.: Tracking in a dense crowd using multiple cameras. Int. J. Comput. Vis. **88**, 129–143 (2010)
56. Teubl, J., Bischof, H.: Comparison of multiple view strategies to reduce false positives in breast imaging. Digital Mammogram. pp. 537–544 (2010)
57. Carrasco, M., Pizarro, L., Mery, D.: Visual inspection of glass bottlenecks by multiple-view analysis. Int. J. Comput. Integr. Manuf. **23**(10), 925–941 (2010)
58. Mery, D., Filbert, D.: Automated flaw detection in aluminum castings based on the tracking of potential defects in a radioscopic image sequence. IEEE Trans. Robot. Autom. **18**(6), 890–901 (2002)

59. Noble, A., Gupta, R., Mundy, J., Schmitz, A., Hartley, R.: High precision X-ray stereo for automated 3D CAD-based inspection. IEEE Trans. Robot. Autom. **14**(2), 292–302 (1998)
60. Mery, D.: Exploiting multiple view geometry in X-ray testing: part I, theory. Mater. Eval. **61**(11), 1226–1233 (2003)
61. Lowe, D.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)

Chapter 2
Images for X-ray Testing

Abstract In this chapter, we present the dataset that is used in this book to illustrate and test several methods. The database consists of 19,407 X-ray images. The images are organized in a public database called 𝔾𝔻𝕏ray that can be used free of charge, but for research and educational purposes only. The database includes five groups of X-ray images: castings, welds, baggage, natural objects, and settings. Each group has several series, and each series several X-ray images. Most of the series are annotated or labeled. In such cases, the coordinates of the bounding boxes of the objects of interest or the labels of the images are available in standard text files. The size of 𝔾𝔻𝕏ray is 3.5 GB and it can be downloaded from our website.



Cover image: *X-ray image of cherries in an egg crate (X-ray image* `N0006_0027` *colored with 'jet' colormap).*

D. Mery, *Computer Vision for X-Ray Testing*, DOI 10.1007/978-3-319-20747-6_2

2.1 Introduction

Public databases of X-ray images can be found for medical imaging,[1] however, to the best knowledge of the author, up until now there have not been any public databases of digital X-ray images for X-ray testing.[2]

As a service to the X-ray testing community, we collected more than 19,400 X-ray images for the development, testing and evaluation of image analysis and computer vision algorithms. The images are organized in a public database called $\mathbb{GDX}$ray.[3] In order to illustrate our database, a random selection of 120 X-ray images is shown in Fig. 2.1. The database includes five groups of X-ray images: castings, welds, baggage, natural objects, and settings. Each group has several series, and each series several X-ray images. Some samples of each series are illustrated in Fig. 2.2. Most of the series are annotated or labeled. In those cases, the coordinates of the bounding boxes of the objects of interest or the labels of the images are available. In Table 2.1, we can see some statistics. The size of $\mathbb{GDX}$ray is 3.49 GB and it can be downloaded from our website (see Fig. 2.2).

In this chapter, we will view the structure of $\mathbb{GDX}$ray database, a description for each group (with some series examples), some examples of applications that have been published using images of $\mathbb{GDX}$ray and some examples in Matlab that can be used to manipulate the database. More details about $\mathbb{GDX}$ray are given in Appendix A.

2.2 Structure of the Database

$\mathbb{GDX}$ray is available in a public repository. The repository contains five group folders one for each group: `Castings`, `Welds`, `Baggage`, `Nature`, and `Settings`. For each group we define an initial: `C`, `W`, `B`, `N`, and `S`, respectively. As shown in Table 2.1, each group has several series. Each series is stored in an individual subfolder of the corresponding group folder. The subfolder name is `Xssss`, where `X` is the initial of the group and `ssss` is the number of the series. For example, the third series of group Castings is stored in subfolder `C0003` of folder `Castings` (see more examples in Fig. 2.2). The X-ray images of a series are stored in file `Xssss_nnnn.png`. Again

[1] See for example a good collection in http://www.via.cornell.edu/databases/.

[2] There are some galleries of X-ray images available on the web with a few samples, see for instance http://www.vidisco.com/ndt_solutions/ndt_info_center/ndt_x_ray_gallery with approximately 50 X-ray images.

[3] Available on http://dmery.ing.puc.cl/index.php/material/gdxray. The name comes originally from 'The Grima X-ray database' (Grima is the name of our Machine Intelligence Group at the Department of Computer Science of the Pontificia Universidad Católica de Chile http://grima.ing.puc.cl). The X-ray images included in $\mathbb{GDX}$ray can be used free of charge, but for research and educational purposes only. Redistribution and commercial use is prohibited. Any researcher reporting results which use this database should acknowledge the $\mathbb{GDX}$ray database by citing [1].

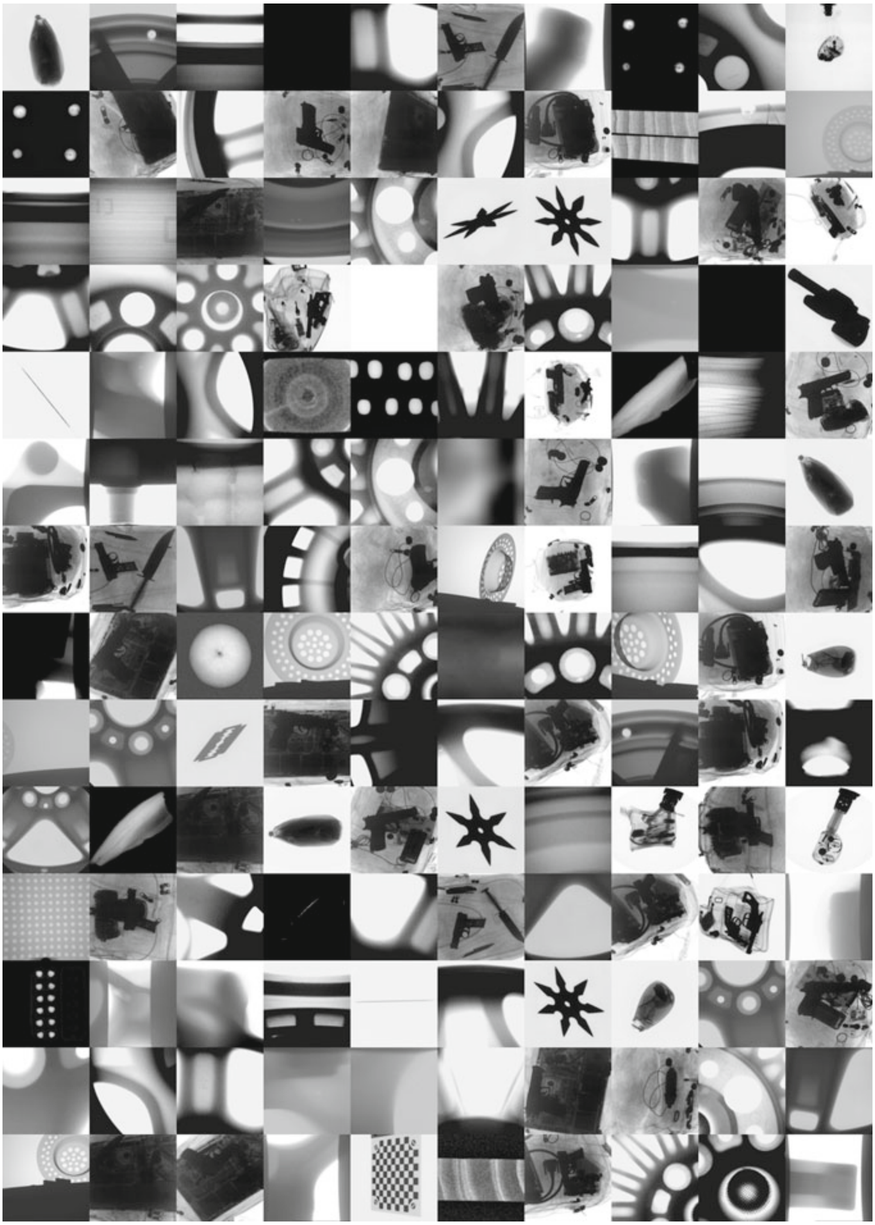
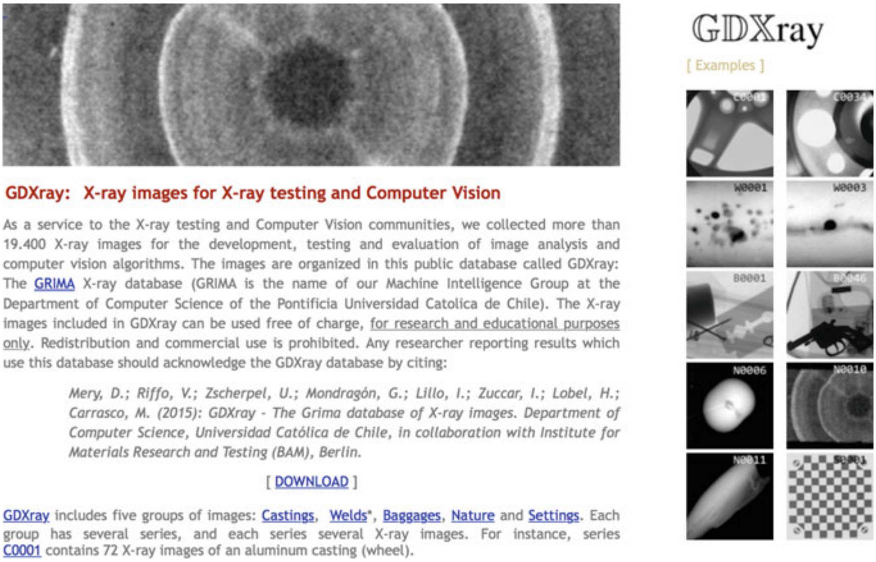


Fig. 2.1 Random X-ray images of 𝔾𝔻𝕏ray database

`Xssss` is the name of the series. The number `nnnn` corresponds to the number of the X-ray image of this series. For example, the fifth X-ray image of series `C0003` is `C0003_0005.png` and is stored in folder `Castings/C0003`. The whole



GDXray: X-ray images for X-ray testing and Computer Vision

As a service to the X-ray testing and Computer Vision communities, we collected more than 19.400 X-ray images for the development, testing and evaluation of image analysis and computer vision algorithms. The images are organized in this public database called GDXray: The GRIMA X-ray database (GRIMA is the name of our Machine Intelligence Group at the Department of Computer Science of the Pontificia Universidad Catolica de Chile). The X-ray images included in GDXray can be used free of charge, for research and educational purposes only. Redistribution and commercial use is prohibited. Any researcher reporting results which use this database should acknowledge the GDXray database by citing:

> *Mery, D.; Riffo, V.; Zscherpel, U.; Mondragón, G.; Lillo, I.; Zuccar, I.; Lobel, H.; Carrasco, M. (2015): GDXray - The Grima database of X-ray images. Department of Computer Science, Universidad Católica de Chile, in collaboration with Institute for Materials Research and Testing (BAM), Berlin.*

[DOWNLOAD]

GDXray includes five groups of images: Castings, Welds*, Baggages, Nature and Settings. Each group has several series, and each series several X-ray images. For instance, series C0001 contains 72 X-ray images of an aluminum casting (wheel).

Fig. 2.2 Screenshot of 𝔾𝔻𝕏ray website. Some X-ray images of ten series are shown at the *right-hand side*: `C0001` and `C0034` for castings, `W0001` and `W0003` for welds, `B0001` and `B0046` for baggage, `N0006` (cherry), `N0010` (wood), and `N0011` (salmon) for natural objects and `S0001` for settings (a calibration pattern)

Table 2.1 Statistics of 𝔾𝔻𝕏ray database

Groups	Series	Images	Size (MB)
Castings	67	2,727	307.5
Welds	3	88	209.4
Baggage	77	8,150	2,734.8
Nature	13	8,290	191.9
Settings	7	152	45.5
Total	167	19,407	3,489.0

structure is summarized in Table 2.2. All X-ray images of 𝔾𝔻𝕏ray are stored in 'png' (Portable Network Graphics)[4] format.

2.3 Castings

The group Castings contains 2,727 X-ray images arranged in 67 series. The X-ray images were taken mainly from automotive parts (aluminum wheels and knuckles). Some examples are illustrated in Figs. 2.3, 2.4 and 2.5. The details of each series are

[4] See http://www.libpng.org/pub/png/.

Table 2.2 Structure of 𝔾𝔻𝕏ray

Database	Groups	Series	X-ray images
GDXray →	Castings →	C0001 → : C0067 →	C0001_0001.png ... C0001_0072.png C0067_0001.png ... C0067_0083.png
	Welds →	W0001 → : W0003 →	W0001_0001.png ... W0001_0010.png W0003_0001.png ... W0003_0068.png
	Baggage →	B0001 → : B0077 →	B0001_0001.png ... B0001_0014.png B0077_0001.png ... B0077_00576.png
	Nature →	N0001 → : N0013 →	N0001_0001.png ... N0001_0013.png N0013_0001.png ... N0013_0006.png
	Settings →	S0001 → : S0007 →	S0001_0001.png ... S0001_0018.png S0007_0001.png ... S0007_0029.png

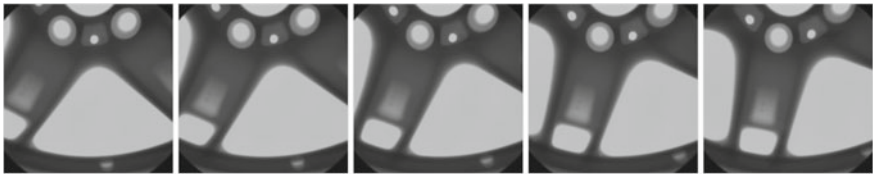


Fig. 2.3 Some X-ray images of an aluminum wheel (group `Castings` series `C0001`)

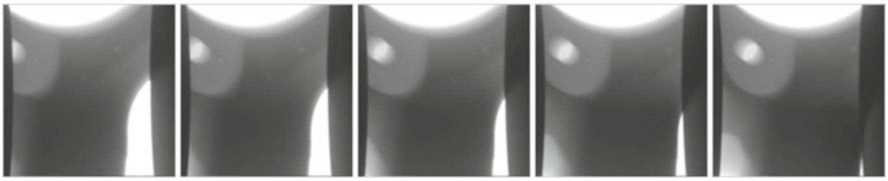


Fig. 2.4 Some X-ray images of a knuckle (group `Castings` series `C0059`)

given in Table A.2. Experiments on these data can be found in several publications as shown in Table 2.3. It is interesting to highlight that series `C0001` (see Fig. 2.3) contains not only a sequence of 72 X-ray images taken from an aluminum wheel by rotating its central axis in 5°, but also annotations of bounding boxes of the ground truth of 226 small defects and the calibration matrix of each image that relates the 3D coordinates of the aluminum wheel with 2D coordinates of the X-ray image.

Table 2.3 Applications of series Castings

Series	Application	References
C0001	Detection of defects in multiple views	[2–7]
	Estimation of epipolar geometry with distortion	[8]
	Calibration of X-ray imaging system with image intensifiers	[2]
	Simulation of casting defects	[2]
C0002	Experiments on detection of defects in single views	[9–12]
C0008	Simulation of casting defects	[13]
C0017	Simulation of casting defects	[14, 15]
C0032	Experiments on detection of defects in multiple views	[3]
C0037	Simulation of casting defects	[14, 15]
C0049	Image restoration in blurred X-ray images	[16]
C0054	Detection of casting on moving castings	[17]
C0055	Image restoration in blurred X-ray images	[16]

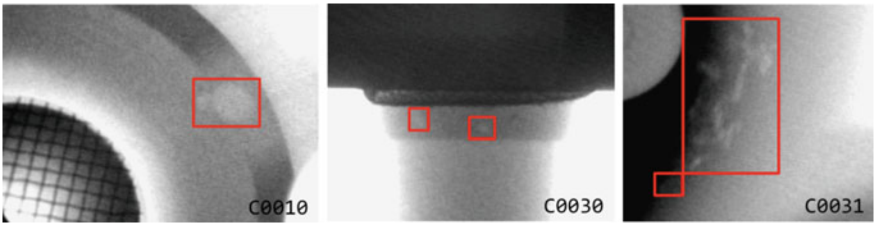


Fig. 2.5 Some annotated images showing bounding boxes of casting defects

2.4 Welds

The group Welds contains 88 images arranged in 3 series. The X-ray images were taken by the Federal Institute for Materials Research and Testing, Berlin (BAM).[5] Some examples are illustrated in Fig. 2.6. The details of each series are given in Table A.4. Experiments on these data can be found in several publications as shown in Table 2.4. It is interesting to highlight that series W0001 and W0002 (see Fig. 2.7) contains not only 10 X-ray images selected from the whole BAM database (series W0003), but also annotations of bounding boxes and the binary images of the ground truth of 641 defects.

[5] The X-ray images of series W0001 and W0003 are included in GDXray thanks to the collaboration of the Institute for Materials Research and Testing (BAM), Berlin http://dir.bam.de/dir.html.

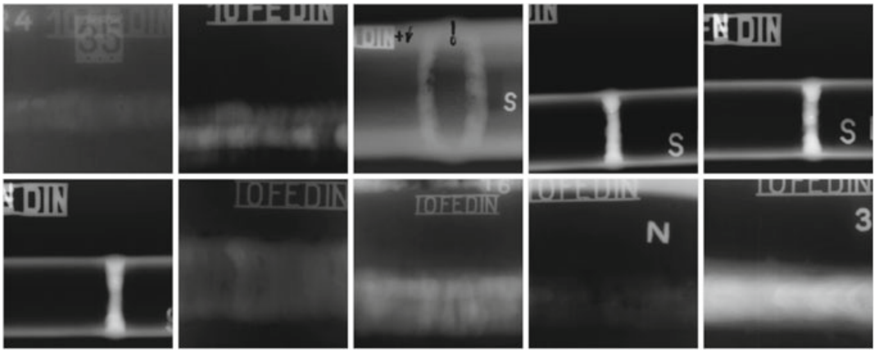


Fig. 2.6 Some X-ray images of group `Welds` series `W0003`. This series corresponds to the BAM database

Table 2.4 Applications of series Welds

Series	Application	References
`W0001`	Detection of defects in welds	[18–21]
	Simulation of welding defects	[15, 20]
`W0002`	Evaluation of performance of detection algorithm	[18]
`W0003`	Detection of defects in welds	[22, 23]

2.5 Baggage

The group Baggage contains 8,150 X-ray images arranged in 77 series. The X-ray images were taken from different containers such as backpacks, pen cases, wallets, etc. Some examples are illustrated in Figs. 2.8, 2.9 and 2.10. The details of each series are given in Table A.3. Experiments on these data can be found in several publications as shown in Table 2.5. It is interesting to highlight that series `B0046`, `B0047` and `B0048` (see for example Fig. 2.8) contains 600 X-ray images that can be used for automated detection of handguns, shuriken and razor blades (bounding boxes for these objects of interest are available as well). In this case, the training can be performed using series `B0049`, `B0050`, and `B0051` that includes X-ray images of individual handguns, shuriken and razor blades respectively taken from different points of view as shown in Fig. 2.9.

2.6 Natural Objects

The group Nature contains 8,290 X-ray images arranged in 13 series. The X-ray images were taken from different natural objects such as salmon filets, fruit, and wood pieces. Some examples are illustrated in Figs. 2.11, 2.12 and 2.13. The details

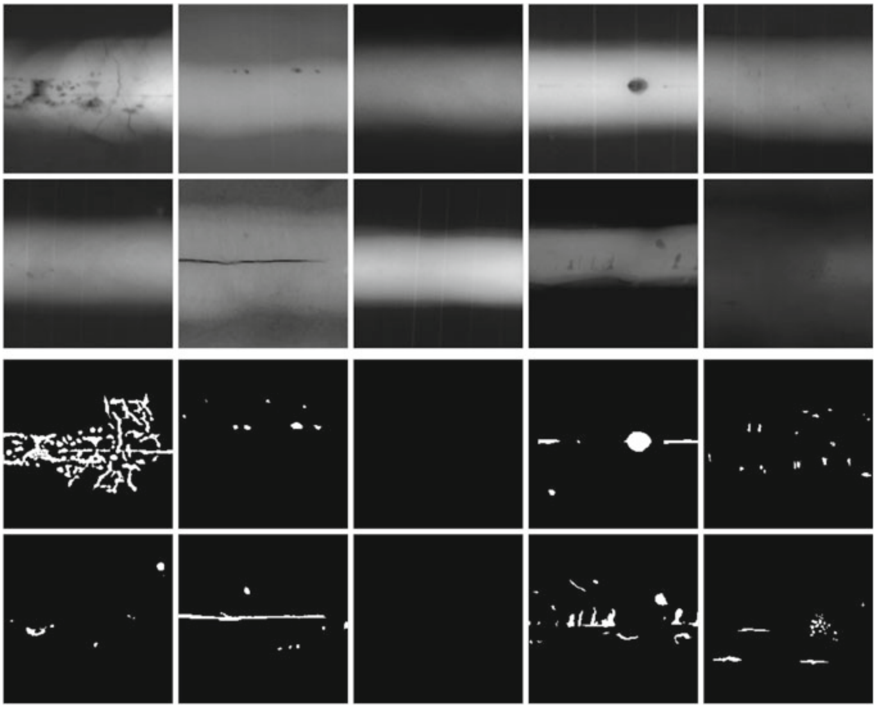


Fig. 2.7 Some images of group `Welds` series `W0001` (X-ray images) and `W0002` (ground truth)

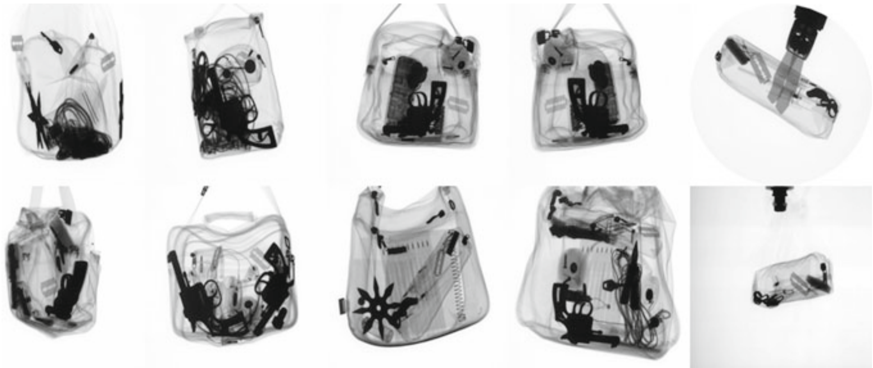


Fig. 2.8 Some X-ray images of a bag containing handguns, *shuriken* and razor blades (group `Baggage` series `B0048`)

of each series are given in Table A.1. Experiments on these data can be found in several publications as shown in Table 2.6. It is interesting to highlight that series `N0012` and `N0013` (see Fig. 2.14) contains not only six X-ray images of salmon filets, but also annotations of bounding boxes and the binary images of the ground

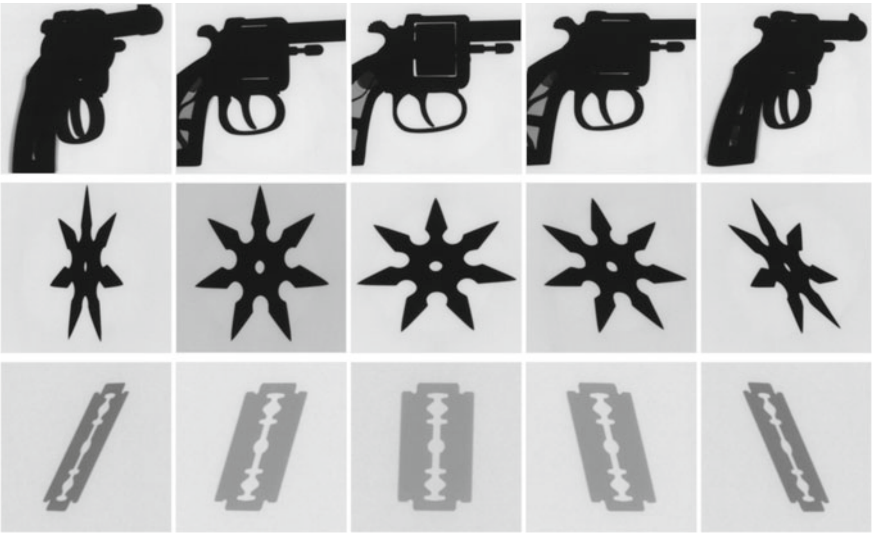


Fig. 2.9 Some X-ray images of handguns (series `B0049`), *shuriken* (series `B0050`) and razor baldes (series `B0051`) of group `Baggage`

Table 2.5 Applications of series Baggage

Series	Application	References
`B0005`	Experiments on detection of pins in multiple views	[3, 24]
	Detection of razor blades using active vision	[24]
`B0007`	Training of a classifier of razor blades	[24]
`B0009-43`	Experiments on detection of handguns	[25, 26]
`B0045`	Experiments on detection of objects in multiple views	[27, 28]
	Active vision	[24]
`B0055`	Experiments on detection of objects in sequences of four views	[27]
`B0056`	Experiments on detection of objects in sequences of six views	[27]
`B0057`	Experiments on detection of objects in sequences of eight views	[27]
`B0058`	Training of a classifier for clips, springs, and razor blades	[27, 28]
`B0061-73`	Detection of razor blades using active vision	[24]

truth of 73 fish bones. For training purposes, there are more than 7,500 labeled small crops (10 × 10 pixels), of regions of X-ray of salmon filets with and without fish bones in series `N0003`.

2.7 Settings

The group Settings contains 151 X-ray images arranged in 7 series. The X-ray images were taken from different calibration objects such checkerboards and 3D objects with regular patterns. Some examples are illustrated in Figs. 2.15 and 2.16. The details

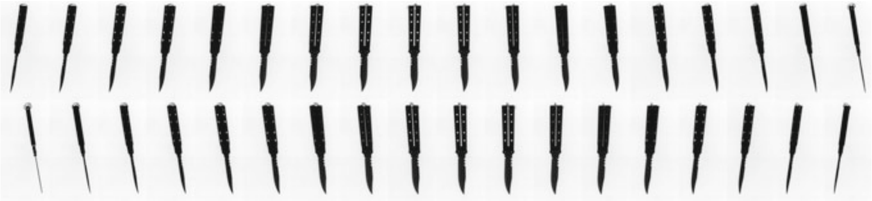


Fig. 2.10 A knife was rotated in 1° and by each position an X-ray image was captured. In this figure, X-ray images at 0°, 10°, 20°, . . . 350° are illustrated (see series `B00008` of group `Baggage`)

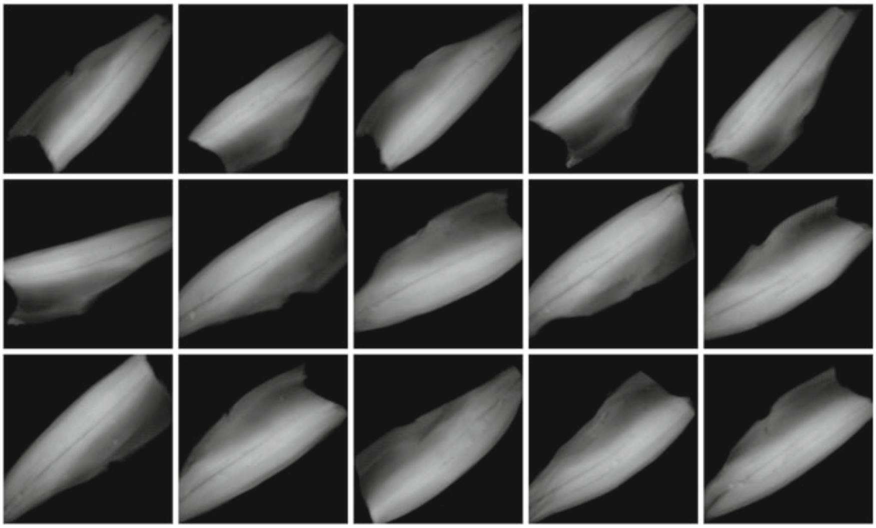


Fig. 2.11 Some X-ray images of salmon filets (group `Nature` series `N0011`)

of each series are given in Table A.5. Experiments on these data can be found in several publications as shown in Table 2.7. It is interesting to highlight that series `S0001` (see Fig. 2.15) contains not only 18 X-ray images of a copper checkerboard, but also the calibration matrix of each view. In addition, series `S0007` can be used for modeling the distortion of an image intensifier. The coordinates of each hole of the calibration pattern in each view are available, and the coordinates of the 3D model are given as well.

2.8 Matlab Commands

In order to manipulate 𝔾𝔻𝕏ray database easily, some helpful Matlab commands were developed in 𝕏VIS Toolbox. In this section, we present a summary of them with some examples.

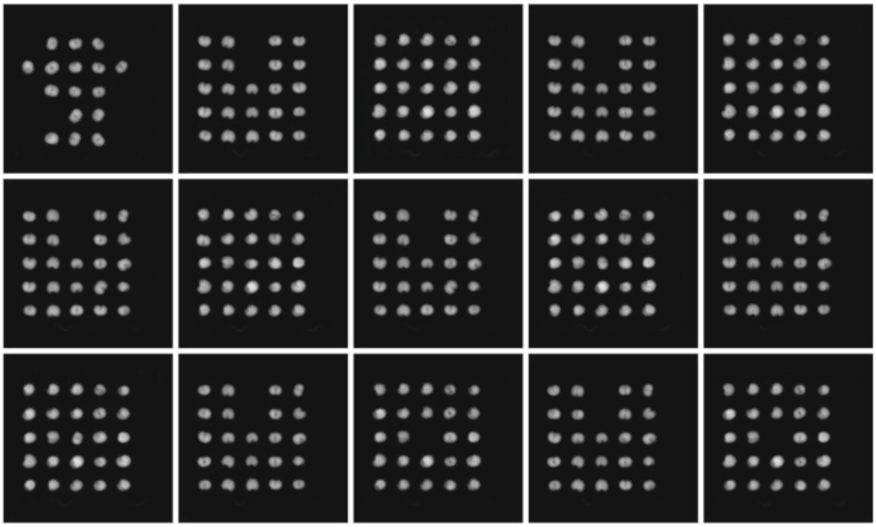


Fig. 2.12 Some X-ray images of cherries (group `Nature` series `N0006`)

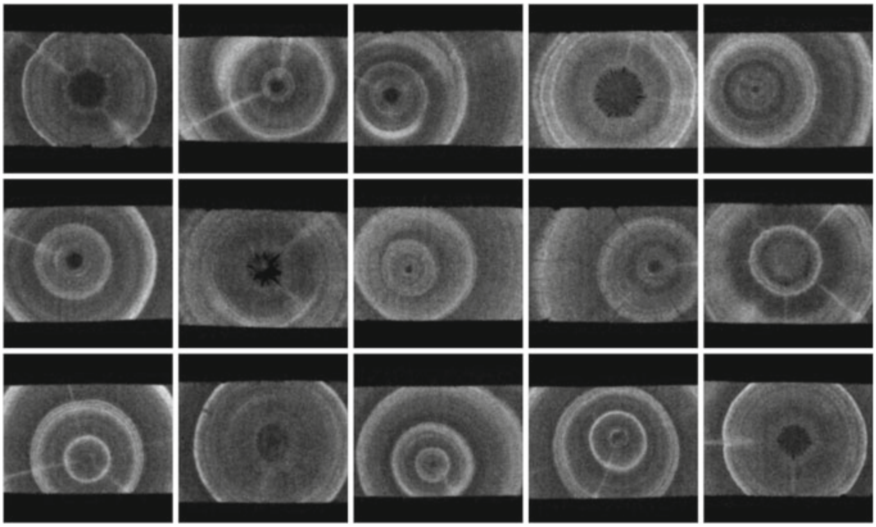


Fig. 2.13 Some X-ray images of wood (group `Nature` series `N0010`)

- Xgdxbrowse (see Appendix B): This GUI function[6] is used to browse $\mathbb{GDX}$ray database. An example is illustrated in Fig. 2.17. An additional example using pseudo-coloring is shown in Fig. 2.18, the user can select one of 10 different color maps.
- Xshowseries (see Appendix B): This function is used to display several images of a series in only one figure. For example, Fig. 2.10 was obtained using command:

[6] GUI: Graphic User Interface.

Table 2.6 Applications of series Nature

Series	Application	References
`N0003`	Automated design of a visual food quality system	[29]
`N0003`	Automated fish bone detection	[30]
`N0008`	Quality control of kiwis	[31]
`N0011`	Automated fish bone detection	[30]

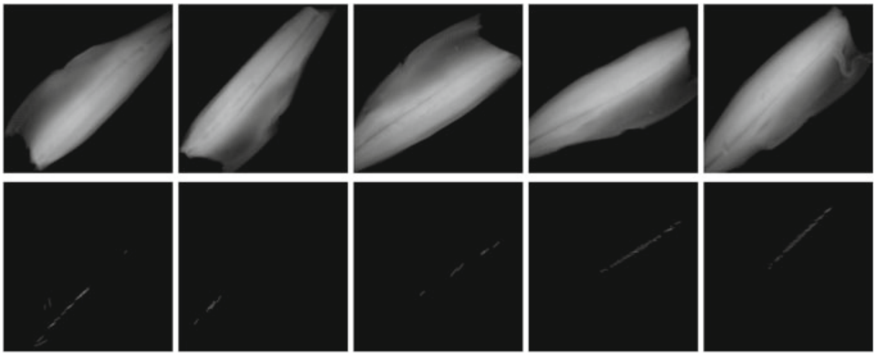


Fig. 2.14 Some images of group `Nature` series `S0012` (X-ray images of salmon filets) and `S0013` (ground truth for fish bones)

```
I = Xshowseries('B',8,1:10:351,18,0.2);
```

In this example, the images 1, 11, ..., 351 of the eighth series of group ‘B’ are displayed using 18 images per row, and a new size per image scaled to 0.2 of the original size. The output is stored in matrix `I`.

- Xgdxdir (see Appendix B): This function is used to ascertain the path of a series of 𝔾𝔻𝕏ray. For example, the directory of series `N0012` can be obtained with command:

```
str = Xgdxdir('N',12);
```

- Xgdxstats (see Appendix B): This function is used to compute some statistics of 𝔾𝔻𝕏ray. The output is Table 2.1.
- Xloadimg (see Appendix B): This function is used to load an image of 𝔾𝔻𝕏ray. For example, `N0012_0004.png` can be stored in matrix `I` using command:

```
I = Xloadimg('N',12,4,1);
```

In this example, the last parameter can be ‘1’ or ‘0’, if the user wants to display the image or not.

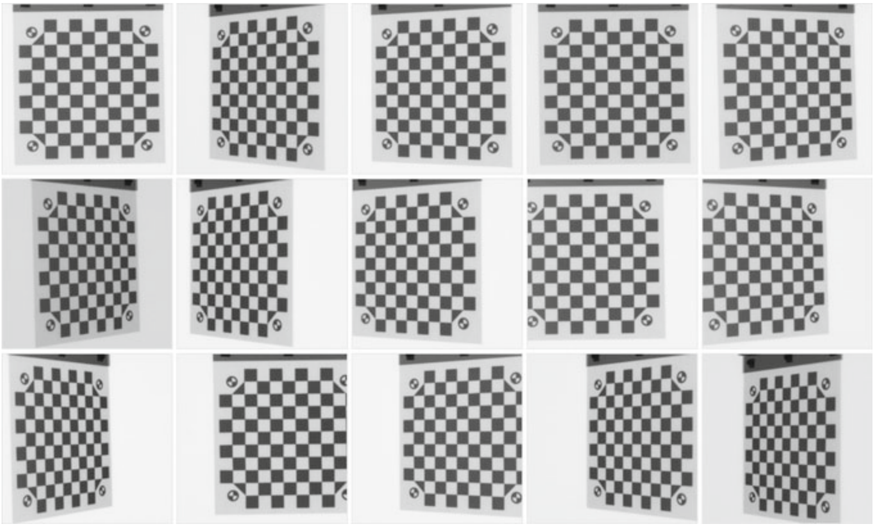


Fig. 2.15 Some X-ray images of a copper checkerboard used by calibration (group `Settings` series `S0001`)

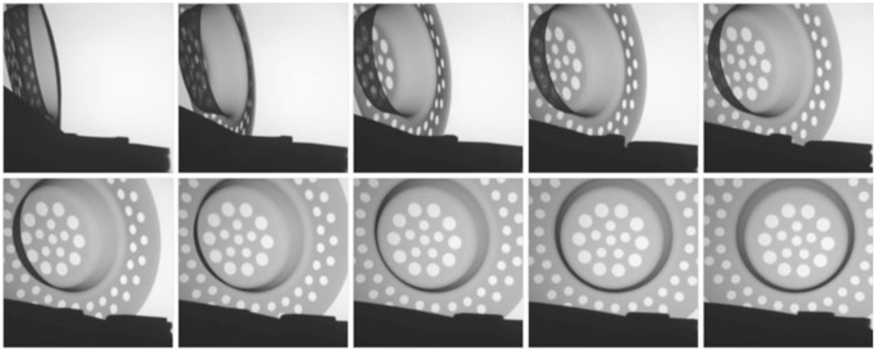
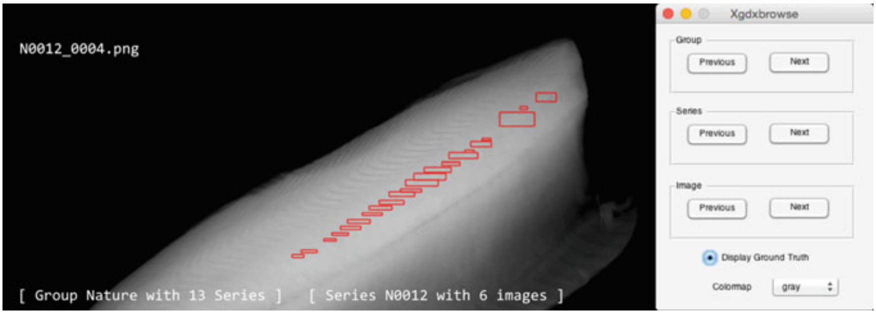


Fig. 2.16 Some X-ray images of circular pattern in different points of view used by calibration (group `Settings` series `S0007`)

Table 2.7 Applications of series Settings

Series	Application	References
`S0001`	Calibration of a multiple view X-ray imaging system for active vision	[24]
`S0002`	Distortion model of an image intensifier	[2, 8]
`S0007`	Explicit geometric model of a radioscopic imaging system	[32]



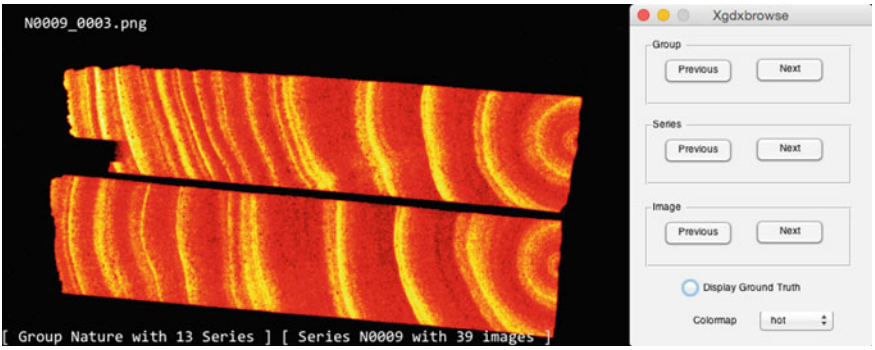
N0012_0004.png

[Group Nature with 13 Series] [Series N0012 with 6 images]

● Display Ground Truth

Colormap gray

Fig. 2.17 Example of command Xgdxbrowse (see Appendix B) that can be used to browse 𝔾𝔻𝕏ray. The user can click buttons `[Previous]` and `[Next]` to display the next groups, series or images. In addition, the ground truth option can be used to display manual annotations when they are available. In this example, the fish bones of a salmon filet are highlighted. For colored images see Fig. 2.18



N0009_0003.png

[Group Nature with 13 Series] [Series N0009 with 39 images]

○ Display Ground Truth

Colormap hot

Fig. 2.18 Example of command Xgdxbrowse (see Appendix B) using pseudo coloring of a wood X-ray image. For another example in grayscale see Fig. 2.17

- XshowGT (see Appendix B): This function is used to display the bounding boxes of an X-ray image. For example, in figure `N0012_0004.png` the bounding boxes of the ground truth (see Fig. 2.17) can be displayed using command:

```
[I,bb] = XshowGT('N',12,4,'ground_truth.txt');
```

  In this example, image `N0012_0004.png` is stored in matrix `I`, and each bounding box in a row of matrix `bb`. The ground truth was previously stored in ASCII file `ground_truth.txt`. The format of this file is as follows: one bounding box per row; the first number of the row is the number of the image of the series, and the next four values are the coordinates x_1, x_2, y_1, y_2 of a bounding box. Thus, the rectangle of a bounding box is defined by its opposite vertices: (x_1, y_1) and (x_2, y_2). Another example is given in Fig. 2.5.

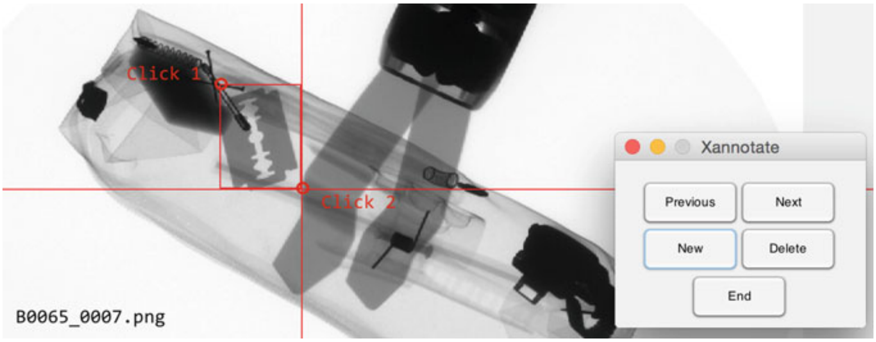


Fig. 2.19 Example of command Xgdxannotate (see Appendix B) that can be used to manually annotate the ground truth of a series. In this example, the user is annotating the razor blades of series `B0065`. With buttons `[Previous]` and `[Next]`, the user can browse the series. Button `[New]` is used to define a new bounding box in the current image by giving two clicks (the *red* axes can help to define the vertices of the bounding box). Button `[Delete]` is used to delete the last defined bounding box

- Xloaddata (see Appendix B): This function is used to load a file into workspace. For instance, the ground truth data of series `N0012` can be stored in matrix `GT` using command:

```
GT = Xloaddata('N',12,'ground_truth.txt');
```

- Xgdxannotate (see Appendix B): This function is used to manually annotate bounding boxes of a series of 𝔾𝔻𝕏ray. Function Xgdxannotate (see Appendix B) calls GUI function Xannotate (see Appendix B) that is used to annotate the images of current directory. See an example in Fig. 2.19.

2.9 Summary

In this chapter, we presented the details of a new public dataset called 𝔾𝔻𝕏ray. It consists of more than 19,400 X-ray images. The database includes five groups of X-ray images: castings, welds, baggage, natural objects, and settings. Each group has several series and X-ray images with many labels and annotations that can be used for training and testing purposes in computer vision algorithms. To the best knowledge of the author, up until now there have not been any public databases of digital X-ray images for X-ray testing.

In this chapter, we explained the structure of the 𝔾𝔻𝕏ray database, we gave a description for each group (with some series examples), we presented some examples of applications that have been published using images of 𝔾𝔻𝕏ray, and some examples in Matlab with 𝕏vis Toolbox, that can be used to manipulate the database.

We believe that 𝔾𝔻𝕏ray represents a relevant contribution to the X-ray testing community. On the one hand, students, researchers, and engineers can use these X-ray images to develop, test, and evaluate image analysis and computer vision algorithms without purchasing expensive X-ray equipment. On the other hand, these images can be used as a benchmark in order to test and compare the performance of different approaches on the same data. Moreover, the database can be used in the training programs of human inspectors.

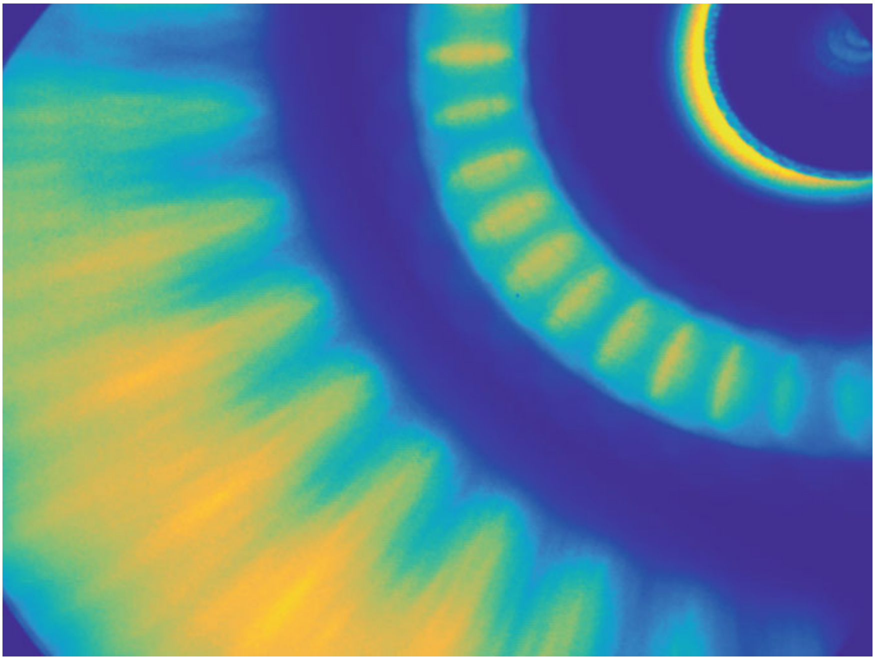
References

1. Mery, D., Riffo, V., Zscherpel, U., Mondragon, G., Lillo, I., Zuccar, I., Lobel, H., Carrasco, M.: GDXray: The GRIMA database of X-ray images (http://dmery.ing.puc.cl/index.php/material/gdxray/) (2015)
2. Mery, D., Filbert, D.: Automated flaw detection in aluminum castings based on the tracking of potential defects in a radioscopic image sequence. IEEE Trans. Robot. Autom. **18**(6), 890–901 (2002)
3. Mery, D.: Automated detection in complex objects using a tracking algorithm in multiple X-ray views. In: Proceedings of the 8th IEEE Workshop on Object Tracking and Classification Beyond the Visible Spectrum (OTCBVS 2011), in conjunction with CVPR 2011, pp. 41–48. Colorado Springs (2011)
4. Carrasco, M., Mery, D.: Automatic multiple view inspection using geometrical tracking and feature analysis in aluminum wheels. Mach. Vis. Appl. **22**(1), 157–170 (2011)
5. Pizarro, L., Mery, D., Delpiano, R., Carrasco, M.: Robust automated multiple view inspection. Pattern Anal. Appl. **11**(1), 21–32 (2008)
6. Pieringer, C., Mery, D.: Flaw detection in aluminium die castings using simultaneous combination of multiple views. Insight **52**(10), 548–552 (2010)
7. Mery, D., Carrasco, M.: Automated multiple view inspection based on uncalibrated image sequences. Lect. Notes Comput. Sci. **3540**, 1238–1247 (2005)
8. Mery, D., Filbert, D.: The epipolar geometry in the radioscopy: Theory and application. at–Automatisierungstechnik **48**(12), 588–596 (2000). (in German)
9. Mery, D.: Crossing line profile: a new approach to detecting defects in aluminium castings. In: Proceedings of the Scandinavian Conference on Image Analysis (SCIA 2003), Lecture Notes in Computer Science **2749**, 725–732 (2003)
10. Ghoreyshi, A., Vidal, R., Mery, D.: Segmentation of circular casting defects using a robust algorithm. Insight-Non-Destruct. Test. Cond. Monit. **47**(10), 615–617 (2005)
11. Ramírez, F., Allende, H.: Detection of flaws in aluminium castings: a comparative study between generative and discriminant approaches. Insight-Non-Destruct. Test. Cond. Monit. **55**(7), 366–371 (2013)
12. Mery, D., Chacón, M., Munoz, L., González, L.: Automated inspection of aluminium castings using fusion strategies. Mater. Eval. (2005). In Press
13. Huang, Q., Wu, Y., Baruch, J., Jiang, P., Peng, Y.: A template model for defect simulation for evaluating nondestructive testing in X-radiography. IEEE Trans. Syst., Man Cybern., Part A: Syst. Hum. **39**(2), 466–475 (2009)
14. Mery, D.: A new algorithm for flaw simulation in castings by superimposing projections of 3D models onto X-ray images. In: Proceedings of the XXI International Conference of the Chilean Computer Science Society (SCCC-2001), pp. 193–202. IEEE Computer Society Press, Punta Arenas (2001)
15. Mery, D., Hahn, D., Hitschfeld, N.: Simulation of defects in aluminum castings using CAD models of flaws and real X-ray images. Insight **47**(10), 618–624 (2005)

16. Mery, D., Filbert, D.: A fast non-iterative algorithm for the removal of blur caused by uniform linear motion in X-ray images. In: Proceedings of the 15th World Conference on Non-Destructive Testing (WCNDT-2000). Rome (2000)
17. Mery, D., Filbert, D.: Automated inspection of moving aluminium castings. In: 8th European Conference on Non-Destructive Testing (ECNDT 2002). Barcelona (2002)
18. Carrasco, M., Mery, D.: Segmentation of welding defects using a robust algorithm. Mater. Eval. **62**(11), 1142–1147 (2004)
19. Mery, D., Berti, M.A.: Automatic detection of welding defects using texture features. Insight-Non-Destruct. Test. Cond. Monit. **45**(10), 676–681 (2003)
20. Mery, D.: Automated detection of welding defects without segmentation. Mater. Eval. **69**(6), 657–663 (2011)
21. Hernández, S., Sáez, D., Mery, D., da Silva, R., Sequeira, M.: Automated defect detection in aluminium castings and welds using neuro-fuzzy classifiers. In: Proceedings of the 16th World Conference on Non-Destructive Testing (WCNDT-2004). Montreal (2004)
22. Perner, P., Zscherpel, U., Jacobsen, C.: A comparison between neural networks and decision trees based on data from industrial radiographic testing. Pattern Recognit. Lett. **22**(1), 47–54 (2001)
23. da Silva, R.R., Siqueira, M.H., de Souza, M.P.V., Rebello, J.M., Calôba, L.P.: Estimated accuracy of classification of defects detected in welded joints by radiographic tests. NDT & E Int. **38**(5), 335–343 (2005)
24. Riffo, V., Mery, D.: Active X-ray testing of complex objects. Insight **54**(1), 28–35 (2012)
25. Mery, D., Riffo, V., Mondragon, G., Zuccar, I.: Detection of regular objects in baggages using multiple X-ray views. Insight **55**(1), 16–21 (2013)
26. Damashek, A., Doherty, J.: Detecting guns using parametric edge matching. Technical Report Project for Computer Vision Course: CS231A, Stanford University (2015)
27. Mery, D.: Inspection of complex objects using multiple-X-ray views. IEEE/ASME Trans. Mechatron. **20**(1), 338–347 (2015)
28. Mery, D., Riffo, V., Zuccar, I., Pieringer, C.: Automated X-ray object recognition using an efficient search algorithm in multiple views. In: Proceedings of the 9th IEEE CVPR Workshop on Perception Beyond the Visible Spectrum, Portland (2013)
29. Mery, D., Pedreschi, F., Soto, A.: Automated design of a computer vision system for visual food quality evaluation. Food Bioprocess Techol. **6**(8), 2093–2108 (2013)
30. Mery, D., Lillo, I., Riffo, V., Soto, A., Cipriano, A., Aguilera, J.: Automated fish bone detection using X-ray testing. J. Food Eng. **2011**(105), 485–492 (2011)
31. Mondragón, G., Leiva, G., Aguilera, J., Mery, D.: Automated detection of softening and hard columella in kiwifruits during postharvest using X-ray testing. In: Proceedings of International Congress on Engineering and Food (2011)
32. Mery, D.: Explicit geometric model of a radioscopic imaging system. NDT & E Int. **36**(8), 587–599 (2003)

Chapter 3
Geometry in X-ray Testing

Abstract Geometry is of basic importance for understanding in X-ray testing. In this chapter, we present a mathematical background of the monocular and multiple view geometry which is normally used in X-ray computer vision systems. The chapter describes an explicit model which relates the 3D coordinates of an object to the 2D coordinates of the digital X-ray image pixel, the calibration problem, the geometric and algebraic constraints between two, three, and more X-ray images taken at different projections of the object, and the problem of 3D reconstruction from *n* views.



Cover image: *Average of X-ray images of a wheel in motion (series* `C0008` *colored with "parula" colormap).*

D. Mery, *Computer Vision for X-Ray Testing*, DOI 10.1007/978-3-319-20747-6_3

3.1 Introduction

In certain nondestructive testing and evaluation applications, it is necessary to deal with some geometric problems. For example, the geometric distortion of an image amplifier must be reduced; 3D information of the object under test must be inferred; or multiple view X-ray images of the same object from different points of view must be analyzed. Multiple view information is required, for example, for inspecting the internal and external geometry of an object under test, for locating its features using stereoscopic techniques, and for finding regions of interest—such as defects—using correspondences of multiple views.

In this chapter, we present a background of geometry which is normally used in X-ray computer vision systems. We start by presenting in Sect. 3.2 projective transformations that are very common in X-ray imaging. In Sect. 3.3, a model which relates the 3D coordinates of an object to the 2D coordinates of the digital X-ray image pixel. In Sect. 3.4, different approaches that can be used to estimate the parameters of the geometric model are outlined. In Sect. 3.5, we establish the geometric and algebraic constraints between two, three, and more X-ray images obtained as different projections of the object. The problem of the 3D reconstruction is explained in Sect. 3.6.

3.2 Geometric Transformations

Before we begin a detailed description of the geometric model of our X-ray computer vision system, it is worthwhile to outline certain geometric transformations that are used by the model.

3.2.1 *Homogeneous Coordinates*

We are familiar with Cartesian coordinates in 2D (x, y) and in 3D (X, Y, Z). As we will see in this section, in an X-ray computer vision system the geometric transformations between different coordinate systems can be handled in an easy way if *homogeneous coordinates* are used [1]. In this approach, the commonly used Cartesian coordinates are called *nonhomogeneous* coordinates.

In general, a point $\mathbf{a} \in \mathbb{R}^N$ given in nonhomogeneous coordinates can be expressed as a point $\mathbf{b} \in \mathbb{R}^{N+1}$ in homogeneous coordinates as follows:

$$(a_1, a_2, \ldots, a_N) \rightarrow (b_1, b_2, \ldots, b_N, b_{N+1}) \tag{3.1}$$

where $a_i = b_i / b_{N+1}$ for $i = 1, \ldots N$.

Using (3.1), a 2D point (x, y) is expressed as a homogeneous vector with three elements (b_1, b_2, b_3), where $x = b_1/b_3$ and $y = b_2/b_3$. Thus, we can convert a nonhomogeneous point (x, y) into a homogeneous point as $(x, y, 1)$, or as $\lambda(x, y, 1)$

Table 3.1 Transformation nonhomogeneous ↔ homogeneous coordinates

	Nonhomogeneous coordinates	↔	Homogeneous coordinates
2D:	(x, y) $(x = b_1/b_3, y = b_2/b_3)$	→ ←	$\lambda(x, y, 1)$ (b_1, b_2, b_3)
3D:	(X, Y, Z) $(X = b_1/b_4, Y = b_2/b_4, Z = b_3/b_4)$	→ ←	$\lambda(X, Y, Z, 1)$ (b_1, b_2, b_3, b_4)

where λ is a scalar $\lambda \neq 0$. It is worth noting that the homogeneous coordinates (4, 8, 2) and (6, 12, 3) represent the same 2D nonhomogeneous point because they can be expressed as $2 \cdot (2, 4, 1)$ and $3 \cdot (2, 4, 1)$, respectively. That means, $x = 2$ and $y = 4$ in nonhomogeneous coordinates.

Similar examples could be given for a 3D point (X, Y, Z). The transformations between homogeneous and nonhomogeneous coordinates are summarized in Table 3.1 for 2D and 3D.

In this book we use the notation of Faugeras [1], where we differentiate between the projective geometric objects themselves and their representations, e.g., a point in the 2D space will be denoted by m whereas its vector in homogeneous coordinates will be denoted by $\mathbf{m}$.

We can use homogeneous coordinates to represent points and lines as well. For instance, a point m and a line ℓ in 2D space can be represented as $\mathbf{m} = [x\ y\ 1]^\mathsf{T}$ and $\ell = [a\ b\ 1]^\mathsf{T}$, respectively. Thus, if m lies on ℓ then $\mathbf{m} \cdot \ell = \mathbf{m}^\mathsf{T}\ell = 0$. It is worth noting that $\lambda\mathbf{m}$ for $\lambda \neq 0$ represents the same 2D point and $k\ell$ for $k \neq 0$ represents the same line, and they fulfill $\mathbf{m}^\mathsf{T}\ell = 0$.

Two 2D points m_1 and m_2 that lie on lines ℓ fulfill $\mathbf{m}_1^\mathsf{T}\ell = 0$ and $\mathbf{m}_2^\mathsf{T}\ell = 0$, where $\mathbf{m}_1$ and $\mathbf{m}_2$ are homogeneous representations of points m_1 and m_2, respectively (see Fig. 3.1). Using cross product, we find a new vector $\mathbf{w} = \mathbf{m}_1 \times \mathbf{m}_2$ with following properties: (i) $\mathbf{w}$ is a 3D vector, (ii) $\mathbf{m}_1 \perp \mathbf{w}$, (iii) $\mathbf{m}_2 \perp \mathbf{w}$. According to properties (ii) and (iii) , $\mathbf{m}_1^\mathsf{T}\mathbf{w} = 0$ and $\mathbf{m}_2^\mathsf{T}\mathbf{w} = 0$, interestingly that means that $\mathbf{w} = \ell$. Thus,

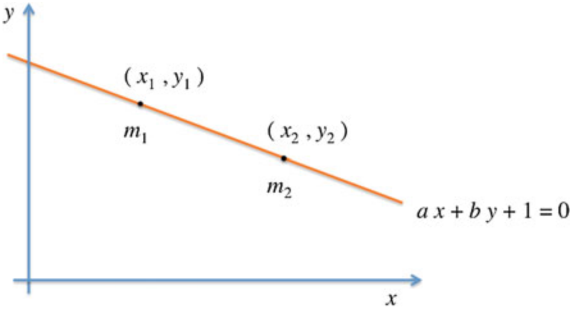


Fig. 3.1 Two points on a line in 2D space

given $\mathbf{m}_1$ and $\mathbf{m}_2$ the homogeneous representation of the line that contains both points can be easily calculated by:

$$\ell = \mathbf{m}_1 \times \mathbf{m}_2. \tag{3.2}$$

The reader can demonstrate that given ℓ_1 and ℓ_2 (the homogeneous representations of two lines in 2D space), the homogenous representation of the intersection of both lines can be computed by:

$$\mathbf{m} = \ell_1 \times \ell_2. \tag{3.3}$$

3.2.2 2D → 2D Transformation

Sometimes, a 2D point that is given in a coordinate system (x', y'), must be expressed in another coordinate system (x, y) as illustrated in Fig. 3.2. In this example, there is a rotation θ and a translation (t_x, t_y). It is the same 2D point m, however, it is defined in two different coordinate systems. It is easy to demonstrate that the transformation between both coordinate systems is given in nonhomogeneous coordinates by:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \underbrace{\begin{bmatrix} +\cos(\theta) & -\sin(\theta) \\ +\sin(\theta) & +\cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x' \\ y' \end{bmatrix} + \underbrace{\begin{bmatrix} t_x \\ t_y \end{bmatrix}}_{\mathbf{t}} = \mathbf{R} \begin{bmatrix} x' \\ y' \end{bmatrix} + \mathbf{t}. \tag{3.4}$$

Matrix $\mathbf{R}$ is known as the *rotation matrix* in 2D. It is an orthonormal matrix, i.e., $\mathbf{R}^{\mathsf{T}}\mathbf{R} = \mathbf{I}_{2\times 2}$, where $\mathbf{I}_{2\times 2}$ is the 2-by-2 identity matrix. The same transformation (3.4) can be expressed in homogenous coordinates as:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} +\cos(\theta) & -\sin(\theta) & t_x \\ +\sin(\theta) & +\cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{3.5}$$

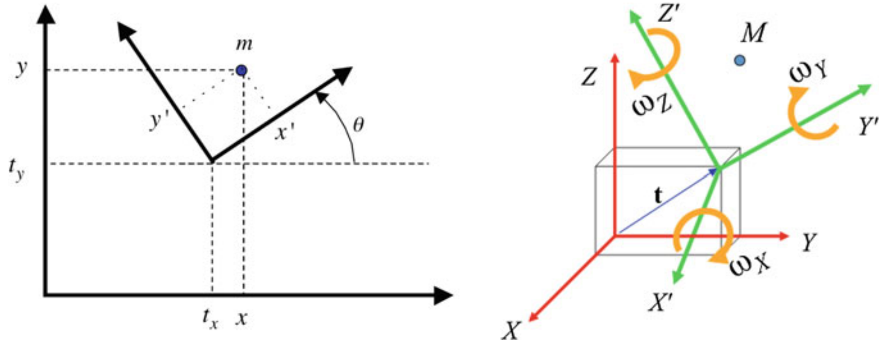


Fig. 3.2 Euclidean transformation: *left* 2D (→ Example 3.1), *right* 3D (→ Example 3.2)

or using a matrix notation as:

$$\mathbf{m} = \mathbf{H}\mathbf{m}' \tag{3.6}$$

where $\mathbf{m} = [x\ y\ 1]^\mathsf{T}$, $\mathbf{m}' = [x'\ y'\ 1]^\mathsf{T}$, and $\mathbf{H}$ is a 3 × 3 matrix defined as:

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{3.7}$$

where $\mathbf{0} = [0\ 0]$.

Equation (3.6) defines the transformation $\mathbf{m}' \to \mathbf{m}$. The inverse transformation $\mathbf{m} \to \mathbf{m}'$ can be established by computing the inverse of matrix $\mathbf{H}$, i.e., $\mathbf{m}' = \mathbf{H}^{-1}\mathbf{m}$. Since $\mathbf{R}$ is orthonormal, $\mathbf{R}^{-1} = \mathbf{R}^\mathsf{T}$. Thus, the inverse of $\mathbf{H}$ is:

$$\mathbf{H}^{-1} = \begin{bmatrix} \mathbf{R}^\mathsf{T} & -\mathbf{R}^\mathsf{T}\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{3.8}$$

Matlab Example 3.1 In Fig. 3.2, $t_x = 4.1$ cm, $t_y = 3.2$ cm, and $\theta = 35°$. The coordinates of m are given as $x = 4.9$ cm and $y = 5.5$ cm. If we want to find the coordinates of this point in (x', y') coordinate system, we can use the following Matlab code:

Listing 3.1 : Euclidean transformation 2D → 2D.

```
% EuclideanTrans2D.m
th = 35/180*pi;     % Rotation in radians
t  = [4.1 3.2]';    % Translation tx,ty in cm
R  = Xmatrixr2(th); % Rotation matrix
H  = [R t; 0 0 1];  % Euclidean transformation matrix
x  = 4.9;           % x coordinate
y  = 5.5;           % y coordinate
m  = [x y 1]';      % 2 D point in homogeneous coordinates
m p = inv(H)*m;     % Transformation m -> m p
x p = mp(1)/mp(3)   % x' coordinate
y p = mp(2)/mp(3)   % y' coordinate
```

The output of this code is: `xp = 1.9745` cm and `yp = 1.4252` cm. In this code, we use function Xmatrixr2 (see Appendix B) of 𝕏vis Toolbox. This function computes matrix $\mathbf{R}$ as defined in (3.4). It is worth noting that the division by `mp(3)` in this case is not necessary because `mp(3) = 1`, since the last row of $\mathbf{H}$ is [0 0 1]. □

This projective transformation is known as *Euclidean* or *isometric* transformation because the Euclidean distance between two points in both coordinate systems, (x, y) and (x', y'), is invariant. That means, the distance d' between two points in the first coordinate systems $\mathbf{m}'_i = [x'_i\ y'_i\ 1]^\mathsf{T}$, for $i = 1, 2$, is equal to the distance d between the two transformed points in the second coordinate systems $\mathbf{m}_i = [x_i\ y_i\ 1]^\mathsf{T}$ using (3.6): $\mathbf{m}_i = \mathbf{H}\mathbf{m}_i$. The distances are the same ($d' = d$) and they can be calculated by:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \text{ and } d' = \sqrt{(x'_1 - x'_2)^2 + (y'_1 - y'_2)^2} \qquad (3.9)$$

Other projective transformations are:

- *Similarity* transformation, in which matrix $\mathbf{R}$ is replaced by $s\mathbf{R}$ in (3.7). Factor s is a scalar that is used to change the scale of the original coordinate system. Thus, using (3.9) if the distance between two points in (x', y') coordinate system is d', in the transformed coordinate system (x, y) the distance between the transformed points will be $d = sd'$. In this transformation, matrix $s\mathbf{R}$ is no longer orthonormal but still orthogonal: $[s\mathbf{R}]^\mathsf{T}[s\mathbf{R}] = s^2\mathbf{I}_{2\times 2}$.
- *Affine* transformation, in which matrix $\mathbf{R}$ is replaced by any rank 2 matrix $\mathbf{A}$ in (3.7). In affine transformation, parallel lines in (x', y') coordinate system remain parallel in the transformed coordinate system (x, y).
- *General* transformation, in which matrix $\mathbf{H}$ in (3.7) is a general nonsingular matrix. In general transformation, a straight line in (x', y') coordinate system remains a straight line in the transformed coordinate system (x, y). This transformation is known as *homography* [2].

3.2.3 3D → 3D Transformation

A 3D point M can be defined as (X, Y, Z) and (X', Y', Z') in two different coordinate systems as shown in Fig. 3.2, where the axes are translated and rotated. Using homogeneous coordinates (similar to 2D → 2D transformation), a 3D Euclidean transformation can be expressed by:

$$\mathbf{M} = \mathbf{H}\mathbf{M}' \qquad (3.10)$$

where $\mathbf{M} = [X\ Y\ Z\ 1]^\mathsf{T}$, $\mathbf{M}' = [X'\ Y'\ Z'\ 1]^\mathsf{T}$, and $\mathbf{H}$ is a 4×4 matrix defined as

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}. \qquad (3.11)$$

where $\mathbf{0} = [0\ 0\ 0]$. Again we differentiate between M and $\mathbf{M}$. The first notation is a 3D point in space, where the second notation is its homogenous representation in a specific coordinate system. Note that $\mathbf{M}$ and $\mathbf{M}'$ represent the same 3D point M but in different coordinate systems.

The transformation is considered as rigid displacement of the coordinate system (X', Y', Z') represented by a 3×1 translation vector $\mathbf{t} = [t_X\ t_Y\ t_Z]^\mathsf{T}$ and a 3×3 rotation matrix $\mathbf{R}$. Matrix $\mathbf{R}$ considers three rotations as shown in Fig. 3.3. Each rotation can be modeled using a rotation matrix of two coordinates as shown in Table 3.2. Thus, matrix $\mathbf{R}$ can be modeled as a rotation about axis Z, then a rotation in Y-axis, and finally a rotation in X-axis:

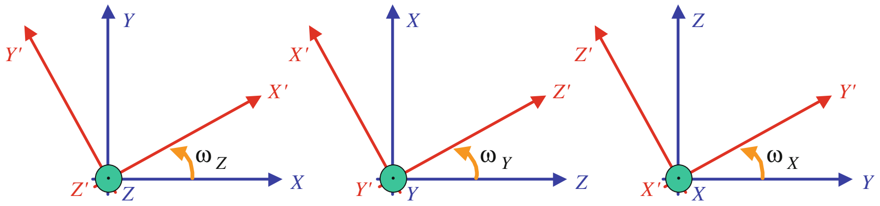


Fig. 3.3 Rotation of axes Z, Y, and X

Table 3.2 Rotation matrices of axes Z, Y, and X

Rotation	Rotation matrix
Axis Z	$\mathbf{R_Z} = \begin{bmatrix} \cos(\omega_Z) & -\sin(\omega_Z) & 0 \\ \sin(\omega_Z) & \cos(\omega_Z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Axis Y	$\mathbf{R_Y} = \begin{bmatrix} \cos(\omega_Y) & 0 & \sin(\omega_Y) \\ 0 & 1 & 0 \\ -\sin(\omega_Y) & 0 & \cos(\omega_Y) \end{bmatrix}$
Axis X	$\mathbf{R_X} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\omega_X) & -\sin(\omega_X) \\ 0 & \sin(\omega_X) & \cos(\omega_X) \end{bmatrix}$

$$\mathbf{R}(\omega_X, \omega_Y, \omega_Z) = \mathbf{R}_X(\omega_X)\mathbf{R}_Y(\omega_Y)\mathbf{R}_Z(\omega_Z) = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}, \tag{3.12}$$

where the elements R_{ij} can be expressed as a function of cosine and sine of the Euler angles ω_X, ω_Y, and ω_Z that describe the rotation of the X, Y, and Z axes, respectively [3]:

$$\begin{aligned}
R_{11} &= \cos(\omega_Y)\cos(\omega_Z) \\
R_{12} &= -\cos(\omega_Y)\sin(\omega_Z) \\
R_{13} &= \sin(\omega_Y) \\
R_{21} &= \sin(\omega_X)\sin(\omega_Y)\cos(\omega_Z) + \cos(\omega_X)\sin(\omega_Z) \\
R_{22} &= -\sin(\omega_X)\sin(\omega_Y)\sin(\omega_Z) + \cos(\omega_X)\cos(\omega_Z)\,. \\
R_{23} &= -\sin(\omega_X)\cos(\omega_Y) \\
R_{31} &= -\cos(\omega_X)\sin(\omega_Y)\cos(\omega_Z) + \sin(\omega_X)\sin(\omega_Z) \\
R_{32} &= \cos(\omega_X)\sin(\omega_Y)\sin(\omega_Z) + \sin(\omega_X)\cos(\omega_Z) \\
R_{33} &= \cos(\omega_X)\cos(\omega_Y)
\end{aligned} \tag{3.13}$$

Matlab Example 3.2 In Fig. 3.2, $t_X = 1$ mm, $t_Y = 3$ mm, $t_Z = 2$ mm, $\omega_X = 35°$, $\omega_Y = 0°$, and $\omega_Z = 0°$. The coordinates of the blue point are given as $X' = 0$ mm, $Y' = 1$ mm, and $Z' = 1$ mm. If we want to find the coordinates of this point in (X, Y, Z) coordinate system, we can use the following Matlab code:

Listing 3.2 : Euclidean transformation 3D → 3D.

```
% EuclideanTrans3D.m
w  = 35/180*pi;         % Rotation wX in radians
t  = [1 3 2]';          % Translation tX,tY,tZ in mm
R  = Xmatrixr3(w,0,0); % Rotation matrix
H  = [R t; 0 0 0 1];    % Euclidean transformation matrix
Xp = 0;                 % Xp coordinate
Yp = 1;                 % Yp coordinate
Zp = 1;                 % Zp coordinate
Mp = [Xp Yp Zp 1]';     % 3D point in homogeneous coordinates
M  = H*Mp;              % Transformation Mp -> M
X  = M(1)/M(4)          % X coordinate
Y  = M(2)/M(4)          % Y coordinate
Z  = M(3)/M(4)          % Y coordinate
```

The output of this code is: `X = 1` mm, `Y = 3.2456` mm and `Z = 3.3927` mm. In this code, we use function Xmatrixr3 (see Appendix B) of 𝕏vis Toolbox. This function computes matrix **R** as defined in (3.13). It is worth noting that the division by `M(4)` in this case is not necessary because `M(4)` $= 1$, since the last row of **H** is [0 0 0 1]. □

3.2.4 3D → 2D Transformation

In an X-ray computer vision system, a 3D point is projected using a *perspective* transformation as illustrated in Fig. 3.4. Besides applying different physical principles and technologies, it is common in X-ray testing to use terminology as introduced for optical imaging, such as optical axis, focal length, and so forth. In this model, a 3D point M is defined in (X, Y, Z) coordinate system, which is projected into projection plane $Z = f$ (called the retinal plane Π), where f is the focal length. All X-rays come from optical center C defined in $(X = 0, Y = 0, Z = 0)$. We define t as the straight line on which C and M lie (see Fig. 3.4). This line will be denoted as $\langle C, M \rangle$. Thus, the projection point m defined in (x, y) coordinate system is given by the intersection of t with the projection plane Π. This operation is called *central projection* [2]. The origin o of (x, y) coordinate system is pierced by the optical axis (Z-axis). After intercept theorem, it should be clear that:

$$\frac{Z}{f} = \frac{Y}{y} = \frac{X}{x}, \tag{3.14}$$

that can be expressed as:

$$\begin{cases} Zx = fX \\ Zy = fY \end{cases} \tag{3.15}$$

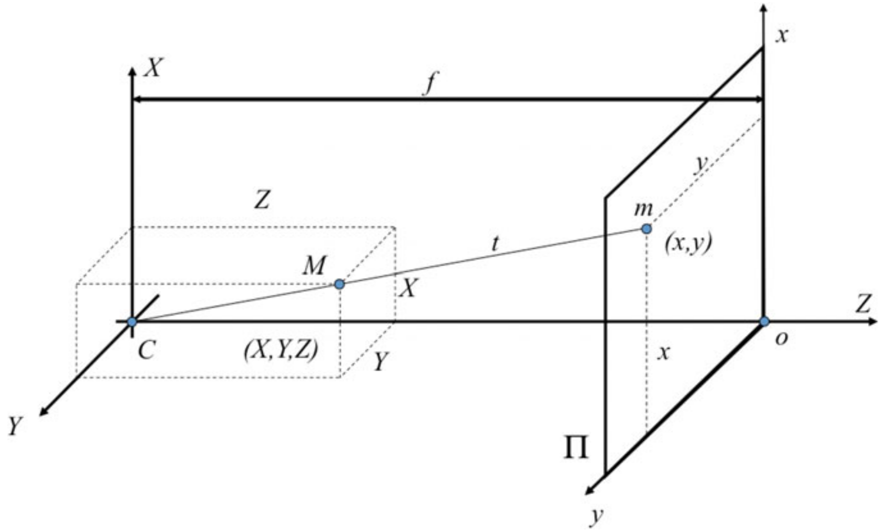


Fig. 3.4 Perspective transformation in an X-ray computer vision system (→ Example 3.3)

or using a matrix notation:

$$Z\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{P}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{3.16}$$

Matrix **P** is a 3 × 4 matrix known as the perspective projection matrix. Thus, the projected point given in homogeneous coordinates $\mathbf{m} = [x\ y\ 1]^{\mathsf{T}}$ is proportional to **PM**, where **M** is the 3D point given in homogeneous coordinates $\mathbf{M} = [X\ Y\ Z\ 1]^{\mathsf{T}}$. Usually, (3.16) is written in the following form:

$$\lambda\mathbf{m} = \mathbf{PM}. \tag{3.17}$$

where λ is a scale factor with $\lambda \neq 0$.

Matlab Example 3.3 In Fig. 3.4, $f = 100\,\text{cm}$, $X = 20\,\text{cm}$, $Y = 30\,\text{cm}$, and $Z = 50\,\text{cm}$. If we want to find the coordinates of projected point in (x, y) coordinate system, we can use the following Matlab code:

Listing 3.3 : Euclidean transformation 3D → 2D.

```
% PerspectiveTrans.m
f = 100;         % Focal distance in cm
X = 20;          % X coordinate in cm
Y = 30;          % Y coordinate in cm
Z = 50;          % Z coordinate in cm
M = [X Y Z 1]';  % 3D point in homogeneous coordinates
P = Xmatrixp(f); % Rotation matrix
m = P*M;         % Transformation M -> m
x = m(1)/m(3)    % X coordinate
y = m(2)/m(3)    % Y coordinate
```

The output of this code is: `x = 40` cm and `y = 60` cm. In this code, we use function Xmatrixp (see Appendix B) of $\mathbb{X}$VIS Toolbox. This function computes matrix **P** as defined in (3.16). □

As we can see, if nonhomogeneous coordinates are used, will be nonlinear (see (3.14)), however, it is linear in homogeneous coordinates (see (3.17)). In addition, all explained transformations are linear in homogeneous coordinates. This is the reason why we use homogeneous coordinates in X-ray computer vision systems. Thus, we can handle all projective transformations easily. For instance, if M is given in another coordinate system (X', Y', Z') ($\mathbf{M} = \mathbf{HM}'$ as shown in (3.11)), it is very simple to replace this transformation in (3.17) yielding $\lambda\mathbf{m} = \mathbf{PHM}'$. The reader can note that the same is valid for point m that can be given in another coordinate system.

3.3 Geometric Model of an X-ray Computer Vision System

The geometric model of the X-ray computer vision system establishes the relationship between 3D coordinates of the object under test and their corresponding 2D digital X-ray image coordinates. The model is required by both reconstructing 3D information from image coordinates and reprojecting 2D image coordinates from 3D information. As explained in Sect. 1.4, the principal aspects of an X-ray computer vision system are shown in Fig. 1.8, where an X-ray image of a casting is taken. Typically, it comprises the following five steps: (i) a manipulator for handling the test piece, (ii) an X-ray source, which irradiates the test piece with a conical beam to generate an X-ray image of the test piece, (iii) an image intensifier which transforms the invisible X-ray image into a visible one, (iv) a CCD camera which records the visible X-ray image, and (v) a computer to process the digital image of the X-ray image and then classify the test piece by accepting or rejecting it. Steps (iii) and (iv) can be replaced by a flat panel. Flat amorphous silicon detectors can be used as image sensors in some industrial inspection systems. In such detectors, using a semiconductor, energy from the X-ray is converted directly into an electrical signal (without image intensifier). Nevertheless, NDT using flat detectors is less feasible due to their higher cost compared to image intensifiers. In this section, we will present a geometric model for computer vision systems for both flat detectors

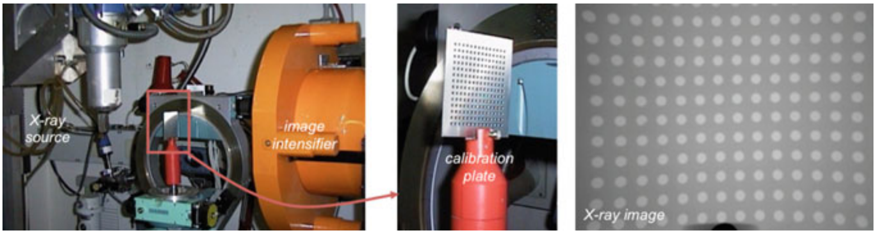


Fig. 3.5 Geometric and electromagnetic distortions obtained in an X-ray image of a regular object using an image intensifier

and image intensifiers. Image intensifiers suffer from two significant distortions: geometric and electromagnetic field distortions (see an example in Fig. 3.5). On the other hand, computer vision systems based on flat detectors do not suffer from these distortions, and they can be easily modeled with a simple pinhole camera model [1].

In this section, we will give a geometric viewpoint about how an X-ray computer vision system can be explicitly modeled. When using explicit models, the physical parameters of the computer vision system, like image center, focal length, etc., are considered independently [4]. The model presented in this section maps the 3D object into a digital X-ray image using two transformations as shown in Fig. 1.8: (i) linear central projection in the X-ray projection; (ii) digital image formation, i.e., perspective transformation in the image intensifier and 2D projective transformation in the CCD camera, or a single 2D projective transformation when using a flat panel. When modeling the image intensifier a high accuracy explicit model is presented, which takes into account the nonlinear distortion caused by the curved input screen of the image intensifier (see Fig. 1.8), and the nonlinear projection in the image intensifier caused by electromagnetic fields.

3.3.1 A General Model

In this section, we present a general model which relates the 3D coordinates of the test object to the 2D coordinates of the digitalized X-ray image pixel. The model consists of two parts as shown in Fig. 1.8: X-ray projection and digital image formation. The coordinate systems used in our approach are illustrated in Fig. 3.6.

First, we will describe how a 3D point M is projected onto a projection plane Π, called the retinal plane of the X-ray projection, in which the X-ray image is formed through central projection. In case of image intensifiers, the retinal plane is fictitious and is located tangentially to the input screen of the image intensifier, as shown in Fig. 3.7. The optical center C of the central projection corresponds to the X-ray source, modeled as a point.[1] The optical center is located at a distance f, the focal

[1] Although industrial X-ray generators use standard tubes with larger focal size that blur the X-ray images slightly, the assumption that the X-ray source can be modeled as a point is valid for

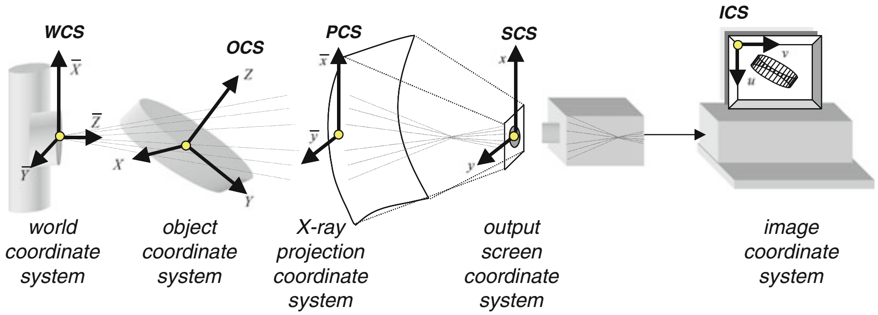


Fig. 3.6 Diagram of the coordinate systems (see Fig. 1.8)

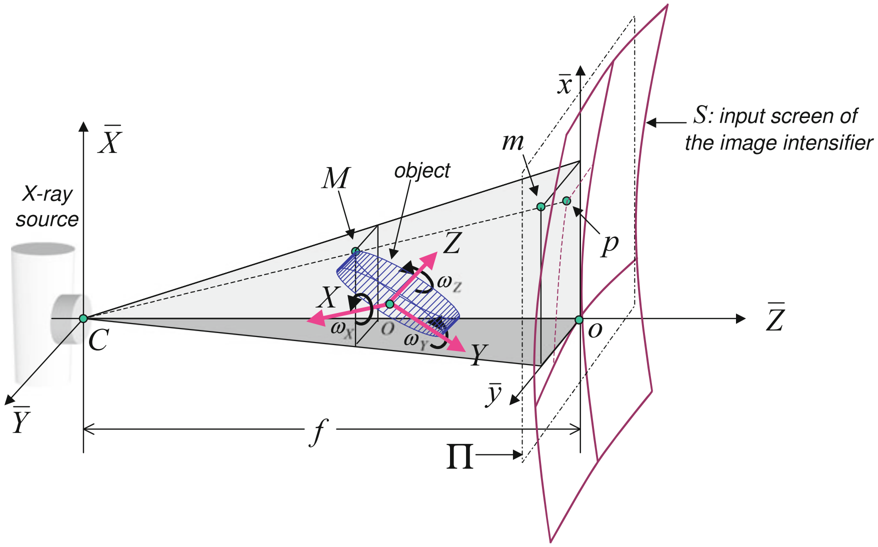


Fig. 3.7 X-ray projection using an image intensifier (see S surface) or a flat panel (see Π plane)

length of the retinal plane. The central projection of M onto projection plane Π is the point m. It is defined as the intersection of the line that contains the points C and M with the retinal plane Π. The optical axis is defined as the line going through the optical center C and perpendicular to the retinal plane Π.

We define a 3D *world coordinate system* (WCS) in the optical center C of the central projection. The coordinates of this coordinate system are $\bar{X}$, $\bar{Y}$, and $\bar{Z}$, where the $\bar{Z}$-axis coincides with the optical axis, as represented in Fig. 3.7. In WCS, the

(Footnote 1 continued)
geometrical measurements. This is because the position of a point in the X-ray image can still be estimated as the center of the blurred point [5].

retinal plane Π is defined by $\bar{Z} = f$. The coordinates of the 3D point M are denoted by $(\bar{X}, \bar{Y}, \bar{Z})$ in this coordinate system.

Now, we define a 3D *object coordinate system* (OCS) that is attached to the object to be projected. The coordinates of the 3D point M are denoted by (X, Y, Z) in OCS. The center of the object is assumed to be at the origin O of this coordinate system, as shown in Fig. 3.7. The OCS is then considered as a rigid displacement of the WCS represented by a translation 3-component vector $\mathbf{t} = [t_X \; t_Y \; t_Z]^{\mathsf{T}}$ and a 3×3 rotation matrix $\mathbf{R}$ as explained in Sect. 3.2.3. Vector $\mathbf{t}$ represents the origin of OCS given in coordinates of WCS, and matrix $\mathbf{R}$ depends on the Euler angles ω_X, ω_Y, and ω_Z as explained in (3.13).

The perspective projection of M onto the projection plane is the 2D point m that is represented as $(\bar{x}, \bar{y})$ in a new 2D coordinate system called the *X-ray projection coordinate system* (PCS). The $\bar{x}$, $\bar{y}$-axis are parallel to the $\bar{X}$, $\bar{Y}$-axis, respectively. Applying intercept theorem, the coordinates of m in this 2D system are $\bar{x} = f\bar{X}/\bar{Z}$ and $\bar{y} = f\bar{Y}/\bar{Z}$. Using homogenous coordinates as in Sects. 3.2.3 and 3.2.4 we obtain:

$$\lambda \begin{bmatrix} \bar{x} \\ \bar{y} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \qquad (3.18)$$

where $\mathbf{0} = [0\;0\;0]$, and λ is a scale factor $\lambda \neq 0$. Equation (3.18) can be rewritten in matrix form as:

$$\lambda \mathbf{m} = \mathbf{AM}, \qquad (3.19)$$

where $\mathbf{A} = \mathbf{PH}$, and the 3-component vector $\mathbf{m}$ and the 4-component vector $\mathbf{M}$ are homogeneous representations of $(\bar{x}, \bar{y})$ and (X, Y, Z), respectively (e.g., $\mathbf{m} = [\bar{x}\;\;\bar{y}\;\;1]^{\mathsf{T}}$ and $\mathbf{M} = [X\;\;Y\;\;Z\;\;1]^{\mathsf{T}}$). Equation (3.19) is a linear equation that maps object coordinates to projection plane coordinates. This equation depends on seven parameters:

$$\theta_{ext} = [f \; \omega_X \; \omega_Y \; \omega_Z \; t_X \; t_Y \; t_Z]^{\mathsf{T}}. \qquad (3.20)$$

They are called the *extrinsic* parameters of the X-ray computer vision system. Thus, $\mathbf{A} := \mathbf{A}(\theta_{ext})$.

Finally, we introduce the 2D *image coordinate system* (ICS) to represent the pixel coordinates (u, v) of the digital image. The point (u, v) in ICS can be calculated from the point $(\bar{x}, \bar{y})$ in PCS using a function γ:

$$\mathbf{w} = \gamma(\mathbf{m}, \theta_{int}), \qquad (3.21)$$

where the 3-component vectors $\mathbf{w}$ and $\mathbf{m}$ are homogeneous representations of (u, v) and $(\bar{x}, \bar{y})$, respectively, and θ_{int} is a vector with the parameters of the transformation called the *intrinsic* parameters. Several linear and nonlinear models of γ, that were developed for X-ray computer vision systems and CCD cameras, will be discussed in Sect. 3.3.2. On the one hand, the geometric model of image formation using a flat panel is linear and can be modeled using a 2D → 2D geometric transformation as explained in Sect. 3.2.2. On the other hand, image intensifiers must be modeled using nonlinear transformations due to geometric and electromagnetic distortions.

To summarize, using (3.19) for the perspective projection and (3.21) for the digital image formation, an object point M, whose homogeneous coordinates are $\mathbf{M} = [X\ Y\ Z\ 1]^{\mathsf{T}}$ (in OCS), can be mapped into a 2D point of the digital X-ray image as w, the homogeneous coordinates of which are $\mathbf{w} = [u\ v\ 1]^{\mathsf{T}}$ (in ICS) using the following expression:

$$\mathbf{w} = \gamma(\mathbf{A}(\theta_{ext})\mathbf{M}, \theta_{int}) := \mathbf{F}(\theta, \mathbf{M}) \tag{3.22}$$

where $\theta^{\mathsf{T}} = [\theta_{ext}\ ;\ \theta_{int}]$ is the vector of parameters involved in the projection model.

As we will explain in Sect. 3.4, in a process termed *calibration*, we estimate the parameters θ of the model based on n points whose object coordinates $\mathbf{M}_i = [X_i\ Y_i\ Z_i\ 1]^{\mathsf{T}}$ are known and whose image coordinates $\tilde{\mathbf{w}}_i = [\tilde{u}_i\ \tilde{v}_i\ 1]^{\mathsf{T}}$ are measured, for $i = 1, \ldots, n$. Using (3.22) we obtain the *reprojected* points $\mathbf{w}_i = [u_i\ v_i\ 1]^{\mathsf{T}}$, i.e., the inferred projections in the digital image computed from the calibration points $\mathbf{M}_i$ and the parameter vector θ. The parameter vector is then estimated by minimizing the distance between measured points ($\tilde{\mathbf{w}}_i$) and inferred points ($\mathbf{w}_i = \mathbf{F}(\theta, \mathbf{M}_i)$). Thus, the calibration is performed by minimizing the objective function $\mu(\theta)$ defined as the mean-square discrepancy between these points:

$$\mu(\theta) = \frac{1}{n}\sum_{i=1}^{n} \parallel \tilde{\mathbf{w}}_i - \mathbf{w}_i \parallel \rightarrow \text{min.} \tag{3.23}$$

The calibration problem is a nonlinear optimization problem. Generally, the minimization of $\mu(\theta)$ has no closed-form solution. For this reason, the objective function must be iteratively minimized starting with an initial guess θ° that can be obtained from nominal values or preliminary reference measurements.

3.3.2 Geometric Models of the Computer Vision System

In this section, we present seven existing models that can be used to calibrate an X-ray computer vision system. Five models were conceived to calibrate cameras with and without distortion. The others were developed to calibrate computer vision systems with image intensifiers. In all these models, the perspective projection OCS → PCS is done using (3.19). For this reason, in this section only the transformation PCS → ICS will be described. We use the definition given in (3.21), where a point $\mathbf{m} = [\bar{x}\ \bar{y}\ 1]^{\mathsf{T}}$ in PCS is transformed by a function γ into a point

$\mathbf{w} = [u\ v\ 1]^{\mathsf{T}}$ in ICS. Recall that the parameters of γ are the intrinsic parameters of the computer vision system.

Camera models

Faugeras and Toscani present in [6] a linear model without considering distortion:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & s & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \\ 1 \end{bmatrix}. \tag{3.24}$$

The five (intrinsic) parameters of the model consider scale factors (k_u, k_v) in each ordinate, a skew factor (s) that models non-orthogonal u, v-axis, and a translation of the origin (u_0, v_0) that represents the projection of $(\bar{x}, \bar{y}) = (0, 0)$ in ICS. This model can be used to model an X-ray computer vision system with a flat panel. In this linear model, the focal length is normalized to $f = 1$. A linear approach based on a least squares technique is proposed in [6] to estimate the intrinsic and extrinsic parameters in a closed form. However, Faugeras in [1] proposes minimizing the distances between the observations and the model in ICS using the objective function μ of (3.23). Faugeras reported that this nonlinear method clearly appears to be more robust than the linear method of Faugeras and Toscani when the measured data is perturbed by noise.

In order to model the distortion, a positional error (δ_u, δ_v) can be introduced:

$$\underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \\ 1 \end{bmatrix}}_{\mathbf{w}'} + \begin{bmatrix} \delta_u(\bar{x}, \bar{y}) \\ \delta_v(\bar{x}, \bar{y}) \\ 0 \end{bmatrix}. \tag{3.25}$$

In this model, the ideal nonobservable position w' is displaced to the real position w as illustrated in Fig. 3.8. The amount of the displacements, δ_u and δ_v, usually depends on the point position $(\bar{x}, \bar{y})$. Several models for the positional error were reported in the literature to calibrate a camera [3, 5, 7, 8]. In these models, the skew s is zero, because in modern digital cameras the u, v-axis can be considered as orthogonal.

The distortion is decomposed into two components: *radial* and *tangential* distortions as shown in Fig. 3.8.

Radial and tangential distortions depend on r and ϕ, respectively, where (r, ϕ) are the polar coordinates of the ideal position $(\bar{x}, \bar{y})$ represented in PCS. Tsai in [3] uses a simple radial distortion model with only one additional parameter, because his experience with cameras shows that only radial distortion, which is principally caused by flawed radial curvature of the lens elements, needs to be considered.

Weng et al. propose in [5] an implicit model that includes radial, decentering, and prism distortions. Decentering distortion arises when the optical centers of the lens elements are not exactly collinear, whereas the prism distortion occurs from

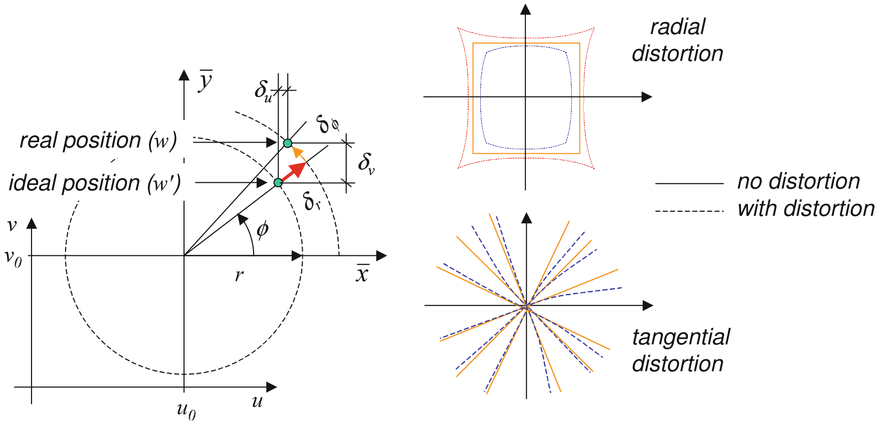


Fig. 3.8 Radial and tangential distortions [5]

imperfection in lens design, manufacturing, and camera assembly. The last two distortions, modeled with five parameters, have both radial and tangential components.

Heikkilä introduces in [7] an implicit model for radial and decentering distortions that take into account an inverse distortion model to express the distorted image coordinates in terms of their undistorted coordinates. The number of parameters of this model is four.

Swaminathan and Nayar present in [8] a model for wide-angle lenses and polycameras. The model considers a shift of the optical center, radial distortion, and decentering distortion. A shift of the optical center means a shift of the image detector in the image plane. The suggested total distortion includes four parameters.

Image intensifier models

Two models were reported in the literature to calibrate an X-ray computer vision system composed by image intensifier and CCD camera. The first model was proposed independently by Jaeger in [9] and Brack et al. in [10]. They propose an implicit model between PCS and ICS. The transfer function γ (3.21) is a third degree polynomial with 20 intrinsic parameters ($a_i, b_i, i = 0, \dots 9$) given by:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_0 \; a_1 \; \dots \; a_9 \\ b_0 \; a_2 \; \dots \; b_9 \end{bmatrix} \begin{bmatrix} 1 \; \bar{x} \; \bar{y} \; \bar{x}\bar{y} \; \bar{x}^2 \; \bar{y}^2 \; \bar{y}\bar{x}^2 \; \bar{x}\bar{y}^2 \; \bar{x}^3 \; \bar{y}^3 \end{bmatrix}^{\mathsf{T}}. \tag{3.26}$$

This cubic function can model not only the distortion caused by the (curved) input screen, but also the distortion introduced by electromagnetic fields present around the image intensifier. An example of this model is shown in Example 3.4.

The second model was developed by Mery and Filbert in [11, 12], in which a hyperbolic surface is used to model the input screen of the image intensifier [13] that is defined by:

$$\bar{Z} = S(\bar{X}, \bar{Y}) = f\sqrt{1 + (\bar{X}/a)^2 + (\bar{Y}/b)^2}, \tag{3.27}$$

with f (the focal length of the X-ray projection) being the real half axis of the hyperboloid; and a and b the imaginary half axis. The projection of point M onto the input screen of the image intensifier is denoted by p. It is calculated as the intersection of the line that contains points C, M, and m with the 3D surface S (see Fig. 3.7). Its coordinates are given by: $x' = \bar{x}/k(\bar{x}, \bar{y})$ and $y' = \bar{y}/k(\bar{x}, \bar{y})$, with $k(\bar{x}, \bar{y}) = \sqrt{1 - (\bar{x}/a)^2 - (\bar{y}/b)^2}$. The point p is imaged at the CCD camera as w, whose coordinates can be estimated approximately using an affine transformation [1]:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} +\cos(\alpha) & +\sin(\alpha) & 0 \\ -\sin(\alpha) & +\cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{x}/k(\bar{x}, \bar{y}) \\ \bar{y}/k(\bar{x}, \bar{y}) \\ 1 \end{bmatrix}, \tag{3.28}$$

where α represents rotation between $\bar{x}$, $\bar{y}$-, and u, v-axis. This model has only three additional parameters a, b, and α.

Matlab Example 3.4 In Fig. 3.5, the holes of the calibration plate are uniformly distributed in a grid. The horizontal and vertical distance between two consecutive holes is 1 cm. The parameters of the cubic model (3.26) can be obtained using a regression approach as illustrated in the following Matlab code:

Listing 3.4 : Cubic model of an image intensifier

```
% CubicModel.m
close all

I    = Xloadimg('S',2,1);                    % input image of a plate
Data = Xloaddata('S',2,'points');            % centers of mass of the holes
figure
imshow(I,[]);
hold on
um   = Data.ii(:);                            % u coordinate of measured holes
vm   = Data.jj(:);                            % v coordinate of measured holes
plot(vm,um,'go')
xb   = repmat((-6.5:6.5),[11 1]); xb = xb(:); % x bar coordinate of holes in cm
yb   = repmat((-5:5)'   ,[1 14]); yb = yb(:); % y bar coordinate of holes in cm
n    = length(xb);
XX   = [ones(n,1) xb yb xb.*yb xb.^2 yb.^2 yb.*xb.^2 xb.*yb.^2 xb.^3 yb.^3];
a    = regress(um,XX);                        % linear regression for a
b    = regress(vm,XX);                        % linear regression for b
us   = XX*a;                                  % reprojected coordinate u
vs   = XX*b;                                  % reprojected coordinate v
iis = zeros(size(Data.ii)); iis(:) = us;
jjs = zeros(size(Data.jj)); jjs(:) = vs;
plot(vs,us,'r+')
legend({'Detected points','Reprojected points'})
d    = [um-us vm-vs];
plot(jjs,iis,'r:')
plot(jjs',iis','r:')
axis on
err = mean(sqrt(sum(d.*d,2)))                 % mean error in pixels
```

The output of this code is shown in in Fig. 3.9. The detected (or measured) points correspond to the centers of mass of the holes. They were found using an image processing algorithm. Their coordinates are stored in file `plate_points.mat`. The mean error between measured and modeled points is `0.7699` pixels. □

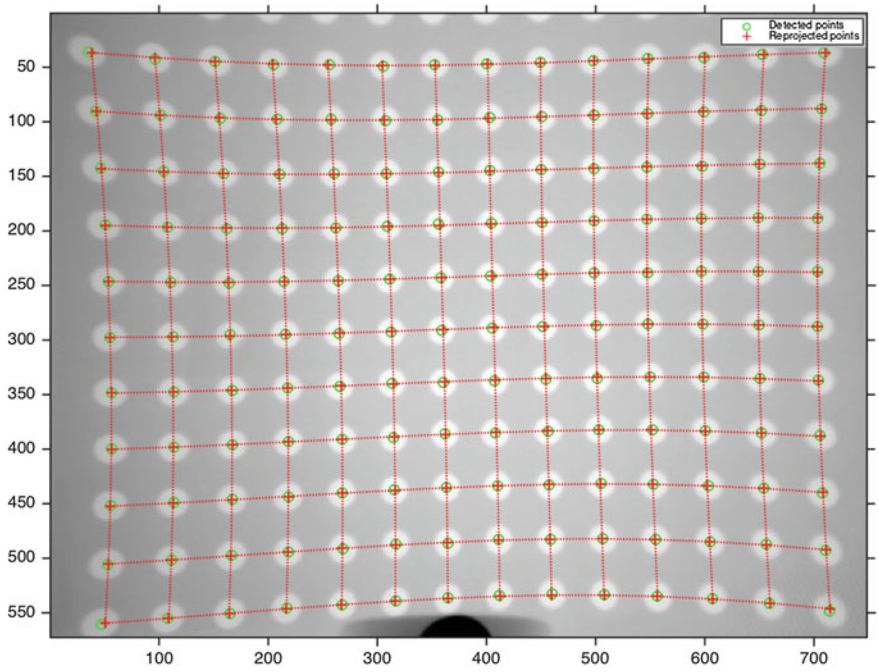


Fig. 3.9 Cubic model of the X-ray projection of regular grid (see Fig. 3.5) (→ Example 3.4)

Matlab Example 3.5 In Fig. 3.10 we show how a 3D point M is projected onto five different X-ray images of an aluminum wheel that has been rotated. The coordinates of M given in the object coordinate system are the same for each projection. In this example, we model the image intensifier using the hyperbolic model explained in this section:

Listing 3.5 : Transformation 3D → 2D using hyperbolic model

```
% ProjectionHyperbolicModel.m
M = [60 60 –40 1]’;                        % 3D point in mm

h  = Xloaddata(’C’,1,’HyperbolicModel.txt’);
P  = Xloaddata(’C’,1,’ManipulatorPosition.txt’);

for p=38:2:46                              % for position p:
    t  = [P(p,1) P(p,2) P(p,3)]’;          % translation vector
    Rp = Xmatrixr3(P(p,4),P(p,5),P(p,6));  % rotation matrix
    Hp = [Rp t;0 0 0 1];                   % 3D Euclidean transformation
    w  = Xhyperproj(M,h,Hp);               % projection using hyperbolic model
    Xloadimg(’C’,1,p,1);                   % display image p
    hold on
    plot(w(2),w(1),’rx’);                  % 2D projection of 3D point M
    pause
end
```

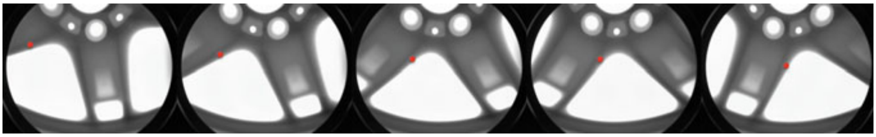


Fig. 3.10 Projection of a 3D point onto 5 different X-ray images of the same object in 5 different positions (→ Example 3.5)

The output of this code is Fig. 3.10. In this code, we use function Xhyperproj (see Appendix B) of 𝕏VIS Toolbox. This function computes the transformation 3D → 2D defined in (3.28). □

3.3.3 *Explicit Geometric Model Using an Image Intensifier*

In this section, we present an explicit model [14] based on the hyperbolic model of Mery and Filbert [11, 12] to perform the transformation PCS → ICS that takes place in the image intensifier and CCD camera. The original hyperbolic model, presented in the previous section, does not take into account the nonlinear projection between input screen and output screen of the image intensifier, because it is considered as an affine transformation. Additionally, there is no decentering point, since in this model the optical axis of the X-ray projection coincides with the optical axis of the image intensifier. Furthermore, the skew factor of the CCD camera is not included. Finally, the distortion that arises when electromagnetic fields are present around the image intensifier is not considered. In this section, we propose a complete model that incorporates the mentioned distortion effects.

Image intensifier

The image intensifier converts the X-ray image into a bright visual image (see Sect. 1.4.3) that can be captured by a CCD camera [15]. Due to the curvature of the input screen of the image intensifier, the X-ray image received at the output screen is deformed, especially at the corners of the image. An additional distortion can be caused by electromagnetic fields that perform a nonlinear projection. An example of these distortion effects is shown in Fig. 3.5, where an X-ray image of a plate containing holes that have been placed in a regular grid manner is illustrated.

First, we will consider a model without electromagnetic field distortion. The geometry of the model used to compute the distortioned perspective projection is shown in Fig. 3.11. It consists of a (curved) input screen S and an output screen Φ, on which the image is projected. The output screen Φ coincides with the retinal plane of this projection.[2] We have shown in Sect. 3.3.1, how the 3D object point M is projected onto plane Π as point m. Thus, the perspective X-ray projection

[2] The reader should note that at this moment there are two retinal planes: Π for the central projection and Φ for the image intensifier.

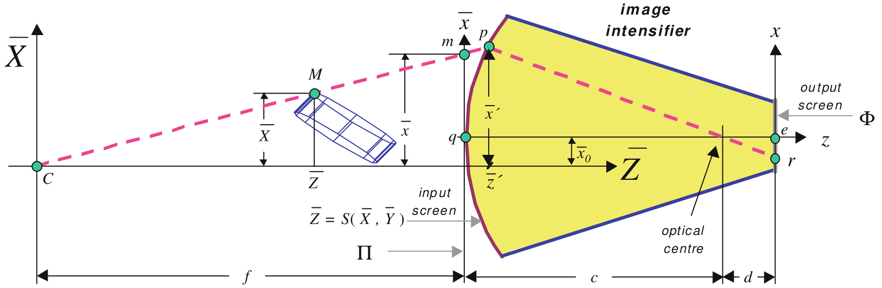


Fig. 3.11 Geometric model of the image intensifier (axis parallel to $\bar{Y}$ are not shown)

OCS → PCS, is given by (3.19). In this section we will calculate, how point m is projected onto input screen S as point p and then onto the retinal plane Φ as point r.

The X-ray image present on the input screen is projected onto the output screen through an optical center of the image intensifier. We may assume without loss of generality that the optical axis of the image intensifier (z-axis) is parallel to the optical axis of the X-ray projection ($\bar{Z}$-axis), because, in a central X-ray projection, there is always a ray that is parallel to the optical axis of the image intensifier. However, the displacement of the axis must be determined. For this reason, we modify the hyperbolic surface of (3.27) by introducing a shift of the center of the hyperboloid as shown in Fig. 3.11. Therefore, the hyperbolic surface S is defined in WCS by:

$$\bar{Z} = S(\bar{X}, \bar{Y}) = f\sqrt{1 + \frac{(\bar{X} - \bar{x}_0)^2}{a^2} + \frac{(\bar{Y} - \bar{y}_0)^2}{b^2}}, \tag{3.29}$$

with f being the real half axis of the hyperboloid; a and b the imaginary half axis; and $(\bar{x}_0, \bar{y}_0)$ the coordinates of the center of the hyperboloid. The focal length of the X-ray projection (f), defined in Sect. 3.3.1, is the minimal value that takes the surface S. This occurs in $(\bar{x}_0, \bar{y}_0)$, that is represented as q in Fig. 3.11. The displacement between $\bar{Z}$- and z-axis is given by $(\bar{x}_0, \bar{y}_0)$.

The projection of point M onto the input screen of the image intensifier is calculated as the intersection of the line that contains points C, M, and m with the 3D surface S. This intersection is denoted by p in Fig. 3.11, whose coordinates in WCS are given by $(\bar{x}', \bar{y}', \bar{z}')$:

$$\bar{x}' = \bar{z}'\bar{x}/f, \qquad \bar{y}' = \bar{z}'\bar{y}/f \qquad \text{and} \qquad \bar{z}' = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \tag{3.30}$$

with

$$A = \tfrac{1}{f^2}\left(1 - \tfrac{\bar{x}^2}{a^2} - \tfrac{\bar{y}^2}{b^2}\right), \quad B = \tfrac{2}{f}\left(\tfrac{\bar{x}\bar{x}_0}{a^2} + \tfrac{\bar{y}\bar{y}_0}{b^2}\right), \quad C = -\left(1 + \tfrac{\bar{x}_0^2}{a^2} + \tfrac{\bar{y}_0^2}{b^2}\right).$$

The coordinates of point p depend on the coordinates $(\bar{x}, \bar{y})$ of point m in PCS. Using homogeneous coordinates, p can be expressed as follows:

$$\mathbf{p} = \mathbf{g}(\mathbf{m}), \tag{3.31}$$

where $\mathbf{p} = [\bar{x}' \ \bar{y}' \ \bar{z}' \ 1]^{\mathsf{T}}$, $\mathbf{m}$ is a homogeneous representation of $(\bar{x}, \bar{y})$, and $\mathbf{g}$ is the nonlinear function defined from (3.30).

As illustrated in Fig. 3.11, point p is projected through the optical center of the image intensifier onto the output screen Φ as point r. The projected point r has coordinates (x, y) in a new 2D coordinate system, called the *output screen coordinate system* (SCS). This coordinate system is centered in e, and its x, y-axis is parallel to the $\bar{x}$, $\bar{y}$-axis of PCS. We can conclude from consideration of similar triangles that:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & -d\bar{x}_0 \\ 0 & d & 0 & -d\bar{y}_0 \\ 0 & 0 & 1 & -(f+c) \end{bmatrix} \begin{bmatrix} \bar{x}' \\ \bar{y}' \\ \bar{z}' \\ 1 \end{bmatrix}, \tag{3.32}$$

where λ is a scale factor, and c and d are the distances of the input and output screen to the optical center of the image intensifier (see Fig. 3.11). This equation can be expressed in matrix form as:

$$\lambda \mathbf{r} = \mathbf{D}\mathbf{p}, \tag{3.33}$$

where the 3-component vector $\mathbf{r}$ is a homogeneous representation of (x, y), and $\mathbf{D}$ is the 3×4 projective matrix of the image intensifier expressed in (3.32). From (3.31) and (3.33) we obtain the nonlinear equation, which depends on six parameters: a, b, c, d, $\bar{x}_0$, and $\bar{y}_0$, that maps a projected point on the retinal plane Π of the X-ray projection onto a point on the retinal plane Φ of the image intensifier:

$$\lambda \mathbf{r} = \mathbf{D}\mathbf{g}(\mathbf{m}). \tag{3.34}$$

To model the effect of the electromagnetic distortion we propose an empirical model, in which a point r on plane Φ will be transformed into a new point r'. We observed that the projection of a regular grid seems to have an additional harmonic signal (see Fig. 3.5). For this reason, we can empirically model this distortion with sinusoidal functions.

The electromagnetic distortion is modeled in two steps. The first step introduces a distortion in the x direction and the second one in the y direction. Thus, x is first transformed into x' from (x, y) and second, y is transformed into y' from (x', y) as follows:

$$\begin{aligned} x' &= x + A_1 \sin(B_1 y + C_1) \\ y' &= y + A_2 \sin(B_2 x' + C_2) \end{aligned} \tag{3.35}$$

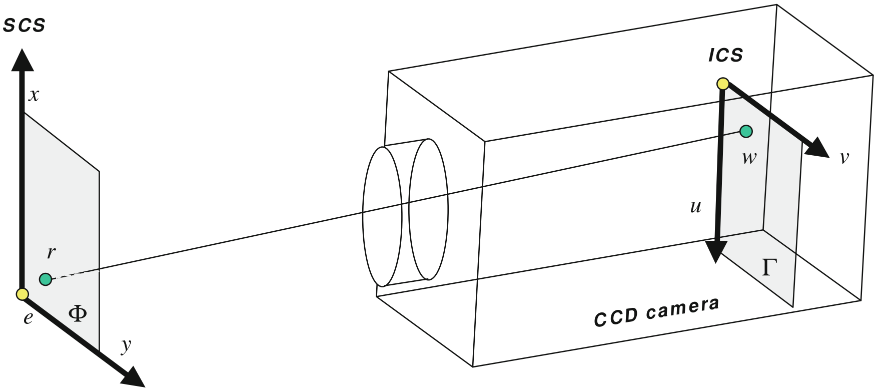


Fig. 3.12 Imaging process in the CCD camera

where A_i, B_i, and C_i, $i = 1, 2$ are the parameters of the electromagnetic distortion model. Formally, r' can be expressed using homogeneous coordinates as follows:

$$\mathbf{r}' = \mathbf{f}(\mathbf{r}) = [x' \ y' \ 1]^{\mathsf{T}}, \tag{3.36}$$

where $\mathbf{f}$ is the nonlinear function defined from (3.35).

Other sinusoidal functions can be used to model the distortion introduced by electromagnetic fields. The reason why we use a two-step-based model is because equation (3.35) can be back-projected in a closed form as shown previously.

CCD camera

The 2D image coordinate system (ICS) is used to represent the pixel coordinates of the X-ray image captured by the CCD camera. The point r (or r' if we consider the electromagnetic field distortion) at the output screen of the image intensifier (see Fig. 3.11) is projected onto the retinal plane Γ of the CCD array[3]as point w as shown in Fig. 3.12.

The camera could be modeled as a general pinhole camera [1], in which a projective mapping from a 3D point of the space to a 2D projective space takes place. However, in our model the 3D points to be mapped belong to a plane, namely the retinal plane Φ. For this reason, in this work we use a homography, i.e., a 2D → 2D general projective transformation as explained in Sect. 3.2.2, which relates the coordinates of retinal plane Φ to retinal plane Γ of the camera. This transformation is defined by:

$$\lambda\mathbf{w} = \mathbf{H}\mathbf{r}, \tag{3.37}$$

where the 3-component vectors $\mathbf{r}$ and $\mathbf{w}$ are homogeneous representations of (x, y) and (u, v) (coordinates of r in SCS and w in ICS), respectively. Matrix $\mathbf{H}$ is a homogeneous 3×3 matrix that causes a general perspective transformation where rotation,

[3] The reader should note that Γ, the retinal plane of the CCD camera, is the third retinal plane of our model (see footnote 2).

translation, scaling, skew, and perspective distortions are considered. Matrix $\mathbf{H}$ has nine elements where only their ratio is significant, so the transformation is defined by only eight parameters, e.g., h_{11}, h_{12}, ..., h_{32}. Parameter h_{33} can be defined as $h_{33} = 1$, or $\mathbf{H}$ can be constrained by $\| \mathbf{H} \| = 1$ [2].

Summary

In this section we described a model which relates the transformation 3D → 2D, from a 3D point M of the test object to a 2D point w of the digitalized X-ray image pixel using homogeneous coordinates. Therefore, the transformation is expressed by $\mathbf{M} \rightarrow \mathbf{w}$, where $\mathbf{M} = [X \ Y \ Z \ 1]^{\mathsf{T}}$ and $\mathbf{w} = [u \ v \ 1]^{\mathsf{T}}$. There are two possibilities for performing the transformation, namely without and with considering the electromagnetic distortion. In the first case, the transformation is given by: $\mathbf{M} \rightarrow \mathbf{m} \rightarrow \mathbf{r} \rightarrow \mathbf{w}$ using Eqs. (3.19), (3.34), and (3.37) respectively:

$$\lambda \mathbf{w} = \mathbf{HDg}(\mathbf{PM}). \tag{3.38}$$

This model has seven extrinsic parameters (defined in (3.20)) and fourteen intrinsic parameters: a, b, c, d, $\bar{x}_0$, $\bar{y}_0$, h_{11}, h_{12}, ..., h_{31}, and h_{32}.

In the second case, where the electromagnetic distortion is modeled, the transformation is expressed by: $\mathbf{M} \rightarrow \mathbf{m} \rightarrow \mathbf{r} \rightarrow \mathbf{r}' \rightarrow \mathbf{w}$ using Eqs. (3.19), (3.34), (3.36), and (3.37), respectively:

$$\lambda \mathbf{w} = \mathbf{HDf}(\mathbf{g}(\mathbf{PM})). \tag{3.39}$$

In comparison with the first case model, the consideration of the electromagnetic distortion requires six more intrinsic parameters: A_i, B_i, and C_i, for $i = 1, 2$.

3.3.4 Multiple View Model

In many applications, a single view of a test object is not enough because there are, for example, occluded parts or intricate projections that cannot be observed with a single view. For this reason the test object must be analyzed from n points of views (with $n \geq 2$). In this section, we present a geometric model that can be used when dealing with *multiple views*, i.e., a geometric model that relates the transformation of a 3D point of the test object into the 2D coordinates of each X-ray projection. For multiple view analysis, i.e., 3D reconstruction or analysis of a part from different points of view, it is required that the n geometric models must share the same 3D object coordinate system (OCS). That means the 3D coordinates (X, Y, Z) of each projection, for $p = 1 \ldots n$, are the same, and we are interested to find the location of the projection of this unique 3D point in each 2D view. Using (3.22) for each view we obtain:

$$\begin{cases} \mathbf{w}_1 = \mathbf{F}(\mathbf{M}, \theta_1) \\ \mathbf{w}_2 = \mathbf{F}(\mathbf{M}, \theta_2) \\ \quad \vdots \\ \mathbf{w}_n = \mathbf{F}(\mathbf{M}, \theta_n) \end{cases} \tag{3.40}$$

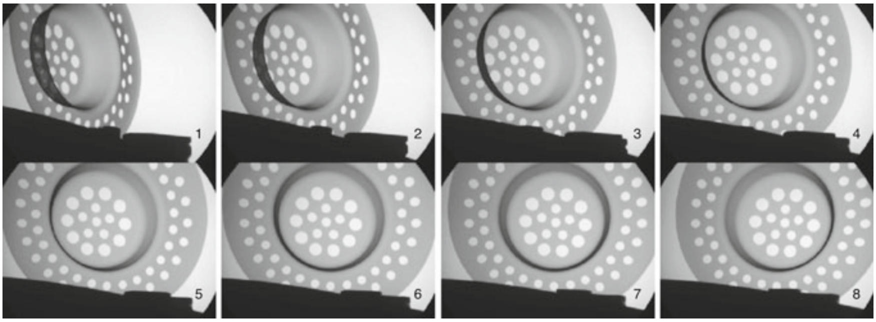


Fig. 3.13 Multiple views of an object acquired using a manipulator that rotates the object around its vertical axis (series S0007 of 𝔾𝔻𝕏ray)

where $\mathbf{M} = [X \; Y \; Z \; 1]^\mathsf{T}$ are the homogenous coordinates of the 3D points in OCS, θ_p are the parameters of the geometric model for pth projection, and $\mathbf{w}_p = [u_p \; v_p \; 1]^\mathsf{T}$ are the homogenous coordinates in pixels in pth X-ray image.

Two views can be simultaneously achieved using two different X-ray detectors. There are some X-ray computer vision systems with three or four detectors as well (see, for example, [16]). In medicine for instance, it is a common practice to take two X-ray images simultaneously (from two different points of views) of certain organs that change their shape and size because they are in motion (e.g., X-ray stereo angiography [17]). In this case, we have independent geometric models (one for each view) that can be obtained using the theory outlined in the previous section. That means, in (3.40), the parameters of the models are independent from each other. Usually in X-ray testing, however, there is a manipulator that is able to locate the test object in different positions, and different views are obtained in different times using a single detector. Given that we are capturing X-ray images of a rigid test object, it is not necessary to acquire the images simultaneously (see Fig. 3.13). In this case, the parameters of the models are not independent from each other because they share the same intrinsic parameters as there is only one X-ray detector. For example, if a manipulator rotates the test object as shown in Fig. 3.6, and for each position a new X-ray image is acquired, it is clear therefore that the transformation from OCS to word coordinate system (WCS) is different for each projection, however, the projection from WCS into image coordinate system (ICS) is exactly the same.

In an X-ray computer vision system with a manipulator and a flat panel (that can be modeled by a simple model with no distortion), the following equation can be used for pth view ($\mathbf{w}_p = \mathbf{F}(\mathbf{M}, \theta_p)$) according to (3.18) and (3.24):

$$\lambda_p \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & s & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{R}_p & \mathbf{t}_p \\ \mathbf{0} & 1 \end{bmatrix}}_{\mathbf{H}_p} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \qquad (3.41)$$

where 4 × 4 matrix $\mathbf{H}_p$, that includes rotation matrix $\mathbf{R}_p$ and translation vector $\mathbf{t}_p$, defines the Euclidean transformation of pth position from OCS to WCS. In this simple model, the vector parameter θ_p for pth projection includes intrinsic parameters (k_u, k_v, s, u_0, v_0) and extrinsic parameters focal length f for the central projection, $(\omega_{Xp}, \omega_{Yp}, \omega_{Zp})$ for the pth rotation, and (t_{Xp}, t_{Yp}, t_{Zp}) for the pth translation. It is clear, that the intrinsic parameters are the same for each projection whereas the extrinsic parameters are different. In many cases, however, the test object is rotated around one axis only, that means (t_{Xp}, t_{Yp}, t_{Zp}) is constant and only one ω angle changes for each position. This is the case of example of Fig. 3.13 where $\omega_{Xp} = \omega_X$, $\omega_{Yp} = \omega_Y$ and $\omega_{Zp} = \omega_Z + p5°$, where the rotation around the vertical axis between two consecutive frames is 5°.

The transformation defined by $\mathbf{H}_p$ (OCS → WCS) can be modeled using a manipulator coordinate system (MCS): OCS → MCS → WCS. Thus, for pth view the transformation OCS → MCS is constant (with some constant rotation and translation), whereas MCS → WCS has a different rotation and translation for each position.

This methodology can be extended to more complex manipulators with several degrees of freedom. For example, in Fig. 3.14 such a manipulator system is presented. This system can be used in the inspection of aluminum castings. The object can be rotated around its vertical axis using rotation R. In addition, it can be translated in X and Z direction using the manipulator table. Moreover, the whole computer vision system can be translated in Y and rotated using rotation T. In order to model the transformation OCS → WCS of this manipulator system, we can include additional coordinate systems as illustrated in Fig. 3.15: OCS → MCS → SCS → WCS. Thus, there are three 3D Euclidean transformations. Each one is modeled

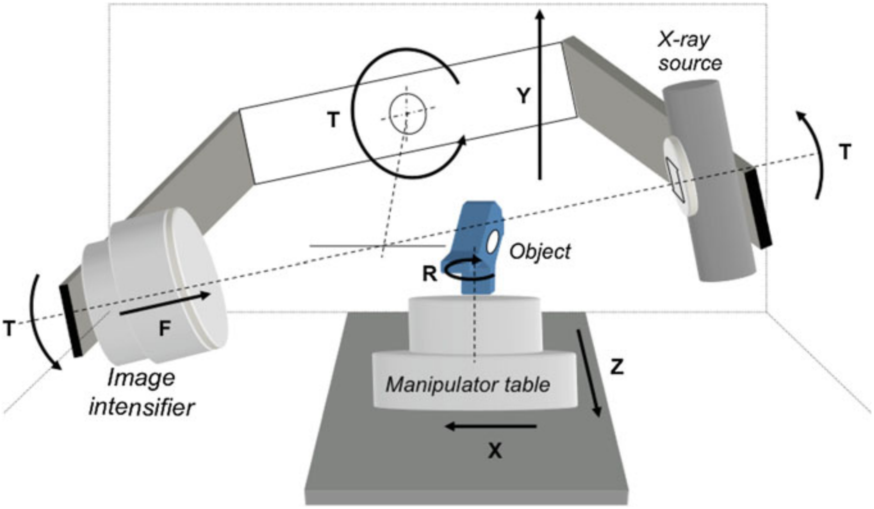


Fig. 3.14 X-ray computer vision system with a manipulator with several degrees of freedom

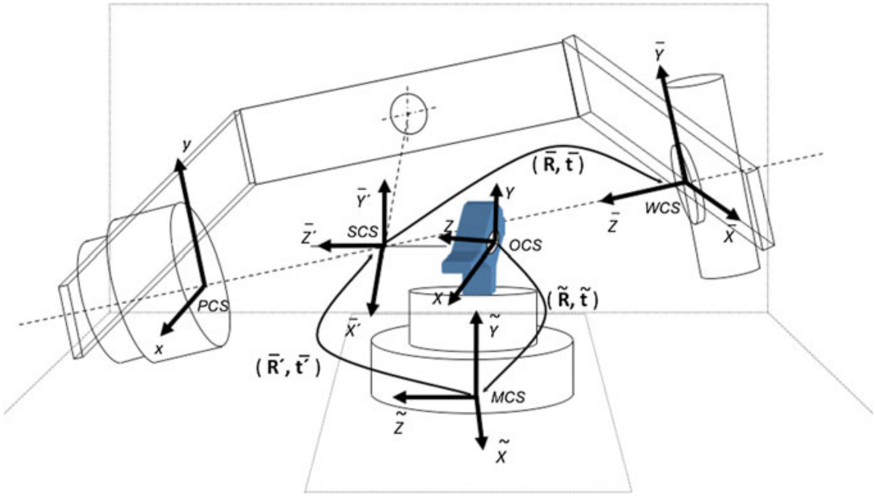


Fig. 3.15 Coordinate systems used in the geometric model of Fig. 3.14

using a 4 × 4 transformation matrix that includes a 3 × 3 rotation matrix and a 3 × 1 translation vector as explained in Sect. 3.2.3. That means, the whole transformation OCS → WCS is a 4 × 4 matrix computed as the multiplication of these three matrices. This matrix corresponds to $\mathbf{H}_p$ in (3.41). The reader can find more details of this model in [18].

3.4 Calibration

The calibration of an X-ray computer vision system—in the context of 3D machine vision—is the process of estimating the parameters of the model, which is used to determine the projection of the 3D object under test into its 2D digital X-ray image. This relationship 3D → 2D can be modeled with the transfer function $\mathbf{F} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ expressed in (3.22).

There are several techniques developed to calibrate a computer vision system. They can be roughly classified into two categories: *photogrammetric calibration* and *self-calibration* [19]. The first one is a 3D reference object-based calibration, where the calibration is performed by observing a calibration object whose geometry in 3D space is known with high accuracy [1]. The second technique uses the identification of matching points in several views of a scene taken by the same camera. Self-Calibration does not use a calibration object with known 3D geometry because it aims to identify the intrinsic parameters of the computer vision system and to reconstruct 3D structure up to a scale similarity [20]. Due to the high precision feature measurement of 3D geometry required in the NDT applications, it

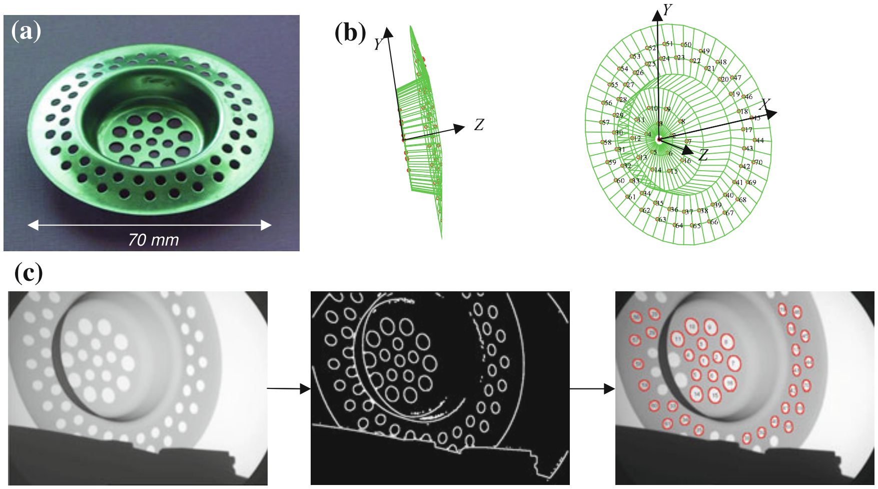


Fig. 3.16 Calibration object: **a** photography; **b** CAD model; and **c** X-ray image of the calibration object and measured calibration points

would be necessary to do a *true* reconstruction of the 3D space without a scale factor. For this reason, usually the calibration technique in X-ray testing belongs to the photogrammetric category.[4]

In calibration, we estimate the parameters of the model based on n points of a *calibration object* whose object coordinates $\mathbf{M}_i = [X_i \; Y_i \; Z_i \; 1]^\mathsf{T}$ are known and whose image coordinates $\tilde{\mathbf{w}}_i = [\tilde{u}_i \; \tilde{v}_i \; 1]^\mathsf{T}$ are measured, for $i = 1, \ldots, n$. In Fig. 3.16 an example of a calibration object is illustrated.

Using the geometric model explained in Sect. 3.3 (see (3.22)), we obtain the *reprojected* points $\mathbf{w}_i = [u_i \; v_i \; 1]^\mathsf{T}$, i.e., the inferred projections in the digital image computed from the calibration points $\mathbf{M}_i$ and the parameter vector θ:

$$\mathbf{w}_i = \gamma(\mathbf{A}\mathbf{M}_i) := \mathbf{F}(\theta, \mathbf{M}_i) \quad \text{for } i = 1 \ldots n \tag{3.42}$$

where $\theta^\mathsf{T} = [\theta_{ext} \; ; \; \theta_{int}]$ is the vector of parameters involved in the projection model including both extrinsic and intrinsic parameters. The parameter vector is then estimated by minimizing the distance between measured points $\tilde{\mathbf{w}}_i$ (see Fig. 3.16c) and inferred points $\mathbf{w}_i = \mathbf{F}(\theta, \mathbf{M}_i)$ (see Fig. 3.16b). Thus, the calibration is performed by minimizing the objective function $\mu(\theta)$ defined as the mean-square discrepancy between these points:

$$\mu(\theta) = \frac{1}{n} \sum_{i=1}^{n} \parallel \tilde{\mathbf{w}}_i - \mathbf{F}(\theta, \mathbf{M}_i) \parallel \rightarrow \text{min}. \tag{3.43}$$

[4] Nevertheless, in Sect. 8.4.3 the reader can find an interesting X-ray testing application where the 3D model is estimated using a self-calibration method based on *bundle adjustment*.

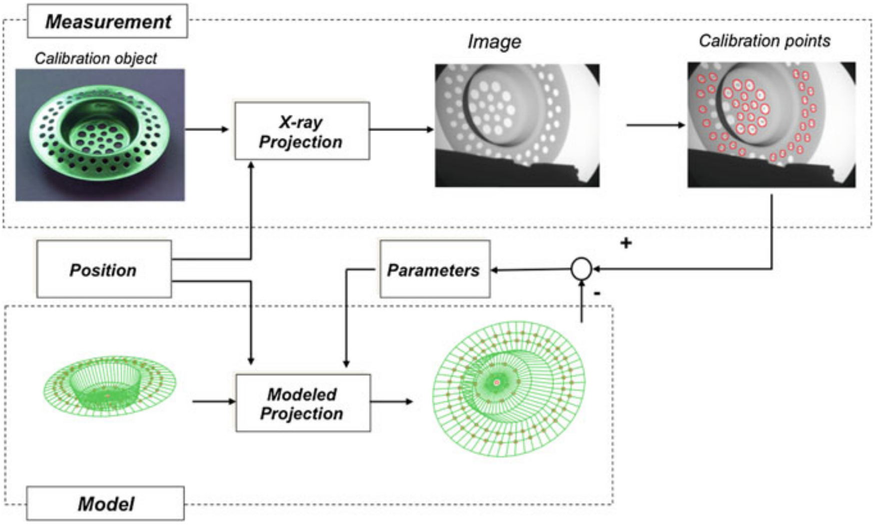


Fig. 3.17 Calibration process

The whole calibration process is summarized in Fig. 3.17. The calibration problem is a nonlinear optimization problem. Generally, the minimization of $\mu(\theta)$ has no closed-form solution. For this reason, the objective function must be iteratively minimized starting with an initial guess θ° that can be obtained from nominal values or preliminary reference measurements. In this section, we present two different methodologies that can be used to calibrate an X-ray computer vision system. The first one was proposed originally in [21] and it is implemented in the Computer Vision Toolbox of Matlab [22]. This technique is very effective and it can be used in the calibration of computer vision with flat panels or with image intensifiers with low distortion. The second technique was proposed in [14] and can be used in computer vision systems with image intensifiers with high distortion.

3.4.1 Calibration Using Matlab

This method can be used to easily calibrate an X-ray computer vision system [22]. It requires a checkerboard as calibration object, and at least two X-ray images taken by the computer vision system to be calibrated. For best results, however, it is recommended to acquire between 10 and 20 images. An example of 18 X-ray images of a calibration pattern is shown in Fig. 2.15.

Matlab Example 3.6 For the calibration of an X-ray computer vision system the X-ray images of Fig. 2.15 and the following Matlab code can be used. In this example, a 3D Gaussian bell is superimposed onto an X-ray image in order to show how we can use the obtained geometric model to reproject 3D points onto the original image.

Listing 3.6 : Calibration of an X-ray computer vision system.

```
% Calibration.m

sd = [ Xgdxdir(’S’,1) ’S0001_00’];
% Define images to process
imageFileNames = {[sd ’01.png’],[sd ’02.png’],[sd ’03.png’],[sd ’04.png’],...
    [sd ’05.png’],[sd ’06.png’],[sd ’07.png’],[sd ’08.png’],[sd ’09.png’],...
    [sd ’10.png’],[sd ’11.png’],[sd ’12.png’],[sd ’13.png’],[sd ’14.png’],...
    [sd ’15.png’],[sd ’16.png’],[sd ’17.png’],[sd ’18.png’]};

% Detect checkerboards in images
[imagePoints, boardSize, imagesUsed] = detectCheckerboardPoints(imageFileNames);
imageFileNames = imageFileNames(imagesUsed);

% Generate world coordinates of the corners of the squares
squareSize  = 25;  % in units of ’mm’
worldPoints = generateCheckerboardPoints(boardSize, squareSize);

% Calibrate the camera
[cameraParams, imagesUsed, estimationErrors] = estimateCameraParameters(imagePoints,
     worldPoints, ...
    ’EstimateSkew’, false, ’EstimateTangentialDistortion’, false, ...
    ’NumRadialDistortionCoefficients’, 2, ’WorldUnits’, ’mm’);

% View reprojection errors
figure; showReprojectionErrors(cameraParams, ’BarGraph’);
figure; showReprojectionErrors(cameraParams, ’ScatterPlot’)

% Visualize pattern locations
figure; showExtrinsics(cameraParams, ’CameraCentric’);

% Display parameter estimation errors
displayErrors(estimationErrors, cameraParams);

% Example: Superimposition of 3D Gaussian bell onto image #6
i      = 6;
I      = imread(imageFileNames{i});

% Projection matrix of image i
R      = cameraParams.RotationMatrices(:,:,i); % Rotation matrix
t      = cameraParams.TranslationVectors(i,:); % Translation vector
P      = cameraMatrix(cameraParams, R, t)’;    % Projection matrix

GaussianSuperimposition(I,P,squareSize)
```

The output of this code is shown in Figs. 3.18, 3.19 and 3.20. The reader who is interested in the computer graphics details of the superimposition can study the program `GaussianSuperimposition` that can be found in `geo` folder. □

3.4.2 Experiments of Calibration

In this section, we present the experiments which we did in order to evaluate the performance of the different models used to calibrate X-ray computer vision systems. The tested models and their principal features are summarized in Table 3.3. They are the seven models outlined in Sect. 3.3.2 and the two proposed models of Sect. 3.3.3 (without and with considering electromagnetic distortion). In the presentation of the

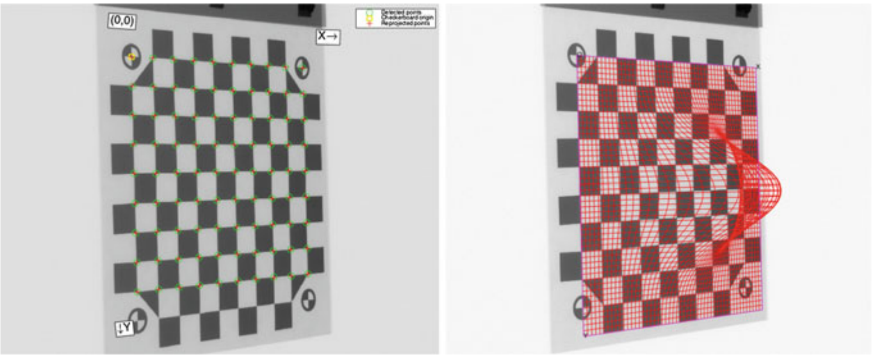


Fig. 3.18 Details of image 6: *left* measured and reprojected points, *right* simulation of a 3D Gaussian bell superimposed onto the checkerboard (→ Example 3.6)

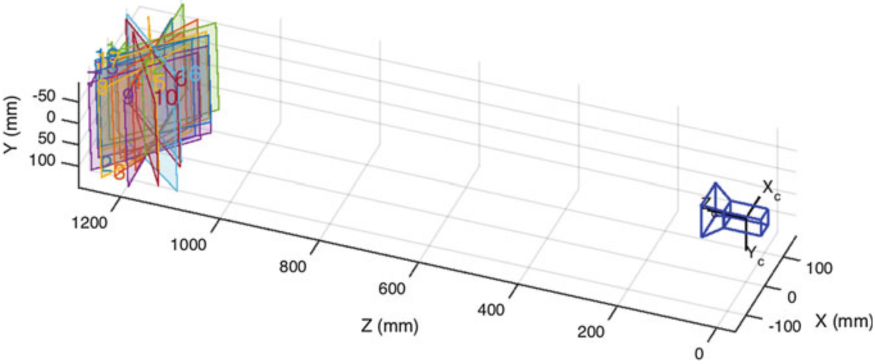


Fig. 3.19 Camera centric view of the planes of the calibration patterns (→ Example 3.6)

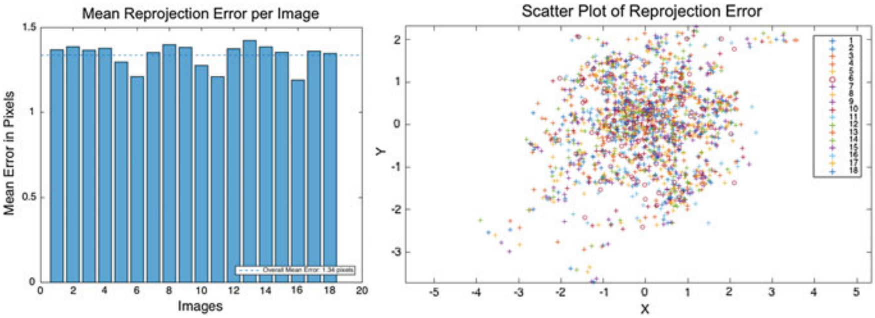


Fig. 3.20 Reprojection error in pixels of the calibration (→ Example 3.6)

results, each model will be identified by the name given in the second column of Table 3.3.

As we explained in the introduction of Sect. 3.4, the calibration process estimates the parameters of a model based on points whose object coordinates are known, and whose image coordinates are measured. The calibration object used in our experiments is shown in Fig. 3.16a. It is an aluminum object with an external diameter of 70 mm. A CAD model was developed by measurement of the calibration object (see Fig. 3.16b). It has seventy small holes ($\phi = 3$–5 mm) distributed on four rings and the center. As we can see in Fig. 3.16, the centers of gravity of the holes are arranged in three heights.

The search for the calibration points within the X-ray image is carried out with a simple procedure that detects regions with high contrast and defined size for the area. The centers of gravity of the detected regions, computed with subpixel accuracy, are defined to be the calibration points. Only complete enclosed regions will be segmented. Figure 3.16c shows an example of the search for the calibration points within an X-ray image. The reader can use series `S0007` of $\mathbb{GDX}$ray with detected points stored in file `ground_truth.txt` and 3D points in file `points_object_3D.txt`. The correspondence between the 3D object points and their images was established manually. The image intensifier used in the experiments was the XRS 232[5] with a 22 cm input screen. The size of the images was 576 × 768 pixels.

In our experiments, the calibration object was placed in different positions using a manipulator. The positions of the calibration object were achieved by rotating one of the axis of the manipulator. Some of the images obtained are illustrated in Fig. 3.21. In order to incorporate the pth position of the manipulator (for each X-ray image) into the geometric model, we modify Eq. (3.18) by:

$$\lambda \begin{bmatrix} \bar{x} \\ \bar{y} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{bmatrix} \bar{\mathbf{H}}_p & \bar{\mathbf{t}}_p \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}}_{\bar{\mathbf{H}}_p} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{3.44}$$

In this equation, we have two 4 × 4 matrices ($\mathbf{H}$ and $\bar{\mathbf{H}}_p$) that define, respectively, two 3D Euclidean transformations: (i) between object and manipulator coordinate systems, and (ii) between manipulator and world coordinate systems. The translation vectors ($\mathbf{t}$ and $\bar{\mathbf{t}}_p$) and the rotation matrices ($\mathbf{R}$ and $\bar{\mathbf{R}}_p$) are related to the corresponding translation and rotation of the mentioned transformations. Since the calibration object is fixed to manipulator, matrix $\mathbf{H}$ is constant for each position: $\mathbf{t} = [t_X \; t_Y \; t_Z]^\mathsf{T}$ and a matrix $\mathbf{R}$ is calculated from the Euler angles ω_X, ω_Y, and ω_Z (see (3.13)). However, matrix $\bar{\mathbf{H}}_p$ depends on the position of the manipulator with respect to the world coordinate system. Matrix $\bar{\mathbf{H}}_p$ is defined by a translation vector $\bar{\mathbf{t}}_p = [\bar{t}_X \; \bar{t}_Y \; \bar{t}_Z]_p^\mathsf{T}$ and a rotation matrix $\bar{\mathbf{R}}_p$ computed from the Euler angles $\bar{\omega}_X$, $\bar{\omega}_Y$,

[5] Image intensifier developed by YXLON International Inc.

Table 3.3 Characterization of the implemented models for calibration

Model	Name	References	Intrinsic parameters	Back-Projection	Calibration	Distortion	Model
1. Faugeras	Linear	[1]	5	Direct	Iterative	None	Explicit
2. Tsai	Radial	[3]	5	Indirect	Iterative	Radial	Implicit
3. Weng et al.	Rad-Tan-1	[5]	9	Indirect	Iterative	Radial, tangential	Implicit
4. Heilikkä	Rad-Tan-2	[7]	8	Direct	Iterative	Radial, tangential	Implicit
5. Swaminathan and Nayar	Rad-Tan-3	[8]	8	Indirect	Iterative	Radial, tangential	Implicit
6. Jaeger/Brack et al.	Cubic	[9, 10]	20	Indirect	Iterative	Cubic	Implicit
7. Mery and Filbert	Hyp-Simple	[11, 12]	7	Direct	Iterative	Hyperbolic simple	Explicit
8. Mery (without EFD)	Hyp-Full	[14]	14	Direct	Iterative	Hyperbolic	Explicit
9. Mery 2 (with EFD)	Hyp-EFD	[14]	20	Direct	Iterative	Hyperbolic, sinusoidal	Explicit

EFD electromagnetic field distortion, *Hyp* Hyperbolic, *Rad* Radial, *Tan* Tangential

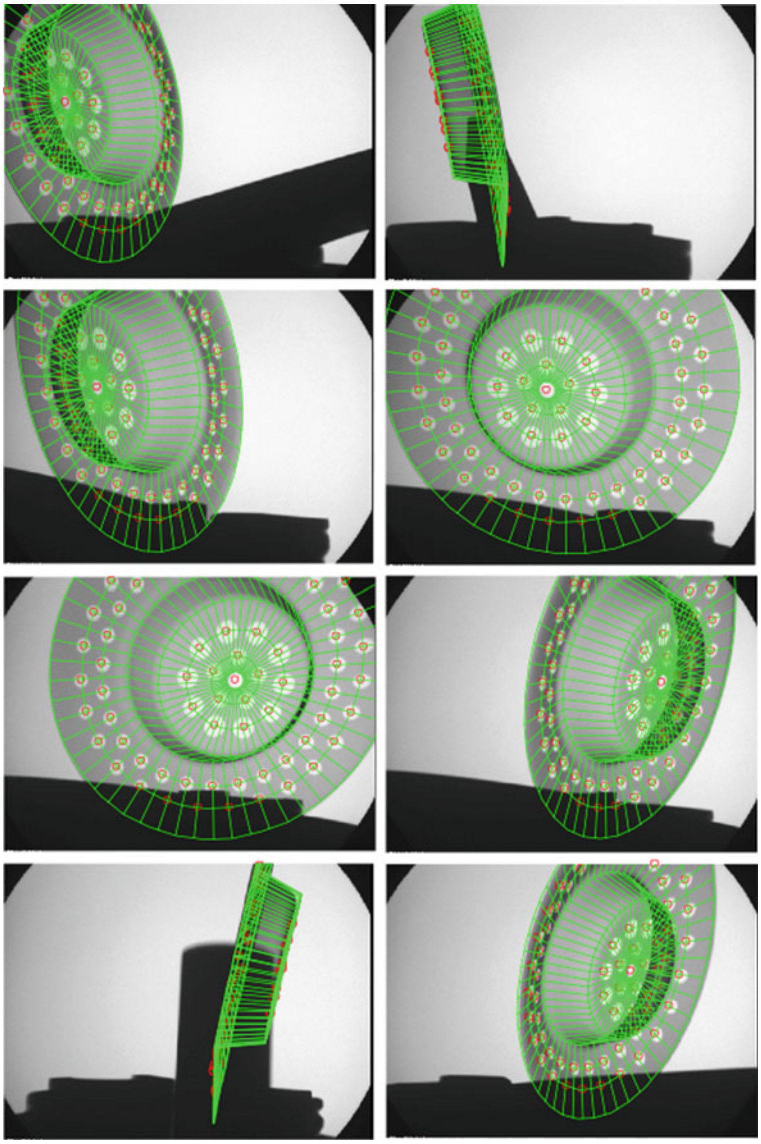


Fig. 3.21 Calibration results using the proposed method Hyp-EFD

and $\bar{\omega}_Z$. In our experiments, $\bar{t}_X$, $\bar{t}_Y$, $\bar{t}_Z$, $\bar{\omega}_X$, and $\bar{\omega}_Y$ were constant. Nevertheless, $\bar{\omega}_Z$ was incremented by the manipulator in constant steps. Thus, the rotation of this axis can be linearly modeled by $\bar{\omega}_Z(p) = \bar{\omega}_{Z0} + p\Delta\bar{\omega}_Z$, where p denotes the number of the position. This new model introduces seven additional extrinsic parameters ($\bar{t}_X$, $\bar{t}_Y$, $\bar{t}_Z$, $\bar{\omega}_X$, $\bar{\omega}_Y$, $\bar{\omega}_{Z0}$, and $\Delta\bar{\omega}_Z$) that must be estimated in the calibration process as well.

The calibration is performed by minimizing the mean reprojection error (μ) computed as the average of the distance between measured points ($\tilde{\mathbf{w}}_{ip}$) and inferred points ($\mathbf{w}_{ip}$)—in the image coordinate system (ICS)—obtained from the pth projection of the ith object point $\mathbf{M}_i$ according to the model of the computer vision system. As we can see, the calibration problem is a nonlinear optimization problem, where the minimization of the objective function has no closed-form solution. For this reason, the objective function must be iteratively minimized starting with an initial estimated value for the parameters involved in the model. In our experiments, the estimation is achieved using the well-known algorithm for minimization problems: the *BFGS Quasi-Newton* method,[6] which is implemented by MathWorks Inc. in the optimization toolbox of MATLAB [23].

We subdivided the calibration points into two groups: the points measured from seven images ($p = 1, 3, 5, \ldots 13$) were used as *control points* to calibrate the computer vision system, whereas the points extracted from seven other images ($p = 2, 4, 6, \ldots 14$) were used as *test points* in order to evaluate the accuracy of calibration.

An example of the calibration using our proposed model, called Hyp-EFD, is illustrated in Fig. 3.21. We can see that the modeled projection of a CAD model of the calibration object coincides with the X-ray image very well. Although points of the top right and the bottom left images of Fig. 3.21 could not be used as control (or test) points because they are very intricate, the inferred projection of the CAD model in these positions seems to be fine.

In order to assess the performance of each model, we carried out two experiments: 2D reprojection and 3D reconstruction. The results are summarized in Table 3.4. The first experiment estimates the parameters of each model by minimizing the average error of the reprojection error—in ICS given in pixels—of the control points. The accuracy is assessed with the reprojection error in the test points. Once the calibration is completed, the second experiment is performed using the parameters estimated in the first. The 3D reconstruction of the measured points was performed using a least square technique [1]. As a performance measurement of the second experiment, the Euclidean distance between measured and reconstructed points in OCS was calculated in millimeters. The mean μ and the standard deviation σ of the computed distances errors in control and test points were tabulated for each experiment. For emphasis, we remind the reader that the calibration is performed by minimizing the average of the reprojection error of the control points (first column

[6] This is a gradient method that uses the Broyden–Fletcher–Goldfarb–Shanno formula for updating the approximation of the Hessian matrix iteratively, which reduces the computational cost of the minimization.

Table 3.4 Error of the models in control (C) and test (T) points

Model ↓	2D reprojection (pixels)				3D reconstruction (mm)			
	μ		σ		μ		σ	
	C	T	C	T	C	T	C	T
1. Linear	2.52	2.45	1.57	1.53	0.47	0.41	0.229	0.157
2. Radial	1.70	1.64	0.96	0.83	0.18	0.17	0.087	0.075
3. Rad-Tan-1	1.40	1.39	0.85	0.76	0.15	0.15	0.076	0.071
4. Rad-Tan-2	1.62	1.66	0.87	0.87	0.25	0.21	0.123	0.092
5. Rad-Tan-3	1.64	1.62	0.92	0.81	0.20	0.18	0.102	0.079
6. Cubic	1.16	1.16	0.67	0.59	0.15	0.14	0.079	0.063
7. Hyp-Simple	1.36	1.40	0.77	0.73	0.18	0.17	0.090	0.074
8. Hyp-Full	1.25	1.29	0.72	0.69	0.17	0.16	0.096	0.072
9. Hyp-EFD	1.18	1.17	0.71	0.64	0.16	0.14	0.084	0.069

in Table 3.4), i.e., the control points in the experiments of 3D reconstruction were not used to calibrate, but also as test points too.

As the results obtained on control and test points are very similar, our analysis will consider test point measurements only. The two values x/y given below correspond to the mean error values obtained by computing the 2D reprojection and 3D reconstruction given in pixels and millimeters, respectively. We observed that the best results were obtained by Cubic and Hyp-EFD models in both experiments. In these cases the mean errors were in the order of $1.16 \sim 1.17/0.14$, i.e., $1.16 \sim 1.17$ pixels for the 2D reprojection, and 0.14 mm for the 3D reconstruction. Although the Cubic model obtains a fractionally better accuracy than model Hyp-EFD (see standard deviations), we must take into account that Cubic model uses an implicit model with 20 parameters for the projection, and 20 other parameters for the back-projection. On the other hand, model Hyp-EFD uses the same 20 parameters for both projection and back-projection.

In our experiments, the X-ray computer vision system could not be adequately modeled without consideration of the lens distortion or with only radial and tangential distortions. The models that were originally developed for cameras (Linear, Radial, Rad-Tan-1, Rad-Tan-2, and Rad-Tan-3, where the mean errors were 2.45/0.41, 1.64/0.17, 1.39/0.15, 1.66/0.21, and 1.62/0.18 respectively), did not work appropriately for our X-ray computer vision system. In many cases, the maximum reprojection error was greater than 4 pixels. The reason for this is that the distortion introduced by the image intensifiers is different from the distortion introduced by a camera lens and the camera models do not consider the electromagnetic distortion in the image intensifier.

On the other hand, hyperbolic models are used by Hyp-Simple, Hyp-Full, and Hyp-EFD. The results obtained with model Hyp-Simple are comparable with the best results obtained from camera models (Rad-Tan-1), where the mean errors were 1.40/0.17 and 1.39/0.15, respectively. In relation to model Hyp-Simple, model

Hyp-Full introduces a decentering point and a nonlinear transformation in the image intensifier. This additional complexity in the model has a significant decrease in the reprojection error (1.40/0.17 vs. 1.29/0.16). In addition, another important reduction of both errors is achieved by considering the electromagnetic field distortion in model Hyp-EFD (1.29/0.16 vs. 1.17/0.14).

We conclude that the proposed explicit model considers the physical parameters of the computer vision system, like image center, focal length, etc., independently. The model is able to map the 3D coordinates of a test object to the 2D coordinates of the corresponding pixel on the digital X-ray image. The model consists of three parts: X-ray projection, image intensifier, and CCD camera. The distortion introduced by the image intensifier was modeled using a hyperbolic surface for the input screen and sinusoidal functions for electromagnetic fields. Using our explicit model, the back-projection function—required for 3D reconstruction—can be calculated directly using a closed-form solution.

The suggested model was experimentally compared with seven other models, which are normally used to calibrate a computer vision system with and without lens distortion. Fourteen X-ray images of a calibration object in different positions were taken. Seven of them were used to calibrate the computer vision system and the other seven were employed to test the accuracy of calibration. The results show that the consideration of only radial and tangential distortions is not good enough if we are working with image intensifiers. In this case, other models must be used for high accuracy requirements. For this reason, Cubic or Hyp-EFD models are recommended. Their mean errors are very similar as shown in Table 3.4. However, for the back-projection, it is convenient to use the proposed model Hyp-EFD because the same parameters are used for both the projection and the back-projection model.

3.5 Geometric Correspondence in Multiple Views

As explained in Sect. 3.3.4, in certain X-ray testing applications it is necessary to analyze multiple views of a test object. In general, in this kind of computer vision applications only the images (2D projections) are available and no 3D information of the test object is known.

In multiple view analysis, it is very important to find *corresponding points* because they can be used for 3D reconstruction, or for analysis of the test object from different points of views. Corresponding points are those 2D points (in different views) that are projections of the same 3D point (see Fig. 3.22). An example for two views is shown in Fig. 3.23, in which we can see two perspective projections of a 3D point M. This stereo rig consists of projection p and projection q. It is built using two monocular perspective projection models (Fig. 3.4). In this example, m_p and m_q are corresponding points because they are projections of the same 3D point M.

In this section, we consider geometric and algebraic constraints to solve the *correspondence problem* between X-ray images obtained as different projections

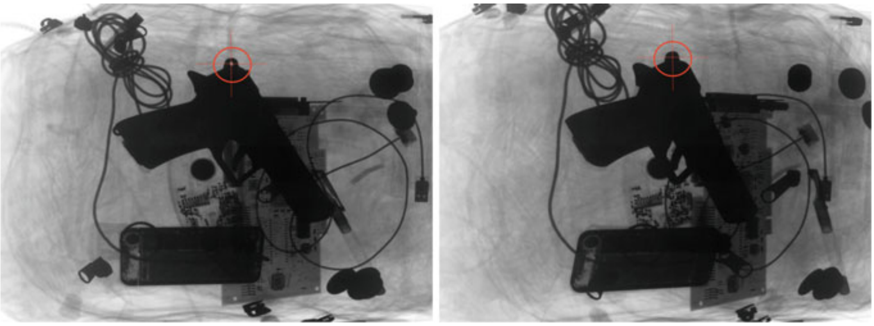


Fig. 3.22 Corresponding points in two different views

of the test object. We will consider the correspondence in two (Sect. 3.5.1), three (Sect. 3.5.2), and more views (Sect. 3.5.3). In order to model the perspective projection in each view, we will use linear model (3.19) with no distortion:

$$\begin{cases} \lambda_p \mathbf{m}_p &= \mathbf{P}_p \mathbf{M} \\ \lambda_q \mathbf{m}_q &= \mathbf{P}_q \mathbf{M} \\ \lambda_r \mathbf{m}_r &= \mathbf{P}_r \mathbf{M} \\ & \vdots \end{cases} \tag{3.45}$$

for different views p, q, $r\ldots$ In general, we assume that there are n views, and indices $p, q, r \ldots \in \{1 \ldots n\}$. It is worth noting that the coordinates of $\mathbf{M}$ are given in the same 3D coordinate system for each projection. That means, $\mathbf{M} = [X\ Y\ Z\ 1]^\mathsf{T}$ in each equation of (3.45).

The correspondence problem with nonlinear projection models will be considered for two views only. The reader, however, will be able to establish correspondences with nonlinear models in more views using the methodology of two views.

3.5.1 Correspondence Between Two Views

Now, the correspondence between two points m_p and m_q (in the X-ray projection coordinate system) is considered. The first point is obtained by projecting the object point M at position p, and the second one at position q:

$$\begin{cases} \lambda_p \mathbf{m}_p &= \mathbf{P}_p \mathbf{M} \\ \lambda_q \mathbf{m}_q &= \mathbf{P}_q \mathbf{M} \end{cases} \tag{3.46}$$

for $\mathbf{M} = [X\ Y\ Z\ 1]^\mathsf{T}$ given in the same 3D coordinate system for each equation. The correspondence problem in two views p and q can be stated as follows: given

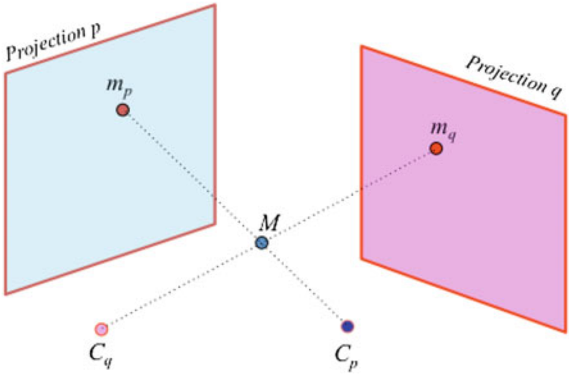


Fig. 3.23 Projection in two views

$\mathbf{m}_p$, $\mathbf{P}_p$, and $\mathbf{P}_q$, is it possible to find $\mathbf{m}_q$? Note that in this problem $\mathbf{M}$ is unknown. Moreover, if we know $\mathbf{m}_p$ and $\mathbf{P}_p$, it is impossible to find an exact location for M. In this section, we will explain a geometric and algebraic approach that can be used to solve the correspondence problem. In addition, the section gives practical considerations and the use of nonlinear projection models.

Epipolar Geometry

We do not know where M is exactly, however, it is known that M lies on the line $\langle C_p, m_p \rangle$ as illustrated in Fig. 3.23. Since m_q is the projection of M onto view q, we can affirm that m_q lies on line ℓ defined as the projection of $\langle C_p, m_p \rangle$ onto view q. Line ℓ is known as the *epipolar line* of m_p in view q.

Thus, to solve the correspondence problem in two views we use *epipolar geometry* [1, 2, 12]. The epipolar constraint is well known in stereo vision: for each projection point m_p at the position p, its corresponding projection point m_q at the position q lies on the epipolar line ℓ of m_p, as shown in Fig. 3.24, where C_p and C_q are the centers of projections p and q, respectively. In this representation, a rotation and translation relative to the object coordinate system is assumed. The epipolar line ℓ can be defined as the projection of line $\langle C_p, m_p \rangle$ by the center of projection C_q onto projection plane q.

Epipolar line can be calculated in three simple steps (see Fig. 3.24):

(i) From m_p and $\mathbf{P}_p$, we find two 3D points M_{p1} and M_{p2} that lies on $\langle C_p, m_p \rangle$. Obviously, one point that lies on $\langle C_p, m_p \rangle$ is C_p, i.e., $M_{p1} = C_p$. Since the location of C_p is unknown, we can find it by considering the following reasoning [2]: it is not possible to project C_p onto view p because C_p is the optical center of this projection. For this reason, if we use the first equation of (3.46) to project $\mathbf{C}_p$ (the homogeneous representation of C_p) we will obtain $\mathbf{P}_p\mathbf{C}_p$. Since this projected point is not defined it can be shown that its homogeneous representation is $[0\ 0\ 0]^\mathsf{T}$. It is not possible to estimate the nonhomogeneous representation of this point because there is a division by zero. For this reason, $\mathbf{P}_p\mathbf{C}_p = [0\ 0\ 0]^\mathsf{T}$. Thus, $\mathbf{C}_p$ can be easily calculated as the null space of $\mathbf{P}_p$: For $\mathbf{A} = \mathbf{P}_p$ and $\mathbf{C}_p = [C_X\ C_Y\ C_Z\ 1]^\mathsf{T}$, $\mathbf{AC}_p = \mathbf{0}$ can be reformulated as:

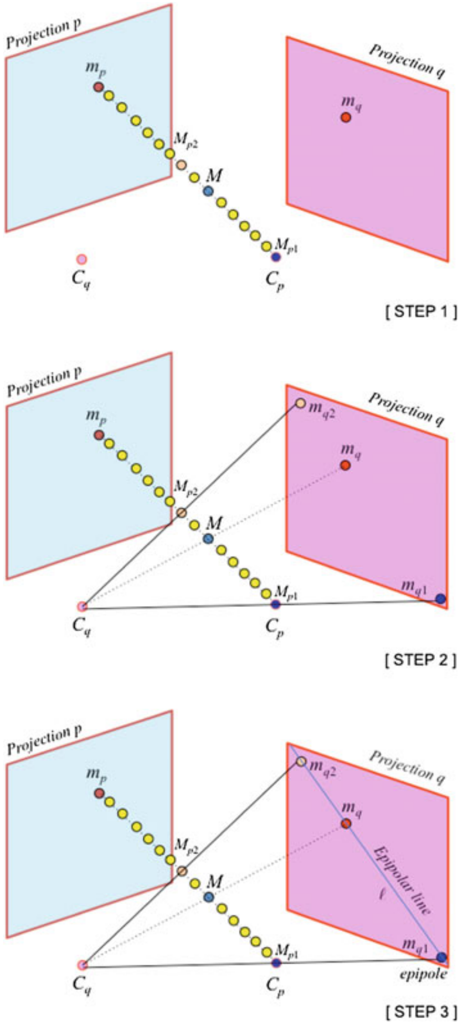


Fig. 3.24 Estimation of epipolar line ℓ in three steps

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}}_{\mathbf{A}_1} \begin{bmatrix} C_X \\ C_Y \\ C_Z \end{bmatrix} = - \underbrace{\begin{bmatrix} a_{14} \\ a_{24} \\ a_{34} \end{bmatrix}}_{\mathbf{a}_4}. \tag{3.47}$$

Then the coordinates of C_p in 3D coordinate system are: $[C_X \; C_Y \; C_Z]^\mathsf{T} = -\mathbf{A}_1^{-1}\mathbf{a}_4$.

The second point should be m_p, however, we do not know the coordinates of m_p in 3D coordinate space, we only know $\mathbf{m}_p = [x_p \; y_p \; 1]^\mathsf{T}$, where (x_p, y_p) are coordinates of m_p in 2D coordinates system of view p. Nevertheless, it can be shown [2], that a point that lies on $\langle C_p, m_p \rangle$ is $\mathbf{M}^+$ calculated as:

$$\mathbf{M}^+ = \mathbf{P}_p^+ \mathbf{m}_p \tag{3.48}$$

where $\mathbf{P}_p^+$ is:

$$\mathbf{P}_p^+ = \mathbf{P}_p^\mathsf{T} [\mathbf{P}_p \mathbf{P}_p^\mathsf{T}]^{-1}. \tag{3.49}$$

The 4×3 matrix $\mathbf{P}_p^+$ is known as the *pseudoinverse* of $\mathbf{P}_p$ because $\mathbf{P}_p \mathbf{P}_p^+ = \mathbf{I}_{3\times 3}$. The reader can demonstrate that the projection of $\mathbf{M}^+$ onto view p is $\mathbf{m}_p$ by substituting (3.48) in the first equation of (3.46).
Thus, the two 3D points that lie on $\langle C_p, m_p \rangle$ are $\mathbf{M}_{p1} = \mathbf{C}_p$ (defined in (3.47)) and $\mathbf{M}_{p2} = \mathbf{M}^+$ (defined in (3.48)).

(ii) From M_{p1} and M_{p2}, we find the projection of them onto view q using $\mathbf{P}_q$. These 2D points will be denoted as m_{q1} and m_{q2}, respectively.
Both 3D points are projected onto view q using the second equation of (3.46):

$$\begin{cases} \lambda_{q1} \mathbf{m}_{q1} = \mathbf{P}_q \mathbf{M}_{p1} = \mathbf{P}_q \mathbf{C}_p \\ \lambda_{q2} \mathbf{m}_{q2} = \mathbf{P}_q \mathbf{M}_{p2} = \mathbf{P}_q \mathbf{P}_p^+ \mathbf{m}_p \end{cases} . \tag{3.50}$$

(iii) From m_{q1} and m_{q2} we find line ℓ which contains them.
Since the epipolar line contains m_{q1} and m_{q2}, line ℓ can be computed in homogeneous coordinates using (3.2):

$$\ell = \mathbf{m}_{q1} \times \mathbf{m}_{q2}. \tag{3.51}$$

The first point $\mathbf{m}_{q1}$, i.e., the projection of C_p onto plane q, as defined in the first equation of (3.50), is the well known *epipole*.[7] The epipolar line is defined as the line that contains the epipole $\mathbf{m}_{q1}$ and the point $\mathbf{m}_{q2}$. We observe that the epipole belongs to any epipolar line obtained from any arbitrary point m_p. In other words, all epipolar lines share a common point: the epipole. Moreover, the epipole does not depend on m_p or m_q. It depends only on the two views geometry.

The projective representation of the epipolar line is obtained by taking the cross product of these two points, i.e., $\ell = \mathbf{m}_{q1} \times \mathbf{m}_{q2}$. Line ℓ can be written using $[\mathbf{m}_{q1}]_\times$, the antisymmetric matrix of $\mathbf{m}_{q1}$, where $\ell = [\mathbf{m}_{q1}]_\times \mathbf{m}_{q2}$. Matrix $[\mathbf{m}_{q1}]_\times$ is defined as the 3×3 matrix such that $[\mathbf{m}_{q1}]_\times \mathbf{s} = \mathbf{m}_{q1} \times \mathbf{s}$ for all vectors $\mathbf{s}$, i.e.,

$$[\mathbf{m}_{q1}]_\times = \begin{bmatrix} 0 & +m_{q1}(3) & -m_{q1}(2) \\ -m_{q1}(3) & 0 & +m_{q1}(1) \\ +m_{q1}(2) & -m_{q1}(1) & 0 \end{bmatrix}.$$

[7] The word *epipole* comes from the Greek ἐπι (*epi*): over and πόλος (*polos*): attractor.

where $\mathbf{m}_{q1} = [m_{q1}(1)\ m_{q1}(2)\ m_{q1}(3)]^{\mathsf{T}}$. Thus, using the antisymmetric matrix, from (3.51) and (3.50), line ℓ can be computed by:

$$\ell = \mathbf{F}_{pq}\mathbf{m}_p \tag{3.52}$$

where $\mathbf{F}_{pq}$ is the *fundamental matrix* known from multiple view computer vision [2, 24] given by:

$$\mathbf{F}_{pq} = [\mathbf{P}_q\mathbf{C}_p]_\times \mathbf{P}_q\mathbf{P}_p^+\mathbf{m}_p. \tag{3.53}$$

Since the point m_q belongs to the epipolar line ℓ, it follows that

$$\mathbf{m}_q^{\mathsf{T}}\ell = \mathbf{m}_q^{\mathsf{T}}\mathbf{F}_{pq}\mathbf{m}_p = 0 \tag{3.54}$$

Equation (3.54) is known as the *epipolar constraint*: If m_p and m_q are corresponding points, then m_q must lie on the epipolar line ℓ of m_p, i.e., $\mathbf{m}_q^{\mathsf{T}}\mathbf{F}_{pq}\mathbf{m}_p$ must be zero.

Matlab Example 3.7 This example shows epipolar lines in two views (p and q). We assume that the projection matrices $\mathbf{P}_p$ and $\mathbf{P}_q$ are known from a calibration process. The code computes the fundamental matrix. We select manually eight m_p points in view p. The code plots the epipolar lines of these points in view q.

Listing 3.7 : Epipolar lines for two views.

```
% EpipolarGeometry.m
close all
Data = Xloaddata('B',44,'Pmatrices'); % projection matrices

p = 1; q = 82;                        % p and q indices
Ip = Xloadimg('B',44,p);
Iq = Xloadimg('B',44,q);
figure(1);imshow(Ip);title('Image p');hold on
figure(2);imshow(Iq);title('Image q');hold on

Pp = Data.P(:,:,p);                   % projection matrix of view p
Pq = Data.P(:,:,q);                   % projection matrix of view q

F = Xfundamental(Pp,Pq);              % fundamental matrix

col = 'bgrcmykw';                     % colors for each point-line pair

for i=1:8
    disp('click a point mp in Figure 1...')
    figure(1);
    mp = ginput(1)';                  % click
    plot(mp(1),mp(2), [col(i) '*'])
    figure(2)
    Xplotepipolarline(F,mp,col(i));   % Epipolar line
end
```

The output of this code is shown in Fig. 3.25. The code uses two functions of 𝕏vis Toolbox: the first one is Xfundamental (see Appendix B) to compute the fundamental matrix and the second one is Xplotepipolarline (see Appendix B) to plot

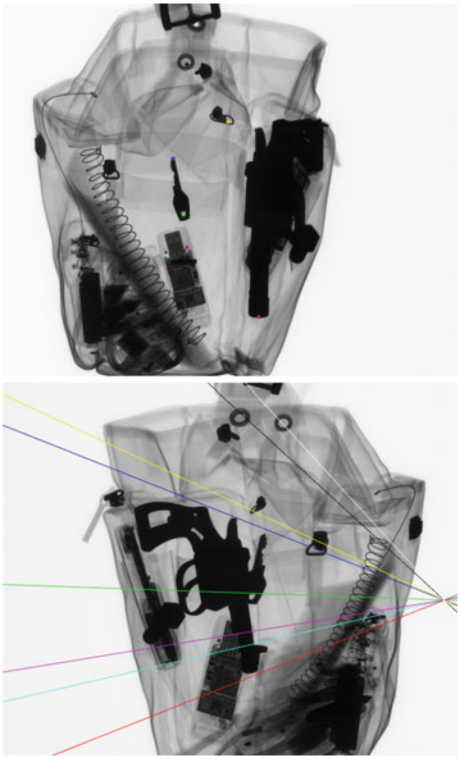


Fig. 3.25 Example of epipolar geometry: *Top view* p with eight points. *Bottom view* q with corresponding epipolar lines. It is clear that the corresponding points in view q lie on the corresponding epipolar lines. The intersection of the epipole lines defines the epipole (→ Example 3.7)

the epipolar lines onto view q. The example uses images $p = 1$ and $q = 82$ of series `B0044` of $\mathbb{GDX}$ray. In this set of images there are 178 different views (taken by rotating the test object around a quasi-vertical axis in 2° between consecutive views). The reader who is interested in other views can change the code in order to define other values for p and q. □

Bifocal tensors

Another way to estimate the epipolar constraint is using *bifocal tensors* [25, 26], as explained next. This can be considered as an algebraic approach. From (3.46) the two projections can be expressed by:

$$\begin{cases} \lambda_p \mathbf{m}_p = \mathbf{P}_p \mathbf{M} := \mathbf{AM} \\ \lambda_q \mathbf{m}_q = \mathbf{P}_q \mathbf{M} := \mathbf{BM} \end{cases} . \tag{3.55}$$

These two equations can also be written as:

$$\underbrace{\begin{bmatrix} \mathbf{a}_1 & x_p & 0 \\ \mathbf{a}_2 & y_p & 0 \\ \mathbf{a}_3 & 1 & 0 \\ \mathbf{b}_1 & 0 & x_q \\ \mathbf{b}_2 & 0 & y_q \\ \mathbf{b}_3 & 0 & 1 \end{bmatrix}}_{\mathbf{G}} \underbrace{\begin{bmatrix} \mathbf{M} \\ -\lambda_p \\ -\lambda_q \end{bmatrix}}_{\mathbf{v}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.56}$$

where $\mathbf{a}_i$ and $\mathbf{b}_i$ denote the ith row of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. If m_p and m_q are corresponding points, then the 3D point M exists. It follows that there must be a nontrivial solution for $\mathbf{v}$ in (3.56), i.e., the determinant of the 6×6 matrix $\mathbf{G}$ must be zero. Expanding the determinant of $\mathbf{G}$ we obtain:

$$|\mathbf{G}| = x_p x_q \begin{vmatrix} \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{vmatrix} - y_p x_q \begin{vmatrix} \mathbf{a}_1 \\ \mathbf{a}_3 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{vmatrix} + x_q \begin{vmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{vmatrix} +$$

$$-x_p y_q \begin{vmatrix} \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{b}_1 \\ \mathbf{b}_3 \end{vmatrix} + y_p y_q \begin{vmatrix} \mathbf{a}_1 \\ \mathbf{a}_3 \\ \mathbf{b}_1 \\ \mathbf{b}_3 \end{vmatrix} - y_q \begin{vmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{b}_1 \\ \mathbf{b}_3 \end{vmatrix} + x_p \begin{vmatrix} \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{vmatrix} - y_p \begin{vmatrix} \mathbf{a}_1 \\ \mathbf{a}_3 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{vmatrix} + \begin{vmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{vmatrix} = 0.$$

that can be expressed by:

$$|\mathbf{G}| = [x_q \; y_q \; 1] \underbrace{\begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix}}_{\mathbf{F}_{pq}} \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \mathbf{m}_q^{\mathsf{T}} \mathbf{F}_{pq} \mathbf{m}_p = 0 \tag{3.57}$$

where $\mathbf{F}_{pq}$ corresponds to the mentioned fundamental matrix of Eq. (3.53) for $\mathbf{A} = \mathbf{P}_p$ and $\mathbf{B} = \mathbf{P}_q$. In this algebraic formulation, the elements of $\mathbf{F}_{pq}$ are called *bifocal tensors* [2]. They can be computed as:

$$F_{ij} = (-1)^{i+j} \begin{vmatrix} \sim\mathbf{a}_j \\ \sim\mathbf{b}_i \end{vmatrix} \qquad \text{for } i, j = 1, 2, 3. \tag{3.58}$$

where $\sim\mathbf{a}_j$ and $\sim\mathbf{b}_i$ mean, respectively, matrix $\mathbf{A}$ without the jth row and matrix $\mathbf{B}$ without the ith row.

Usually, we can express matrix $\mathbf{A}$ in a canonical form:

$$\mathbf{A} = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \end{bmatrix} = [\mathbf{I} \mid \mathbf{0}]. \tag{3.59}$$

The canonical form can be achieved using a general projective transformation of the object coordinate system: $\mathbf{M}' = \mathbf{HM}$, where $\mathbf{M}'$ is the transformation of $\mathbf{M}$, and $\mathbf{H}$ is a 4×4 nonsingular matrix obtained by adding one extra row to $\mathbf{P}_p$ [27]. Thus, Eq. (3.55) can be expressed as:

$$\begin{cases} \lambda_p \mathbf{m}_p = [\mathbf{I} \mid \mathbf{0}]\mathbf{M}' = \mathbf{AM}' \\ \lambda_q \mathbf{m}_q = \qquad\qquad\quad \mathbf{BM}' \end{cases} \tag{3.60}$$

with

$$\begin{array}{ll} \mathbf{M}' & = \mathbf{HM} \\ \mathbf{A} & = \mathbf{P}_p \mathbf{H}^{-1} \\ \mathbf{B} & = \mathbf{P}_q \mathbf{H}^{-1} \end{array}.$$

For the canonical form $\mathbf{A} = [\mathbf{I} \mid \mathbf{0}]$, the bifocal tensors may be expressed by:

$$F_{ij} = b_{i\oplus 1,j} b_{i\oplus 2,4} - b_{i\oplus 2,j} b_{i\oplus 1,4} \tag{3.61}$$

where

$$i \oplus k = \begin{cases} i + k & \text{if } i + k \leq 3 \\ i + k - 3 & \text{otherwise} \end{cases}.$$

Practical considerations

In practice, the projection points m_p and m_q can be corresponding points, if the perpendicular Euclidean distance from the epipolar line ℓ of the point m_p to the point m_q is smaller than a small number ε [28]:

$$d_2 = \frac{|\mathbf{m}_q^{\mathsf{T}} \mathbf{F}_{pq} \mathbf{m}_p|}{\sqrt{\ell_1^2 + \ell_2^2}} < \varepsilon, \tag{3.62}$$

where $\ell = \mathbf{F}_{pq} \mathbf{m}_p = [\ell_1\ \ell_2\ \ell_3]^{\mathsf{T}}$.

An additional criterion to establish the correspondence between two views is that the 3D point reconstructed from the projection points m_p and m_q must belong to the space occupied by the test object [11]. From m_p and m_q the corresponding 3D point $\hat{M}$ can be estimated using 3D reconstruction techniques (see Sect. 3.6). It is necessary to examine if $\hat{M}$ resides in the volume of the test object, the dimensions of which are usually known a priori (e.g., a wheel is assumed to be a cylinder). This criterion implies that the epipolar is delimited as illustrated in Fig. 3.26. It is

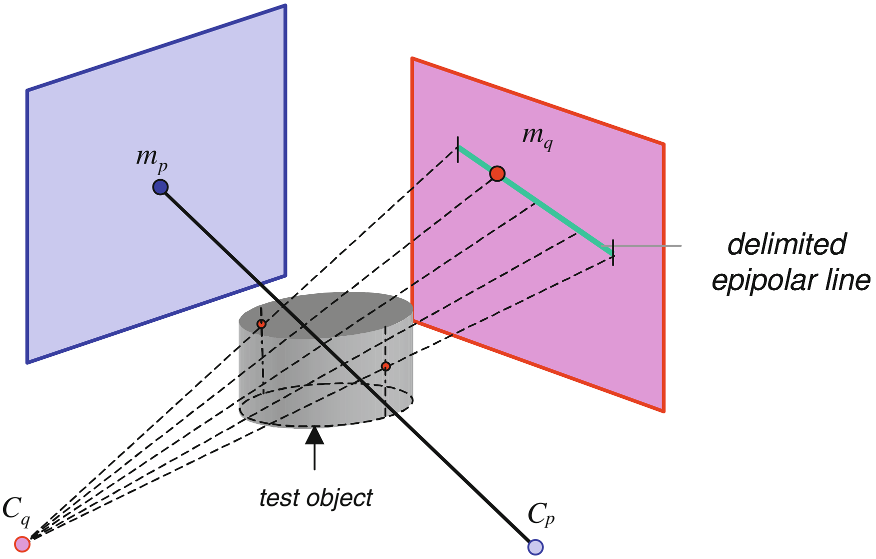


Fig. 3.26 Epipolar geometry in two views using 3D information of the test object

possible to use a CAD model of the test object to evaluate this criterion in a more precise way.

Nonlinear projections

An example of corresponding points using a nonlinear model that considers geometric distortions is illustrated in Fig. 3.27. The idea is simple. The projection model has a linear part (that corresponds to the perspective projection), a nonlinear part (that corresponds to the image intensifier) as illustrated in Fig. 3.6 as PCS and ICS. The epipolar geometry explained in the previous section is defined for the linear part only (for PCS and not for ICS). That means, the epipolar lines are straight lines in PCS, however, they are curves in ICS. In order to use the theory of the epipolar geometry we need the inverse transformation from both coordinate systems: ICS → PCS. Thus a point in ICS defined as $\mathbf{w}$ is transformed into a point in PCS as $\mathbf{m}$ by $\mathbf{m} = \mathbf{f}^{-1}(\mathbf{w})$, where $\mathbf{w} = \mathbf{f}(\mathbf{m})$ is the direct transformation: PCS → ICS in homogeneous coordinates. Using the inverse transformation of this nonlinear model, we can use the epipolar constraint (3.54). Thus the epipolar curves are given by:

$$[\mathbf{f}^{-1}(\mathbf{w}_q)]^{\mathsf{T}}\mathbf{F}_{pq}[\mathbf{f}^{-1}(\mathbf{w}_p)] = 0 \tag{3.63}$$

In the example of Fig. 3.27, the epipolar lines were computed using a hyperbolic model (see Sect. 3.3.3). Other nonlinear model can be used as well.

It is clear that if the transformation between PCS → ICS is linear (for example, using a flat panel), we have $\mathbf{w} = \mathbf{Hm}$. That means, we can substitute $\mathbf{m} = \mathbf{H}^{-1}\mathbf{w}$ in (3.57):

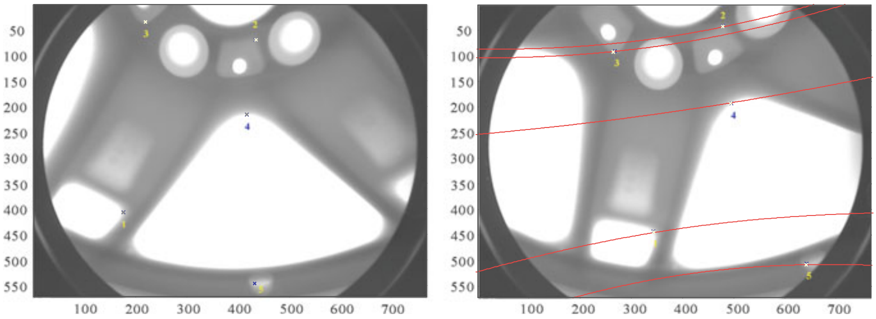


Fig. 3.27 Epipolar lines using a hyperbolic model [12]

$$\mathbf{w}_q^{\mathsf{T}}\,\underbrace{[\mathbf{H}^{-\mathsf{T}}\mathbf{F}_{pq}\mathbf{H}^{-1}]}_{\mathbf{F}'_{pq}}\,\mathbf{w}_p = 0 \tag{3.64}$$

where $\mathbf{F}'_{pq}$ is the fundamental matrix given in ICS.

Nevertheless, using a general nonlinear model, as explained in Sect. 3.3.3, a point w, whose coordinates in ICS are (u, v), can be back-projected onto a point m, whose coordinates in PCS are $(\bar{x}, \bar{y})$. The back-projection is carried out in two steps: transformation ICS → SCS and transformation SCS → PCS.

Transformation ICS → SCS: Without considering the electromagnetic distortion, the transformation ICS → SCS can be directly obtained from (3.37):

$$\lambda\mathbf{r} = \mathbf{H}^{-1}\mathbf{w}. \tag{3.65}$$

However, if the electromagnetic distortion is considered, the inverse function of (3.36) $\mathbf{r} = \mathbf{f}^{-1}(\mathbf{r}')$ must be obtained from (3.35):

$$\begin{array}{l} y = y' - A_2\sin(B_2x' + C_2) \\ x = x' - A_1\sin(B_1y + C_1) \end{array}. \tag{3.66}$$

Therefore, it yields

$$\mathbf{r} = \mathbf{f}^{-1}(\mathbf{H}^{-1}\mathbf{w}). \tag{3.67}$$

Transformation SCS → PCS: The second transformation is nonlinear because it takes into account the geometric distortion of the image intensifier. Given the coordinates (x, y) of point r in SCS, a point p on the surface S (see Fig. 3.11) that is the back-projection of r can be computed by finding the coordinates $(\bar{x}', \bar{y}', \bar{z}')$ that satisfy Eqs. (3.32) and (3.29) simultaneously. The solution is:

$$\bar{x}' = \bar{x}_0 - \tfrac{x}{d}(f + c - \bar{z}')$$
$$\bar{y}' = \bar{y}_0 - \tfrac{y}{d}(f + c - \bar{z}'), \tag{3.68}$$
$$\bar{z}' = \tfrac{-B' + \sqrt{B'^2 - 4A'C'}}{2A'}$$

where

$$A' = \tfrac{g}{f^2} - 1, \quad B' = 2(f + c), \quad C' = -g - (f + c)^2, \quad g = \frac{d^2}{\frac{x^2}{a^2} + \frac{y^2}{b^2}},$$

or using matrix notation:

$$\mathbf{p} = \mathbf{h}(\mathbf{r}), \tag{3.69}$$

where $\mathbf{p} = [\bar{x}' \ \bar{y}' \ \bar{z}' \ 1]^\mathsf{T}$, $\mathbf{r}$ is a homogeneous representation of (x, y), and $\mathbf{h}$ is the nonlinear function defined from (3.68).

Now, the coordinates of the back-projected point m on the projection plane can be calculated from (3.68) and the first two equations of (3.30):

$$\bar{x} = f\bar{x}'/\bar{z}' \qquad \text{and} \qquad \bar{y} = f\bar{y}'/\bar{z}'. \tag{3.70}$$

Equations (3.65), (3.69), and (3.70) can be joined in:

$$\lambda\mathbf{m} = \mathbf{E}\mathbf{h}(\mathbf{H}^{-1}\mathbf{w}), \tag{3.71}$$

where $\mathbf{E}$ is the 3 × 4 perspective projection matrix expressed in (3.18). However, if the electromagnetic distortion is taken into account, the homogeneous representation of m is from (3.67):

$$\lambda\mathbf{m} = \mathbf{E}\mathbf{h}(\mathbf{f}^{-1}(\mathbf{H}^{-1}\mathbf{w})). \tag{3.72}$$

3.5.2 Correspondence Between Three Views

In three views, we have the projection points m_p, m_q, and m_r at pth, qth, and rth positions, respectively. The correspondence in three views can be established by calculating the epipolar lines of m_p and m_q in third view as shown in Fig. 3.28. If the intersection coincides with m_r, then the three points are corresponding. However, the intersection of epipolar lines in trifocal geometry is not well-defined when the epipolar lines are equal. This situation occurs in two cases: (i) when the 3D point M that has generated the points m_p, m_q, and m_r, lie in the plane defined by the three optical centers, and (ii) when the three optical centers are aligned [29].

In order to avoid these singularities, the relationships in three views are generally described using *trifocal tensors* [2]. Analogous to the two views case explained in Sect. 3.5.1, the three projection equations are:

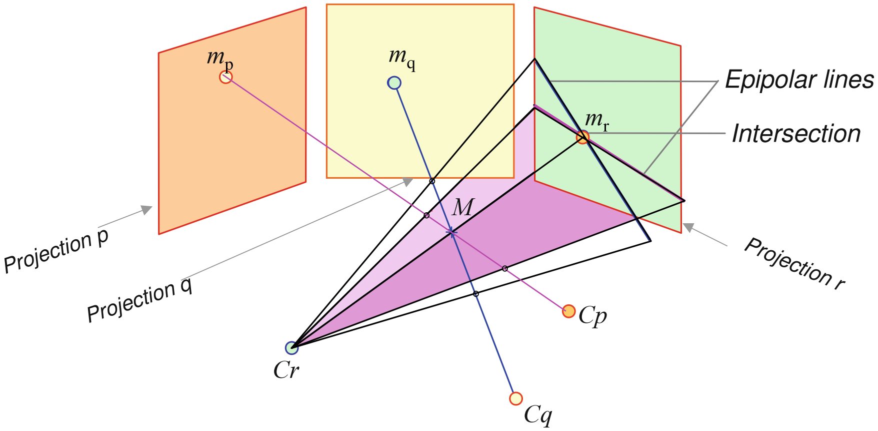


Fig. 3.28 Epipolar geometry in three views

$$\begin{cases} \lambda_p \mathbf{m}_p = \mathbf{P}_p \mathbf{M} := \mathbf{AM} \\ \lambda_q \mathbf{m}_q = \mathbf{P}_q \mathbf{M} := \mathbf{BM} \\ \lambda_r \mathbf{m}_r = \mathbf{P}_r \mathbf{M} := \mathbf{CM} \end{cases} . \tag{3.73}$$

They can be written according to (3.56) by:

$$\underbrace{\begin{bmatrix} \mathbf{a}_1 & x_p & 0 & 0 \\ \mathbf{a}_2 & y_p & 0 & 0 \\ \mathbf{a}_3 & 1 & 0 & 0 \\ \mathbf{b}_1 & 0 & x_q & 0 \\ \mathbf{b}_2 & 0 & y_q & 0 \\ \mathbf{b}_3 & 0 & 1 & 0 \\ \mathbf{c}_1 & 0 & 0 & x_r \\ \mathbf{c}_2 & 0 & 0 & y_r \\ \mathbf{c}_3 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{G}} \underbrace{\begin{bmatrix} \mathbf{M} \\ -\lambda_p \\ -\lambda_q \\ -\lambda_r \end{bmatrix}}_{\mathbf{v}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.74}$$

where $\mathbf{a}_i$, $\mathbf{b}_i$, and $\mathbf{c}_i$ denote the ith row of matrix **A**, **B**, and **C**, respectively. If m_p, m_q, and m_r are corresponding points, then there must be a nontrivial solution for **v**. It follows that the rank of the 9×7 matrix **G** must be at most 6. In other words, all 7×7 submatrices have vanishing determinants. The minors of **G** can be written using Laplace expansions as sums of products of determinants of four rows taken from the first four columns of **G** and products of image coordinates [30]. By expanding the determinants, we can find four linearly independent relationships:

$$\begin{cases} D_1 = (x_r\mathbf{T}^{13} - x_r x_q\mathbf{T}^{33} + x_q\mathbf{T}^{31} - \mathbf{T}^{11})\mathbf{m}_p = 0 \\ D_2 = (y_r\mathbf{T}^{13} - y_r x_q\mathbf{T}^{33} + x_q\mathbf{T}^{32} - \mathbf{T}^{12})\mathbf{m}_p = 0 \\ D_3 = (x_r\mathbf{T}^{23} - x_r y_q\mathbf{T}^{33} + y_q\mathbf{T}^{31} - \mathbf{T}^{21})\mathbf{m}_p = 0 \\ D_4 = (y_r\mathbf{T}^{23} - y_r y_q\mathbf{T}^{33} + y_q\mathbf{T}^{32} - \mathbf{T}^{22})\mathbf{m}_p = 0 \end{cases}, \tag{3.75}$$

where

$$\mathbf{T}^{jk} = [T_1^{jk}\ T_2^{jk}\ T_3^{jk}],$$

and

$$T_i^{jk} = (-1)^{i+1}\begin{vmatrix} \sim\mathbf{a}_i \\ \mathbf{b}_j \\ \mathbf{c}_k \end{vmatrix} \qquad \text{for } i, j, k = 1, 2, 3, \tag{3.76}$$

where $\sim\mathbf{a}_i$ means the matrix $\mathbf{A}$ without row i. The elements T_i^{jk} are called the *trifocal tensors* for the images p, q, and r [25, 27]. For the canonical form $\mathbf{A} = [\mathbf{I} \mid \mathbf{0}]$, the trifocal tensors may be easily obtained by:

$$T_i^{jk} = b_{ji}c_{k4} - b_{j4}c_{ki} \qquad \text{for } i, j, k = 1, 2, 3. \tag{3.77}$$

The equations denoted by (3.75) above, are known as the *trilinearities* [31]. They establish a linear relationship between the coordinates of points m_p, m_q, m_r to find the correspondence. If the three points satisfy the four trilinearities, then they are corresponding points. Equation (3.76) implies that the trifocal tensors do not depend on the points of the images, rather they are computed from the three projection matrices.

The *reprojection* of m_r, i.e., the coordinates $\hat{x}_r$ and $\hat{y}_r$ obtained from the points m_p and m_q, may be simply estimated from the trilinearities (3.75):

$$\lambda\hat{\mathbf{m}}_r = (\mathbf{T}^1 - x_q\mathbf{T}^3)\mathbf{m}_p = (\mathbf{T}^2 - y_q\mathbf{T}^3)\mathbf{m}_p. \tag{3.78}$$

where λ is a scale factor, $\hat{\mathbf{m}}_r = [\hat{x}_r\ \hat{y}_r\ 1]^{\mathsf{T}}$, and $\mathbf{T}^j$ is a 3×3 matrix with the (k, i)-element equal to T_i^{jk}.

In practice, given two corresponding points $\mathbf{m}_p$ and $\mathbf{m}_q$, the third one $\mathbf{m}_r$ can be considered as the corresponding point in third view, if the Euclidean distance between $\mathbf{m}_r$ and its reprojection $\hat{\mathbf{m}}_r$ is smaller than a small number ε:

$$d_3 = \|\hat{\mathbf{m}}_r - \mathbf{m}_r\| < \varepsilon. \tag{3.79}$$

Trifocal geometry is performed for the corresponding points in the X-ray projection coordinate system. In case that the X-ray computer vision system is modeled using a nonlinear geometric model due to an image intensifier, as explained in Sect. 3.5.1, a point w_p, that is found in the pth image, is first transformed into the coordinates m_p of the X-ray projection coordinate system.

Matlab Example 3.8 This example shows how to estimate the coordinates of a corresponding point in view r if we know the trifocal tensors and corresponding points m_p and m_q in views p and q, respectively. Epipolar lines in two views (p and q). This code computes the trifocal tensors from the projection matrices $\mathbf{P}_p$, $\mathbf{P}_q$, and $\mathbf{P}_r$. The reprojection is computed using (3.78).

Listing 3.8 : Reprojection of third point using trifocal tensors.

```
% TrifocalGeometry.m
close all

Data = Xloaddata('B',44,'Pmatrices');    % projection matrices

p = 1; q = 90; r = 170;                  % p, q and r indices
Ip = Xloadimg('B',44,p);
Iq = Xloadimg('B',44,q);
Ir = Xloadimg('B',44,r);
figure(1);imshow(Ip);title('Image p');hold on
figure(2);imshow(Iq);title('Image q');hold on
figure(3);imshow(Ir);title('Image r');hold on

Pp = Data.P(:,:,p);                      % projection matrix of view p
Pq = Data.P(:,:,q);                      % projection matrix of view q
Pr = Data.P(:,:,r);                      % projection matrix of view q

T   = Xtrifocal(Pp,Pq,Pr);               % trifocal tensors
disp('click a point mp in Figure 1...')
figure(1);
mp = [ginput(1) 1]';                     % click
plot(mp(1),mp(2), 'g*')
disp('click a point mp in Figure 2...')
figure(2);
mq = [ginput(1) 1]';                     % click
plot(mq(1),mq(2), 'g*')
mr = Xreproj3(mp,mq,T);                  % reprojection of mr from mp, mq and T
figure(3);
plot(mr(1),mr(2), 'g*')
```

The output of this code is shown in Fig. 3.29. The code uses two functions of 𝕏vis Toolbox: Xtrifocal (see Appendix B) to compute the trifocal tensors and Xreproj3 (see Appendix B) to compute the reprojection of m_r. The example uses images $p = 1$, $q = 90$, and $r = 170$ of series B0044 of 𝔾𝔻𝕏ray. In this set of images there are 178 different views (taken by rotating the test object around a quasi-vertical axis in 2° between consecutive views). The reader who is interested in other views can change the code in order to define other values for p, q, and r. □

3.5.3 Correspondence Between Four Views or More

In the four views case we have the projection points m_p, m_q, m_r, and m_s at pth, qth, rth, and sth positions, respectively. Similar to the previous sections, we can write the four projection equations as a linear equation $\mathbf{Gv} = \mathbf{0}$. Once more, the existence of a nontrivial solution for $\mathbf{v}$ yields in this case to the condition that all 8×8 minors

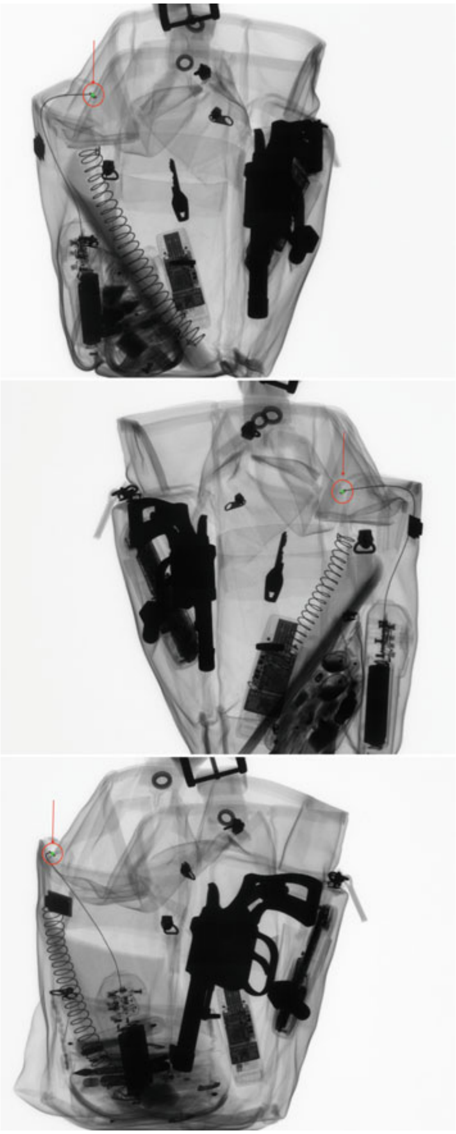


Fig. 3.29 Example of trifocal geometry: views p, q, and r with corresponding points. In this example, the corresponding points in view p and q are known. Corresponding point m_r is estimated from m_p, m_q, and the trifocal tensors $\mathbf{T}$ of these views (→ Example 3.8)

of $\mathbf{G}$ must be zero. Thus, we obtain the well known 81 *quadrifocal tensors* and the corresponding 16 *quadrilinearities* [25, 30].

In practice, the quadrilinearities are not used because they are redundant. Corresponding constraints in four views are obtained from the trilinearities. Thus, the points m_p, m_q, m_r, and m_s are corresponding if m_p, m_q, and m_r are corresponding, and m_q, m_r, and m_s are corresponding as well [18].

For more than four views, a similar approach can be used.

3.6 Three-Dimensional Reconstruction

In X-ray testing, three-dimensional reconstruction is usually related to computed tomography (CT). However, in Computer Vision the attempt is made to estimate only the location (and not the X-ray absorption coefficient) of 3D points in space. In this sense, the reconstruction is based on photogrammetric rather than tomographic methods.

The 3D reconstruction problem can be stated as follows. Given n corresponding points $\mathbf{m}_p$ for $p = 1 \ldots n$ with $n \geq 2$, and the projection matrices of each view $\mathbf{P}_p$, find the "best" 3D point $\mathbf{M}$ which projected by $\mathbf{P}_p$ gives approximately $\mathbf{m}_p$. If we have the projection equation $\lambda_p \mathbf{m}_p = \mathbf{P}_p\mathbf{M}$, we can solve the following system of equation for $\mathbf{M}$:

$$\begin{cases} \lambda_1 \mathbf{m}_1 = \mathbf{P}_1\mathbf{M} \\ \qquad \vdots \\ \lambda_n \mathbf{m}_n = \mathbf{P}_n\mathbf{M} \end{cases} . \tag{3.80}$$

In this section, two approaches that perform the 3D reconstruction, in sense of locating in 3D space, will be described. The 3D reconstruction will be undertaken from corresponding points in the X-ray projection coordinate system. As explained in Sect. 3.5.1, a point w_p, that is found in the pth image, is first transformed into the coordinates m_p of the X-ray projection coordinate system.

3.6.1 Linear 3D Reconstruction from Two Views

Now, we estimate the 3D point M from two corresponding points m_p and m_q using the linear approach introduced by Hartley [27]. Without loss of generality, the method employs the canonical form (see Eq. (3.60)) for the first projection:

$$\lambda_p \mathbf{m}_p = [\mathbf{I} \mid \mathbf{0}]\mathbf{M}'.$$

Thus, the transformed 3D point can be expressed by:

$$\mathbf{M}' = \lambda_p [\mathbf{m}_p^{\mathsf{T}} \; 1/\lambda_p]^{\mathsf{T}}. \tag{3.81}$$

The second projection of this point yields

$$\lambda_q \mathbf{m}_q = \mathbf{B}\mathbf{M}' = \mathbf{B}\lambda_p [\mathbf{m}_p^{\mathsf{T}} \; 1/\lambda_p]^{\mathsf{T}}, \tag{3.82}$$

that is an equation system with three linear equations in the unknowns λ_p and λ_q. If m_p and m_q are corresponding points one may consider only two of these three equations. Taking, for example, the first two equations one may compute λ_p. Substituting the value of λ_p into (3.81) and after some simplifications, we obtain:

$$\mathbf{M} = \mathbf{H}^{-1}\mathbf{M}' = \alpha\mathbf{H}^{-1}\begin{bmatrix} (y_q b_{14} - x_q b_{24})\mathbf{m}_p \\ (x_q \mathbf{b}_2 - y_q \mathbf{b}_1)\mathbf{m}_p \end{bmatrix} \tag{3.83}$$

where α is a scale factor.

3.6.2 3D Reconstruction from Two or More Views

We assume that we have $n \geq 2$ projections at n different positions. In these projections we have found the corresponding points m_p, $p = 1, \ldots, n$, with coordinates (x_p, y_p). To reconstruct the corresponding 3D point M that has produced these projection points, we use a least squares technique [1].

Each projection yields the equation $\lambda_p\mathbf{m}_p = \mathbf{P}_p\mathbf{M}$ as shown in (3.80), with three linear equations in the unknowns (X, Y, Z) and λ_p:

$$\begin{bmatrix} \lambda_1 x_1 \\ \lambda_1 y_1 \\ \lambda_1 \\ \vdots \\ \lambda_n x_n \\ \lambda_n y_n \\ \lambda_n \end{bmatrix} = \begin{bmatrix} s_{11}^1 & s_{12}^1 & s_{13}^1 & s_{14}^1 \\ s_{21}^1 & s_{22}^1 & s_{23}^1 & s_{24}^1 \\ s_{31}^1 & s_{32}^1 & s_{33}^1 & s_{34}^1 \\ \vdots & \vdots & \vdots & \vdots \\ s_{11}^n & s_{12}^n & s_{13}^n & s_{14}^n \\ s_{21}^n & s_{22}^n & s_{23}^n & s_{24}^n \\ s_{31}^n & s_{32}^n & s_{33}^n & s_{34}^n \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{3.84}$$

where s_{ij}^p denotes the (i, j)-element of $\mathbf{P}_p$. With $\lambda_p = s_{31}^p X + s_{32}^p Y + s_{33}^p Z + s_{34}^p$ and after some slight rearranging we obtain:

$$\underbrace{\begin{bmatrix} s_{31}^1 x_1 - s_{11}^1 & s_{32}^1 x_1 - s_{12}^1 & s_{33}^1 x_1 - s_{13}^1 \\ s_{31}^1 y_1 - s_{21}^1 & s_{32}^1 y_1 - s_{22}^1 & s_{33}^1 y_1 - s_{23}^1 \\ \vdots & \vdots & \vdots \\ s_{31}^n x_n - s_{11}^n & s_{32}^n x_n - s_{12}^n & s_{33}^n x_n - s_{13}^n \\ s_{31}^n y_n - s_{21}^n & s_{32}^n y_n - s_{22}^n & s_{33}^n y_n - s_{23}^n \end{bmatrix}}_{\mathbf{Q}} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \underbrace{\begin{bmatrix} s_{14}^1 - s_{34}^1 x_1 \\ s_{24}^1 - s_{34}^1 y_1 \\ \vdots \\ s_{14}^n - s_{34}^n x_n \\ s_{24}^n - s_{34}^n y_n \end{bmatrix}}_{\mathbf{r}}. \tag{3.85}$$

If rank$(\mathbf{Q}) = 3$, the least squares solution for $\hat{\mathbf{M}} = [\hat{X}\ \hat{Y}\ \hat{Z}]^\mathsf{T}$ is then given by:

$$\hat{\mathbf{M}} = [\mathbf{Q}^\mathsf{T}\mathbf{Q}]^{-1}\mathbf{Q}^\mathsf{T}\mathbf{r}. \tag{3.86}$$

Matlab Example 3.9 In this example, we estimate the length of an object in millimeters. There are three views p, q, and r. Two points of the object are given by the user in each view (by mouse clicking). The code estimates the two 3D points and compute the 3D distance between them. Since the calibration of this X-ray computer vision system was implemented using a calibration object with dimensions measured in millimeters, it is clear that the 3D reconstructed points are given in millimeters as well.

Listing 3.9 : 3D Reconstruction.

```
% Reconstruction3D.m
close all

Data = Xloaddata('B',44,'Pmatrices'); % projection matrices

p = 1; q = 40; r = 90;              % p, q and r indices
Ip = Xloadimg('B',44,p);
Iq = Xloadimg('B',44,q);
Ir = Xloadimg('B',44,r);
figure(1);imshow(Ip);title('Image p');hold on
figure(2);imshow(Iq);title('Image q');hold on
figure(3);imshow(Ir);title('Image r');hold on

P1 = Data.P(:,:,p);                 % projection matrix of view p
P2 = Data.P(:,:,q);                 % projection matrix of view q
P3 = Data.P(:,:,r);                 % projection matrix of view q
P = [P1;P2;P3];                     % all projection matrices

figure(1);disp('click first and second points in Figure 1...')
m1 = [ginput(2) ones(2,1)]';        % click
plot(m1(1,:),m1(2,:), 'r*')
plot(m1(1,:),m1(2,:), 'g')

figure(2);disp('click first and second points in Figure 2...')
m2 = [ginput(2) ones(2,1)]';        % click
plot(m2(1,:),m2(2,:), 'r*')
plot(m2(1,:),m2(2,:), 'g')

figure(3);disp('click first and second points in Figure 3...')
m3 = [ginput(2) ones(2,1)]';        % click
plot(m3(1,:),m3(2,:), 'r*')
plot(m3(1,:),m3(2,:), 'g')

mm_1 = [m1(:,1) m2(:,1) m3(:,1)];   % first 2D point in each view
mm_2 = [m1(:,2) m2(:,2) m3(:,2)];   % second 2D point in each view
M1 = Bmv_reco3dn(mm_1,P);           % 3D reconstruction of first point
M2 = Bmv_reco3dn(mm_2,P);           % 3D reconstruction of second point

Md = M1(1:3)-M2(1:3);               % 3D vector from 1st to 2nd point
dist = norm(Md)                     % length of 3D vector in mm
```

The output of this code is shown in Fig. 3.30. The code uses the function of 𝕏vis Toolbox: Xreco3dn (see Appendix B) to compute the 3D reconstruction using (3.86). The estimated length in this example was `dist = 46.2881` mm. The reader who is interested in 3D reconstruction using (3.83) for two views can use command Xreco3d2 (see Appendix B) from 𝕏vis Toolbox. □

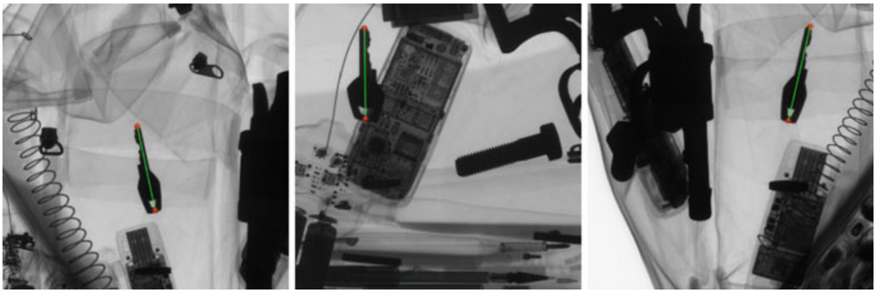


Fig. 3.30 Example of 3D reconstruction using three views. There are two corresponding points in each view (in this example the coordinates of the top and the bottom of the key were manually given). Two 3D points were reconstructed using (3.86) and the distance between these two 3D points was computed. In this example, the estimation of the length of the key was 46.29 mm (→ Example 3.9)

3.7 Summary

In this chapter, we presented several methods that can be used when dealing with geometric problems in X-ray testing. We gave a theoretical background of geometry using homogenous coordinates. Thus, the projective transformations can be easily established. Linear and nonlinear models for X-ray computer vision systems were outlined, in order to relate the 3D coordinates of an object to the 2D coordinates of the digital X-ray image pixel. In addition, calibration approaches that can be used to estimate the parameters of these models were studied. Finally, multiple view geometry was outlined. We presented geometric and algebraic constraints between two, three, and more X-ray images obtained as different projections of the object, and we explained the problem of the 3D reconstruction.

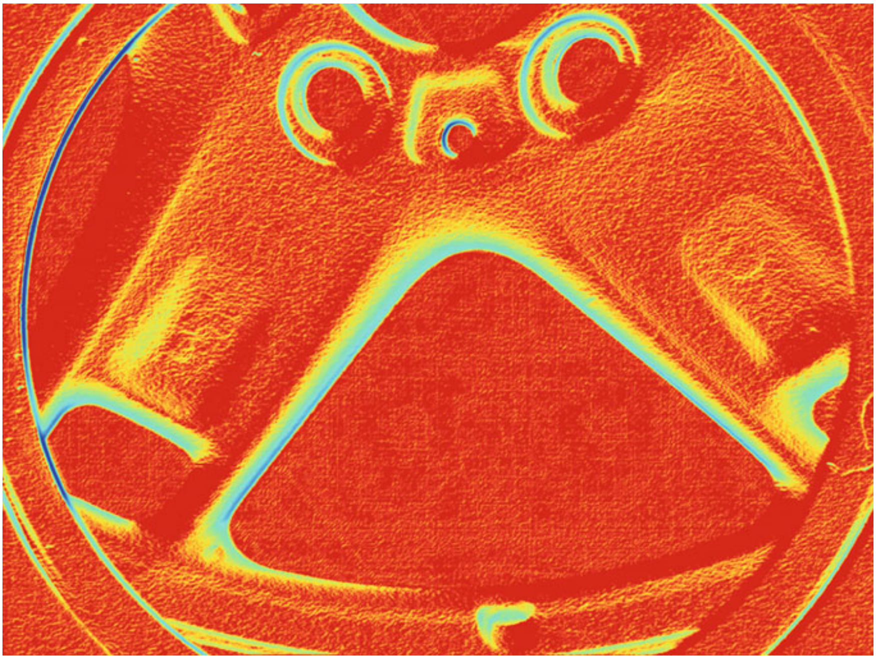
References

1. Faugeras, O.: Three-Dimensional Computer Vision: A Geometric Viewpoint. The MIT Press, Cambridge (1993)
2. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2003)
3. Tsai, R.: A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. IEEE Trans. Robot. Autom. **RA-3**(4), 323–344 (1987)
4. Wei, G.Q., Ma, S.: Implicit and explicit camera calibration: theory and experiments. IEEE Trans. Pattern Anal Mach. Intell. **16**(5), 469–480 (1994)
5. Weng, J., Cohen, P., Herniou, M.: Camera calibration with distortion models and accuracy evaluation. IEEE Trans. Pattern Anal. Mach. Intell. **4**(10), 965–980 (1992)
6. Faugeras, O., Toscani, G.: The calibration problem for stereo. In: Proceedings IEEE Computer Vision and Pattern Recognition, pp. 15–20 (1986)

7. Heikkilä, J.: Geometric camera calibration using circular control points. IEEE Trans. Pattern Anal. Mach. Intell. **22**(10), 1066–1077 (2000)
8. Swaminathan, R., Nayar, S.: Nonmetric calibration of wide-angle lenses and polycameras. IEEE Trans. Pattern Anal. Mach. Intell **22**(10), 1172–1178 (2000)
9. Jaeger, T.: Methods for rectification of geometric distortion in radioscopic images. Master theses, Institute for Measurement and Automation, Faculty of Electrical Engineering, Technical University of Berlin (1990) (in German)
10. Brack, C., Götte, H., Gossé, F., Moctezuma, J., Roth, M., Schweikard, A.: Towards accurate X-ray-camera calibration in computer-assisted robotic surgery. In: Proceedings of the International Symposium Computer Assisted Radiology (CAR), pp. 721–728. Paris (1996)
11. Mery, D., Filbert, D.: Automated flaw detection in aluminum castings based on the tracking of potential defects in a radioscopic image sequence. IEEE Trans. Robot. Autom. **18**(6), 890–901 (2002)
12. Mery, D., Filbert, D.: The epipolar geometry in the radioscopy: theory and application. at - Automatisierungstechnik **48**(12), 588–596 (2000) (in German)
13. Felix, R., Ramm, B.: Das Röntgenbild, 3rd edn. Georg Thieme Verlag, Stuttgart (1988)
14. Mery, D.: Explicit geometric model of a radioscopic imaging system. NDT & E Int. **36**(8), 587–599 (2003)
15. Halmshaw, R.: Non-Destructive-Testing, 2nd edn. Edward Arnold, London (1991)
16. Franzel, T., Schmidt, U., Roth, S.: Object detection in multi-view x-ray images. Pattern Recognit. 144–154 (2012)
17. Grignon, B., Mainard, L., Delion, M., Hodez, C., Oldrini, G.: Recent advances in medical imaging: anatomical and clinical applications. Surg. Radiol. Anat. **34**(8), 675–686 (2012)
18. Mery, D.: Automated Flaw Detection in Castings from Digital Radioscopic Image Sequences. Verlag Dr. Köster, Berlin (2001). Ph.D. thesis in German
19. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach Intell. **22**(11), 1330–1334 (2000)
20. Luong, Q.T., Faugeras, O.: Self calibration of a moving camera from point correspondences and fundamental matrices. Int. J. Comput. Vis. **22**(3), 261–289 (1997)
21. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **22**(11), 1330–1334 (2000)
22. MathWorks: Computer Vision Toolbox for Use with MATLAB: User's Guide. The MathWorks Inc. (2014)
23. MathWorks: Optimization Toolbox for Use with MATLAB: User's Guide. The MathWorks Inc. (2014)
24. Faugeras, O., Luong, Q.T., Papadopoulo, T.: The Geometry of Multiple Images: The Laws that Govern the Formation of Multiple Images of a Scene and Some of Their Applications. The MIT Press, Cambridge (2001)
25. Hartley, R.: Multilinear relationships between coordinates of corresponding image points and lines. In: Proceedings of the International Workshop on Computer Vision and Applied Geometry. International Sophus Lie Center, Nordfjordeid, Norway (1995)
26. Heyden, A.: A common framework for multiple view tensors. In: 5th European Conference on Computer Vision (ECCV-98), pp. 3–19 (1998)
27. Hartley, R.: A linear method for reconstruction from lines and points. In: 5th International Conference on Computer Vision (ICCV-95), pp. 882–887. Cambridge (1995)
28. Hartley, R.: Lines and points in three views and the trifocal tensor. Int. J. Comput. Vis. **22**(2), 125–150 (1997)
29. Faugeras, O., Papadopulo, T.: A nonlinear method for estimating the projective geometry of 3 views. In: 6th International Conference on Computer Vision (ICCV-98), pp. 477–484. Bombay, India (1998)
30. Heyden, A.: Multiple view geometry using multifocal tensors. In: DSAGM. Köpenhamn (1999)
31. Shashua, A., Werman, M.: Trilinearity of three perspective views and its associated tensor. In: 5th International Conference on Computer Vision (ICCV-95). Boston (1995)

Chapter 4
X-ray Image Processing

Abstract In this chapter, we cover the main techniques of image processing used in X-ray testing. They are: (i) image processing to enhance details, (ii) image filtering to remove noise or detect high frequency details, (iii) edge detection to identify the boundaries of the objects, (iv) image segmentation to isolate the regions of interest and (v) to remove the blurriness of the X-ray image. The chapter provides an overview and presents several methodologies with examples using real and simulated X-ray images.



Cover image: *Gradient of an X-ray image of a wheel (from X-ray image* `C0001_0001` *colored with 'jet' colormap).*

D. Mery, *Computer Vision for X-Ray Testing*, DOI 10.1007/978-3-319-20747-6_4

4.1 Introduction

Image processing manipulates a digital image in order to obtain a *new* digital image, i.e., in this process the input is an image and the output is another image. A typical example is *segmentation* as shown in Fig. 4.1, where the input is a grayscale image that contains a clip and the output is a binary image where the pixels that belong to the clip are detected. In our book, we distinguish image processing from image analysis, in which the output is rather an interpretation, a recognition or a measurement of the input image. We will perform image analysis further on, when we learn pattern recognition techniques such as feature extraction (see Chap. 5) and classification (see Chap. 6).

In this chapter, we cover the following image processing techniques that are used in X-ray testing.

- Image preprocessing: The quality of the X-ray image is improved in order to enhance its details.
- Image Filtering: Mainly used to remove noise and detect high frequency details of the X-ray image.
- Edge detection: The details of the images can be highlighted by detecting the boundaries of the objects of the X-ray image.
- Image segmentation: Regions of interest of the X-ray image are identified and isolated from their surroundings.
- Image restoration: This involves recovering details in blurred images.

In this chapter, we provide an overview of these five techniques. Methodologies and principles will also be outlined, and some application examples followed by limitations to the applicability of the used methodologies will be presented.

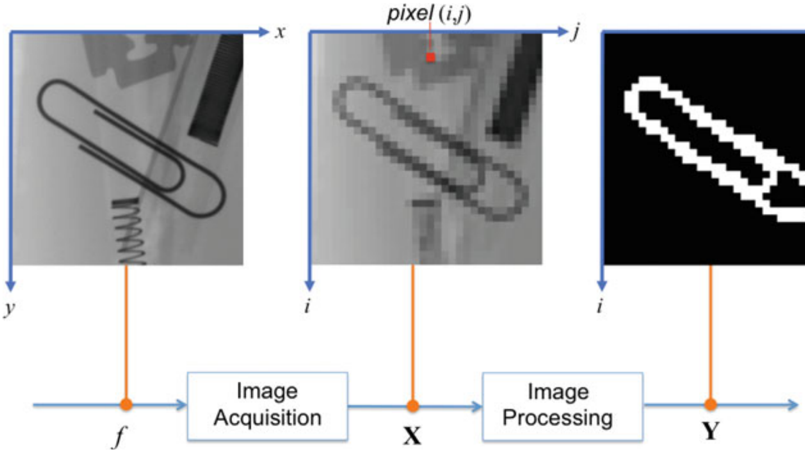


Fig. 4.1 Image processing: input is digital image **X**, output is digital image **Y**

In image processing methodology we have a continuous image f defined in a coordinate system (x, y). Image f is digitalized. The obtained image is a digital image which is stored in matrix $\mathbf{X}$ of size $M \times N$ pixels. The gray value of pixel (i, j) of image $\mathbf{X}$ is $X(i, j)$. Image $\mathbf{X}$ is processed digitally. The output image of this process is image $\mathbf{Y}$, usually a matrix of the same size of $\mathbf{X}$. In this example, the output is a binary image, that means $Y(i, j)$ is '1' (white) and '0' black. Image $\mathbf{Y}$ corresponds to the segmentation of a clip (that is the *object of interest* in this example).

4.2 Image Preprocessing

The X-ray image taken must be preprocessed to improve the quality of the image before it is analyzed. In this section, we will discuss preprocessing techniques that can remove noise, enhance contrast, correct the shading effect and restore blur deformation in X-ray images.

4.2.1 Noise Removal

Noise in an X-ray image can prove a significant source of image degradation and must be taken into account during image processing and analysis. In an X-ray imaging system, *photon noise* occurs given the quantum nature of X-rays. If we have a system that receives μ photons per pixel in a time ΔT on average, the number of photons striking any particular pixel in any time ΔT will be random. At low levels, however, the noise follows a Poisson law, characterized by the probability:

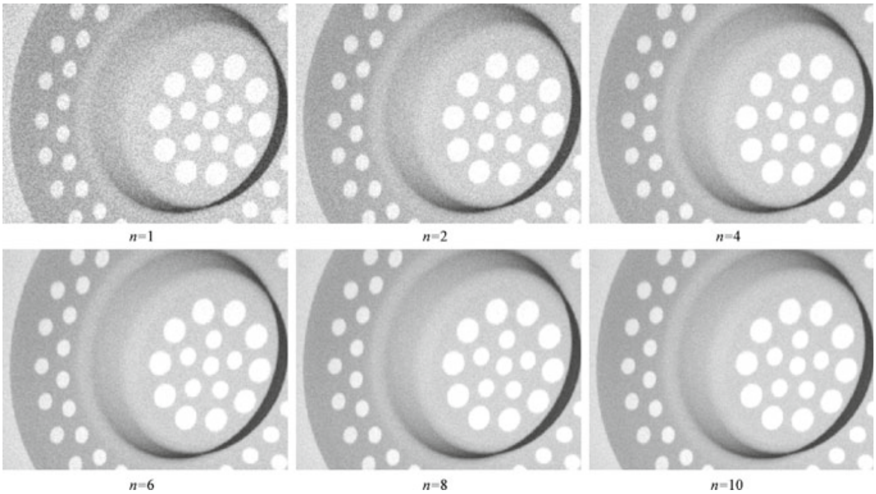
$$p(x|\mu) = \frac{e^{-\mu}}{\mu^x x!} \tag{4.1}$$

to obtain a value x of photons given its average μ photons in a time ΔT. The standard deviation of this distribution is equal to the square root of the mean.[1] This means that the photon noise amplitude is signal-dependent.

Integration (or averaging) is used to remove X-ray image noise. This technique requires n stationary X-ray images. It computes the filtered image as follows:

$$Y(i, j) = \frac{1}{n}\sum_{k=1}^{n} X_k(i, j) \tag{4.2}$$

[1] At high levels, the Poisson distribution approaches the Gaussian with a standard deviation equal to the square root of the mean: $\sigma = \sqrt{\mu}$.



$n=1$ $n=2$ $n=4$

$n=6$ $n=8$ $n=10$

Fig. 4.2 Noise removal after an averaging of n frames. The noise is reduced by factor $\sqrt{n}$

where $X_k(i, j)$ is pixel (i, j) of kth stationary image, and $Y(i, j)$ is the corresponding pixel of the filtered image.

In this technique, the X-ray image noise is modeled using two components: the image of interest (that is constant throughout the n images) and the noise component (that varies from one image to the next). If the noise component has zero mean, by averaging the n images the stationary component is unchanged, while the noise pattern decreases by increasing n. Integrating n stationary X-ray images improves the signal-to-noise ratio by a factor of $\sqrt{n}$ [1, 2].

The effect of image integration is illustrated in Fig. 4.2 that uses n stationary images of an aluminum casting and shows the improvement in the quality of the X-ray image. The larger the number of stationary images n, the better the improvement. Normally, between 10 and 16 stationary images are taken ($10 \leq n \leq 16$).

Matlab Example 4.1 In this example, we have 20 noisy X-ray images obtained from a very thin wood piece. The following Matlab code uses averaging to effectively remove X-ray image noise (4.2):

Listing 4.1 : Noise removal by averaging.

```
% AritmeticAverage.m

close all
S  = double(Xloadimg('N',4,1));                        % S = image 1
n = 20;
for k=2:n
    Xk = double(Xloadimg('N',4,k));                    % image k
    imshow(Xk,[]);
    title(['image with noise ' num2str(k)])
    pause(0.25);
    S = S + Xk;                                        % S = S + image k
end
```

```
Y = S/n;                                              % average

figure(2);
imshow(Y,[]);title('filtered image')

figure(3)
plot([Xk(100,:)' Y(100,:)'])                          % profile of row 100
legend({'original','filtered'})
```

The output of this code is shown in Fig. 4.3. The reduction of noise is not perfect but very satisfactory. The reader can test this approach on series `C0034` and `C0041` of 𝔾𝔻𝕏ray, in which 37 noisy X-ray images of an aluminum wheel with no motion are taken. □

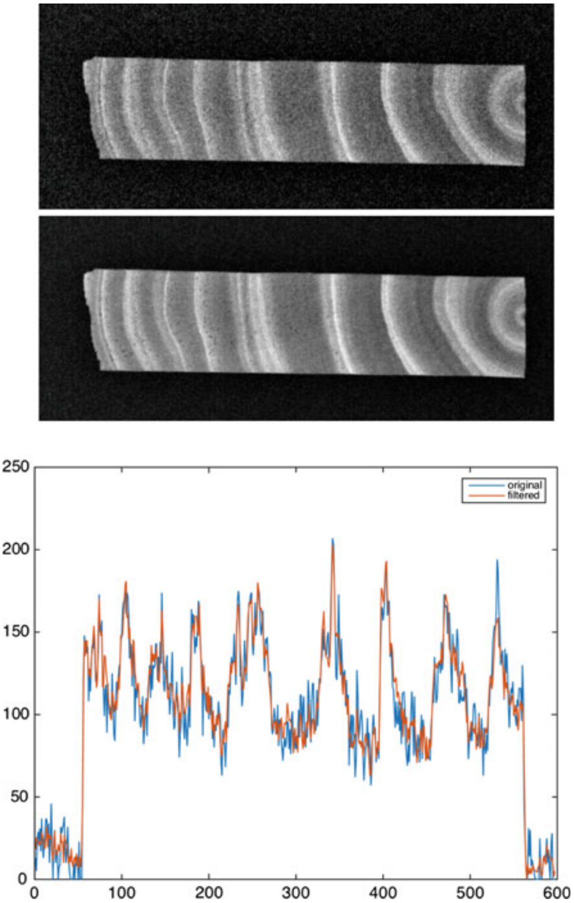


Fig. 4.3 Noise removal of an X-ray image of a wood piece after an averaging of 20 frames. *Top* one of the 20 images. *Middle* filtered image. *Bottom* row 100 of each image (→ Example 4.1)

4.2.2 Contrast Enhancement

The gray values in some X-ray images lie in a relatively narrow range of the gray scale. In this case, enhancing the contrast will amplify the differences in the gray values of the image.

We compute the gray value histogram to investigate how an X-ray image uses the grayscale. The function summarizes the gray value information of an X-ray image. The histogram is a function $h(x)$ that denotes the number of pixels in the X-ray image that have a gray value equal to x. Figure 4.4 shows how each histogram represents the distribution of gray values in the X-ray images.

A transformation can be applied to modify the distribution of gray value in an X-ray image. Simple contrast enhancement can be achieved if we use a linear transformation which sets the minimal and maximal gray values of the X-ray image to the minimal and maximal gray value of the grayscale respectively. Thus, the histogram is expanded to occupy the full range of the grayscale. Mathematically, for a scale between 0 and 255, this transformation is expressed as:

$$Y(i,j) = 255 \cdot \frac{X(i,j) - x_{\min}}{x_{\max} - x_{\min}} \tag{4.3}$$

where $x_{\min}$ and $x_{\max}$ denote the minimal and maximal gray value of the input X-ray image. The output image is stored in matrix $\mathbf{Y}$. This simple function is implemented in command Xlinimg (see Appendix B) from $\mathbb{X}$vis Toolbox. Figure 4.4b shows the result of the transformation applied to the X-ray image in Fig. 4.4a. We observe in the histogram of the enhanced X-ray image, how the gray values expand from '0' to '255'. The mapping is linear, and means that a gray value equal to $\frac{1}{2}(x_{\max} - x_{\min})$ will be mapped to $255/2$. This linear transformation is illustrated in Fig. 4.5a, where the abscissa is the input gray value and the ordinate is the output gray value.

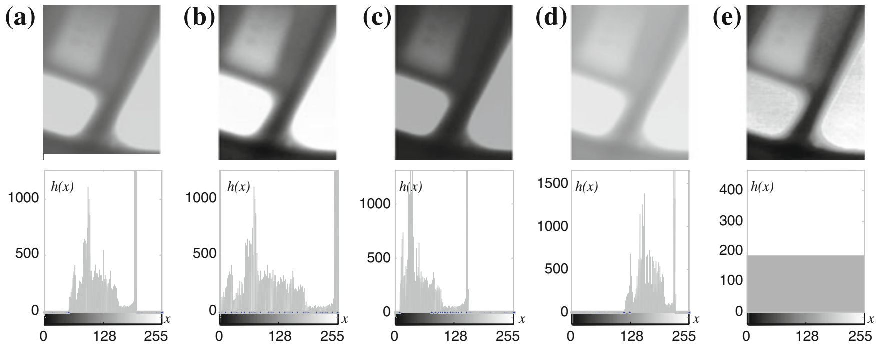


Fig. 4.4 Contrast enhancement: **a** original image, **b** linear transformation ($\gamma = 1$), **c** nonlinear transformation ($\gamma = 2$), **d** nonlinear transformation ($\gamma = 1/2$), **e** gray values uniformly distributed

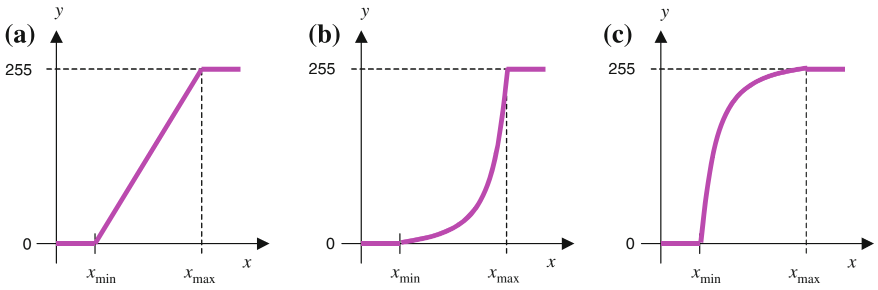


Fig. 4.5 Plots showing different transformations of the gray values: **a** linear transformation ($\gamma = 1$), **b** nonlinear transformation with $\gamma > 1$, **c** nonlinear transformation with $\gamma < 1$

In a similar fashion, gray input image values can be mapped using a nonlinear transformation $y = f(x)$, as illustrated in Fig. 4.5b, c, the results of which are shown in Fig. 4.4c, d, respectively. Here, x and y are the gray values of the input and output images respectively. The nonlinear transformation is usually performed with a γ *correction* [3]. In these examples, if $\gamma > 1$ the mapping is weighted toward darker output values, and if $\gamma < 1$ the mapping is weighted toward brighter output values. Gamma transformation can be expressed as:

$$Y(i,j) = \begin{cases} 0 & \text{for} \quad X(i,j) < x_{\text{min}} \\ 255 \cdot \left[\frac{X(i,j)-x_{\text{min}}}{x_{\text{max}}-x_{\text{min}}}\right]^{\gamma} & \text{for} \quad x_{\text{min}} \le X(i,j) \le x_{\text{max}} \\ 255 & \text{for} \quad X(i,j) > x_{\text{max}} \end{cases} \tag{4.4}$$

Finally, we present a contrast enhancement equalizing the histogram. Here, we can alter the gray value distribution in order to obtain a desired histogram. A typical equalization corresponds to the uniform histogram as shown in Fig. 4.4d. We see that the number of pixels in the X-ray image for each gray value is constant.

Matlab Example 4.2 In this example, we have an X-ray image of a baggage with very dark zones. The user defines a zone to be enhanced by clicking two opposite corners of a rectangle. The code forces the histogram of this zone to be uniform:

Listing 4.2 : Contrast enhancement of a selected area.

```
% ContrastEnhancement.m
close all
X = Xloadimg('B',44,130);                           % input image
figure(1)
imshow(X,[]); title('original image')
hold on
disp('Select top corner of rectangle to be enhanced...')
p = round(ginput(1));
i1 = p(1,2); j1 = p(1,1);                            % coordinates of first corner
plot(j1,i1,'r+')
disp('Select bottom corner of rectangle to be enhanced...')
p = round(ginput(1));
i2 = p(1,2); j2 = p(1,1);                            % coordinates of second corner
```

```
plot([j1 j2 j2 j1 j1],[i1 i1 i2 i2 i1],'r')          % selected area
Xbox = X(i1:i2,j1:j2);                               % subimage
Ybox = Xforceuni(Xbox);                              % equalization of subimage
Y = X;
Y(i1:i2,j1:j2) = Ybox;                               % replacement
figure(2)
imshow(Y,[]); title('enhanced image')
```

The output of this code is shown in Fig. 4.6. For the equalization, the code uses function Xforceuni (see Appendix B) from 𝕏vis Toolbox. □

4.2.3 *Shading Correction*

A decrease in the angular intensity in the projection of the X-rays causes low spatial frequency variations in X-ray images [2, 4]. An example is illustrated in Fig. 4.7a, which shows an X-ray image of an aluminum plate with holes in it. Since the plate is of a constant thickness, we would expect to see a constant gray value for the aluminum part and another constant gray value for the holes. In fact, the X-ray image is darker at the corners. This deficiency can be overcome by using a linear shading correction.

In this technique, we take two images as shown in Fig. 4.8. The first one, $\mathbf{r}_1$, of a thin plate, and the second one, $\mathbf{r}_2$, of a thick plate. We define i_1 and i_2 as the ideal gray values for the first and second image, respectively. From $\mathbf{r}_1$, $\mathbf{r}_2$, i_1 and i_2, offset and gain, correction matrices **a**, and **b** are calculated assuming a linear transformation between the original X-ray image **x** and corrected X-ray image **y**:

$$Y(i, j) = a(i, j) \cdot X(i, j) + b(i, j), \tag{4.5}$$

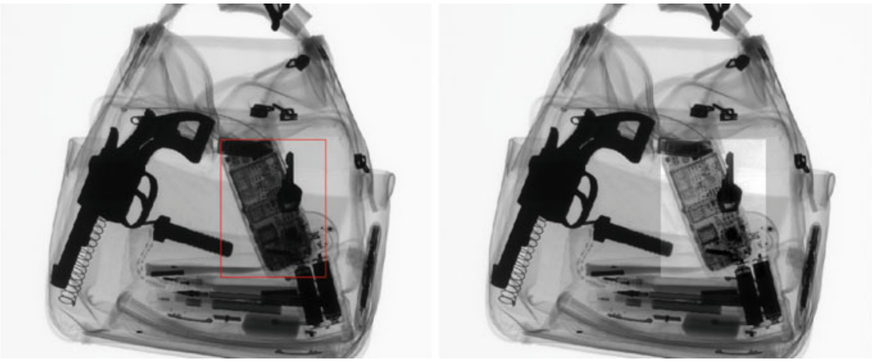


Fig. 4.6 Contrast enhancement by uniforming a histogram of the selected area (→ Example 4.2)

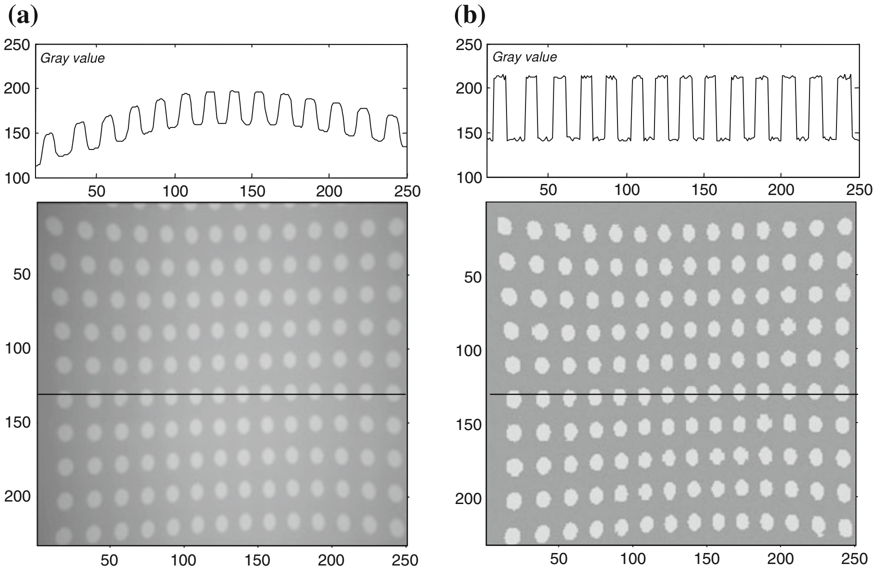


Fig. 4.7 Shading correction: **a** original image, **b** image after shading correction. The corresponding gray values profiles of row number 130 are shown above the images

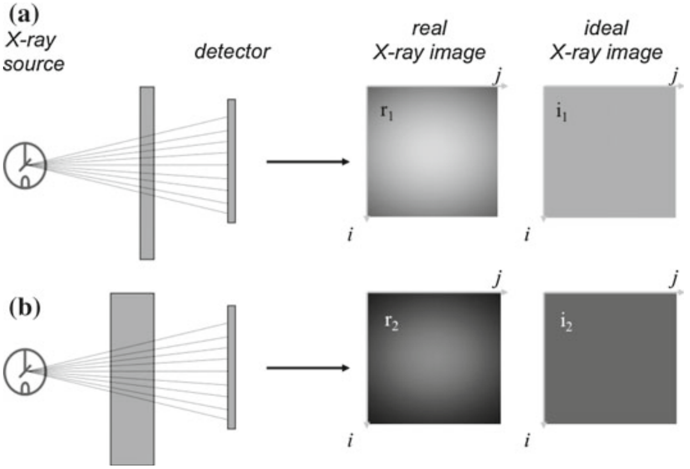


Fig. 4.8 Shading correction: **a** X-ray image for a thin plate, **b** X-ray image for a thick plate. Ideal X-ray images have a constant gray value

where the offset and gain matrices are computed as follows:

$$a(i,j) = \frac{i_2 - i_1}{r_2(i,j) - r_1(i,j)} \qquad b(i,j) = i_1 - r_1(i,j) \cdot a(i,j). \tag{4.6}$$

An example of this technique is illustrated in Fig. 4.7b. In this case, we obtain only two gray values (with noise) one for the aluminum part and another for the holes of the plate.

Matlab Example 4.3 In this example, we simulate images **X** (a plate with a square cavity). In addition, we simulate X-ray images $\mathbf{r}_1$ (a thin plate) and $\mathbf{r}_2$ (a thick plate) as illustrated in Fig. 4.8. The following Matlab code shows how the shading effect of **X** can be corrected:

Listing 4.3 : Shading correction.

```
% ShadingCorrection.m

R1 = fspecial('Gaussian',256,80); R1=R1/max(R1(:))*0.8; % X–ray image of a thin plate
i1 = 0.8;                                               % ideal gray value for R1
R2 = fspecial('Gaussian',256,60); R2=R2/max(R2(:))*0.6; % X–ray image of a thick plate
i2 = 0.4;                                               % ideal gray value for R2

X  = fspecial('Gaussian',256,70); X=X/max(X(:))*0.7;    % simulation of a X–ray image
X(30:80,30:80)  = X(30:80,30:80)*1.5;                   % with a square cacity

figure(1)
imshow(X,[])                                            % input image
figure(2)
Y = Xshading(X,R1,R2,i1,i2);                            % output image
imshow(Y,[])
```

The output of this code is shown in Fig. 4.9. The correction is evident: the appearance of the background is homogenous whereas the square is more distinguishable.

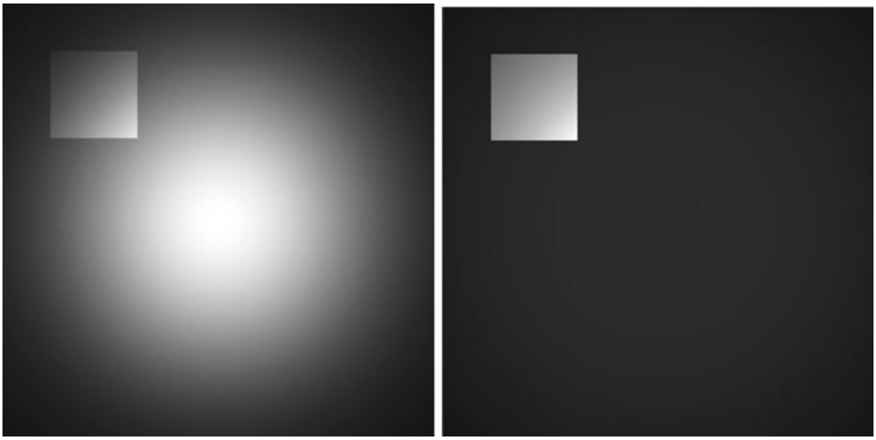


Fig. 4.9 Simulation of shading correction: *Left* X-ray image for a plate with a square cavity (**X**), *Right* corrected image (**Y**) (→ Example 4.3)

In this code, we use function Xshading (see Appendix B) of $\mathbb{X}$VIS Toolbox. This function computes shading correction as defined in (4.5). □

4.3 Image Filtering

2D image filtering is performed in digital image processing using a small neighborhood of a pixel $X(i, j)$ in an input image to produce a new gray value $Y(i, j)$ in the output image, as shown in Fig. 4.10. A *filter mask* defines the input pixels to be processed by an *operator* f. The resulting value is the output pixel. The output for the entire image is obtained by shifting the mask over the input image. Mathematically, the image filtering is expressed as:

$$Y(i, j) = f[\underbrace{X(i-p, j-p), \ldots, X(i, j), \ldots, X(i+p, j+p)}_{\text{input pixels}}], \tag{4.7}$$

for $i = p + 1 \ldots M - p$ and $j = p + 1 \ldots N - p$, where M and N are the number of rows and columns of the input and output images. The size of the filter mask is, in this case, $(2p + 1) \times (2p + 1)$. The operator f can be linear or nonlinear. In this section, the most important linear and nonlinear filters for X-ray testing are outlined.

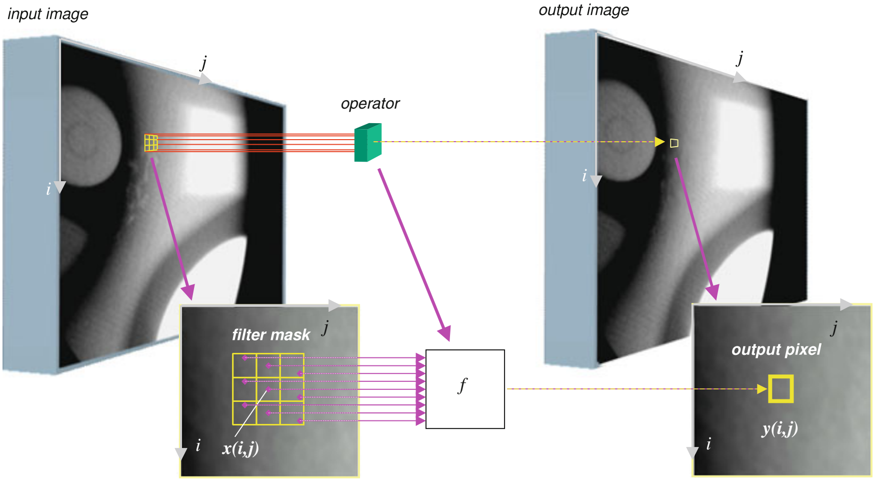


Fig. 4.10 Image filtering

4.3.1 Linear Filtering

The operator f is linear, if the resulting value $Y(i, j)$ is calculated as a linear combination of the input pixels:

$$Y(i, j) = \sum_{m=-p}^{p} \sum_{n=-p}^{p} h(m, n) \cdot X(i - m, j - n), \tag{4.8}$$

where **h** is called the *convolution mask*. The elements of **h** weight the input pixels. The convolution of an image **X** with a mask **h** can be written as:

$$\mathbf{Y} = \mathbf{X} * \mathbf{h}, \tag{4.9}$$

Averaging is a simple example of linear filtering. For a 3 × 3 neighborhood, the convolution mask is

$$\mathbf{h} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Gaussian mask can be used as well

$$h(m, n) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{m^2+n^2}{2\sigma^2}} \tag{4.10}$$

scale factor $1/(2\pi\sigma^2)$ ensures $\sum_{m,n} h(m, n) = 1$ over all elements of **h**. Average and Gaussian filtering are implemented respectively as functions Ximaverage and Ximgaussian (see Appendix B) in $\mathbb{X}$vis Toolbox.

A common application of filtering in X-ray testing is defect detection, e.g., in castings and welds. Filtering out defects detected in an X-ray image will provide a reference *defect-free* image. The defects are detected by finding deviations in the original image from the reference image. The problem is how one can generate a defect-free image from the original X-ray image. Assuming that the defects will be smaller than the regular structure of the test piece, one can use a low-pass filter that does not consider the high frequency components of the image. However, if a linear filter is used for this task, the edges of the regular structure of the specimen are not necessarily preserved and many false alarms are raised at the edges of regular structures. Consequently, a nonlinear filter is used.

4.3.2 Nonlinear Filtering

In order to avoid the mentioned problems of linear filters, nonlinear filters are used. Defect discrimination can be performed with a *median filter*. The median filter is a ranking operator (and thus nonlinear) where the output value is the middle value of the input values ordered in a rising sequence [5]. For an even number of input numbers the median value is the arithmetic mean of the two middle values.

The application of a median filter is useful for generating the reference image because it smoothes noise yet preserves sharp edges, whereas other linear low-pass filters blur such edges (see a comparison with linear filters in Fig. 4.11). Hence, it follows that small defects can be suppressed while the regular structures are preserved. Figure 4.12 shows this phenomenon for a 1D example. The input signal x is

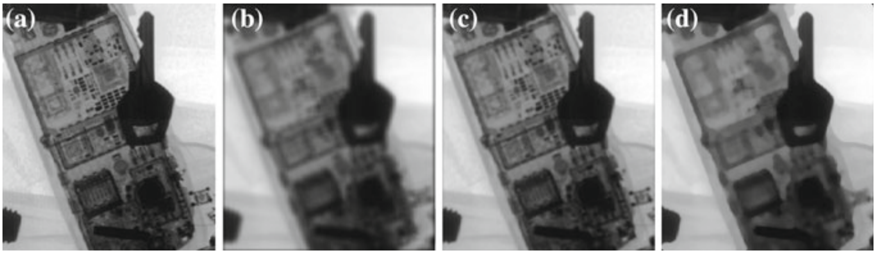


Fig. 4.11 Example of filtering of **a** an X-ray image of 600 × 700 pixels using **b** arithmetic, **c** Gaussian, and **d** median filters with a mask of 19 × 19 pixels. The filtered images where obtained using commands Ximaverage, Ximgaussian and Ximmedian (see Appendix B) of 𝕏VIS Toolbox

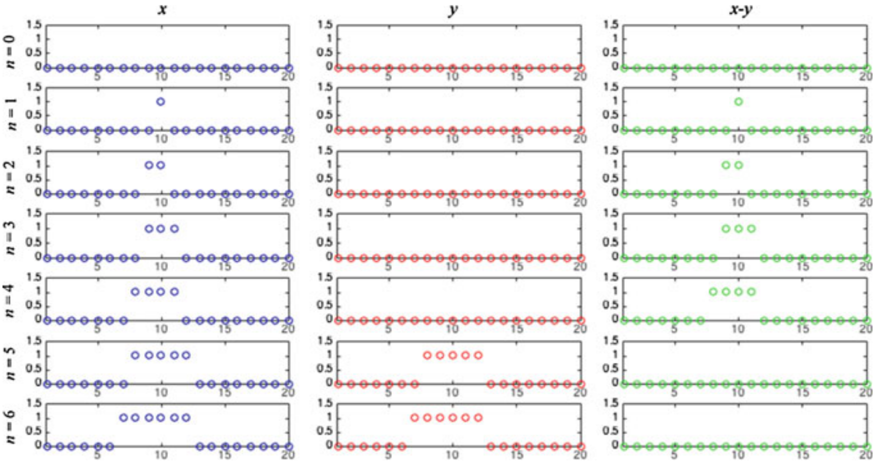


Fig. 4.12 Median filter application on a 1D signal x. The filtered signal is y (the size of the median mask is 9). Structures of length n less than 9/2 are eliminated in y. This filter can be used to detect small structures ($n \leq 4$)

filtered using a median filter with nine input elements, and the resulting signal is y. We can see that structures of length n greater than four cannot be eliminated. The third column shows the detection $x - y$. Large structures of $n \geq 5$ are not detected, as presented in the last two cases.

If the background captured by the median filter is constant, foreground structures could be suppressed if the number of values belonging to the structure is less than one-half of the input value to the filter. This characteristic is utilized to suppress the defect structures and to preserve the design features of the test piece in the image.

An example for the application of a median filter on 2D signals (images) is shown in Fig. 4.13 and includes different structures and mask sizes compared to the effects of two linear low-pass filters. One can appreciate that only the median filter manages to suppress the relatively small structures completely, whereas the large patterns retain their gray values and sharp edges.

The goal of the background image function, therefore, is to create a defect-free image from the test image. A real example is shown in Fig. 4.14. In this example, from an original X-ray image **X** we generate a filtered image **Y** and a difference image $|\mathbf{X} - \mathbf{Y}|$. By setting a threshold, we obtain a binary image whose pixels are '1' (or white), where the gray values in the difference image are greater than the selected threshold. Finally, we eliminate very small regions. The remaining pixels correspond to the detected flaws.

Matlab Example 4.4 In this example, we detect small defects of an aluminum wheel. First, a reference defect-free image is estimated from original image itself using median filtering. Second, the difference between original and reference image is computed. Finally, defects are detected when the difference in gray values is high enough and the size of the detected region is large enough:

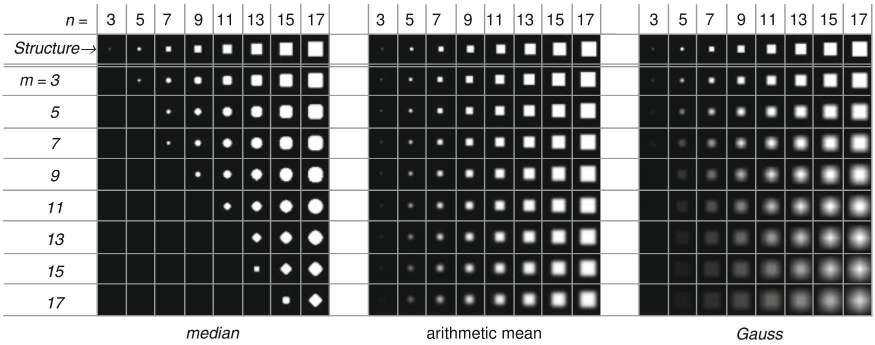


Fig. 4.13 Median filter application on an $n \times n$ structure using an $m \times m$ quadratic mask compared to average and Gauss low-pass filter application

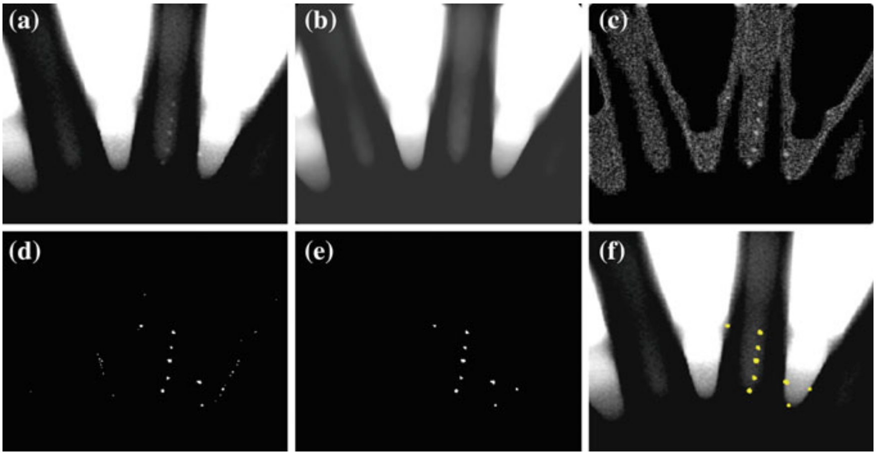


Fig. 4.14 Defect detection using median filtering: **a** original X-ray image of an aluminum wheel with small defects, **b** filtered X-ray image, **c** difference image, **d** binary image using a threshold, **e** elimination of small regions, **f** detection superimposed onto original image (→ Example 4.4)

Listing 4.4 : Defect detection using median filtering

```
% MedianDetection.m

close all;
X   = Xloadimg('C',21,29);                         % original image
X   = Ximgaussian(X,5);                            % low pass filtering
figure(1);
imshow(X,[]); title('Original image with defects');

figure(2)
Y0 = Ximmedian(X,23);
imshow(Y0,[]);title('Median filter');

figure(3);
Y1 = abs(double(X)-double(Y0));
imshow(log(Y1+1),[]); title('Difference image');

figure(4)
Y2 = Y1>18;
imshow(Y2); title('Binary image');

figure(5)
Y3 = bwareaopen(Y2,18);
imshow(Y3); title('Small regions are eliminated');

figure(6)
Y = imclearborder(imdilate(Y3,ones(3,3)));
imshow(Y); title('Regions are dilated and border region are eliminated');

figure(7)
Xbinview(X,Y,'y'); title('Detection');
```

The output of this code—step by step—is shown in the last row of Fig. 4.14. □

4.4 Edge Detection

In this section, we will study how the *edges* of an X-ray image can be detected. The edges correspond to pixels of the image in which the gray value changes significantly over a short distance [1]. Since edges are discontinuities in the intensity of the X-ray image, they are normally estimated by maximizing the gradient of the image. Edge detection image corresponds to a binary image (of the same size of the X-ray image), where a pixel is '1' if it belongs to an edge, otherwise it is '0', as shown in Fig. 4.15. Before we begin a more detailed description of edge detection, it is worthwhile to highlight some aspects of its relevance in the analysis of X-ray images.

The edges of an X-ray image should show the boundary of objects, e.g., boundaries of defects in control quality of aluminum castings, boundaries of the weld in welding inspection and boundaries of objects in baggage screening (Fig. 4.15). Thus, the input X-ray image is transformed into a binary image which shows structural properties of the X-ray image. The key idea is to detect objects of interest, such as defects in case of quality control or threatening objects in case of baggage screening, based on the information provided by edge detection.

In this section, we will review some basic edge detection techniques that have been used in X-ray testing: gradient estimation (Sect. 4.4.1), Laplacian-of-Gaussian (Sect. 4.4.2) and Canny (Sect. 4.4.3). Segmentation techniques based on edge detection will be outlined in Sect. 4.5.

4.4.1 Gradient Estimation

The gradient for a 1D function $f(x)$ is defined by:

$$f'(x) = \frac{\partial f}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{4.11}$$

and for a 2D function $f(x, y)$ is defined by a vector of two elements, one in x direction and the another one in y direction:

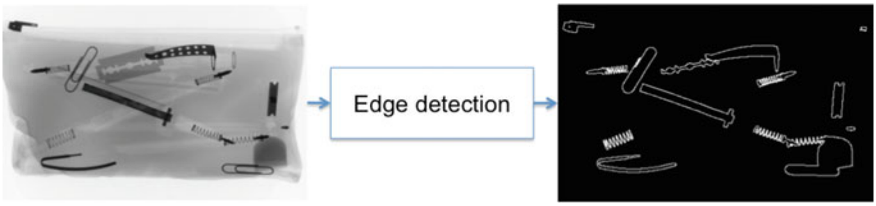


Fig. 4.15 Edge detection of an X-ray image of a pen-case. The edges correspond to the boundaries of the objects that are inside the pen-case (→ Example 4.11)

$$\nabla f(x, y) = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]. \tag{4.12}$$

In digital images, after digitalization of $f(x, y)$, however, corresponding Δx or Δy values cannot be less than one pixel. A simple way to compute the gradient of image **X** in i and j direction can be respectively:

$$G_i(i, j) = X(i+1, j) - X(i, j) \quad \text{and} \quad G_j(i, j) = X(i, j+1) - X(i, j). \tag{4.13}$$

Thus, the magnitude of the gradient can be computed as:

$$G(i, j) = \sqrt{(G_i(i, j))^2 + (G_j(i, j))^2} \tag{4.14}$$

and the direction of the gradient as:

$$A(i, j) = \arctan \frac{G_j(i, j)}{G_i(i, j)}. \tag{4.15}$$

In this formulation, gradient images $\mathbf{G}_i$ and $\mathbf{G}_j$ can be easily calculated by convolution (4.9). Thus,

$$\mathbf{G}_i = \mathbf{X} * \mathbf{h}^{\mathsf{T}} \quad \text{and} \quad \mathbf{G}_j = \mathbf{X} * \mathbf{h}. \tag{4.16}$$

where **h** is the mask used to compute the gradient in horizontal direction. For instance, if we compute the gradient using the simple way (4.13), we can use $\mathbf{h} = [-1 \ \ +1]$ in (4.16). Nevertheless, for noisy images, larger masks are suggested for (4.16). Sobel and Prewitt masks are commonly used in image processing [5]. They are defined as follows:

$$\mathbf{h}_{\text{Sobel}} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \text{and} \quad \mathbf{h}_{\text{Prewitt}} = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}. \tag{4.17}$$

For severe noise, it is recommended to use Gaussian filtering before applying gradient operators. Since Gaussian and gradient operations are linear, the Gaussian gradient operator can be defined by taking the derivative of the Gaussian (4.10):

$$h_{\text{Gauss}}(m, n) = m \cdot e^{-\frac{m^2+n^2}{2\sigma^2}}. \tag{4.18}$$

It should be noted that edges are detected when the magnitude of the gradient is maximal. That means, the location of edge pixels will not be modified if a mask **h** is replaced by $\lambda\mathbf{h}$ with $\lambda \neq 0$. Moreover, the direction of the gradient does not become modified either. For this reason, the elements of **h** are usually shown in its simplest way.

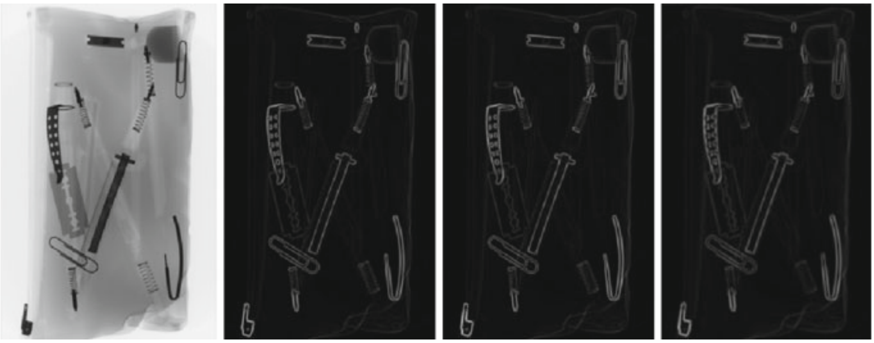


Fig. 4.16 Gradient of an X-ray of a pen-case using different masks (Sobel, Prewitt, and Gaussian). See edge detection in Fig. 4.17 (→ Example 4.5)

An example of estimation of gradient using the explained masks is illustrated in Fig. 4.16. After the gradient image is calculated, the edges are detected by thresholding. Thus, if the magnitude of the gradient is greater than a certain threshold, then the pixel of the output image is set as an edge pixel. The output for the mentioned example is illustrated in Fig. 4.17. We can see how the boundaries are detected, especially for those objects that are very dark in comparison with their background.

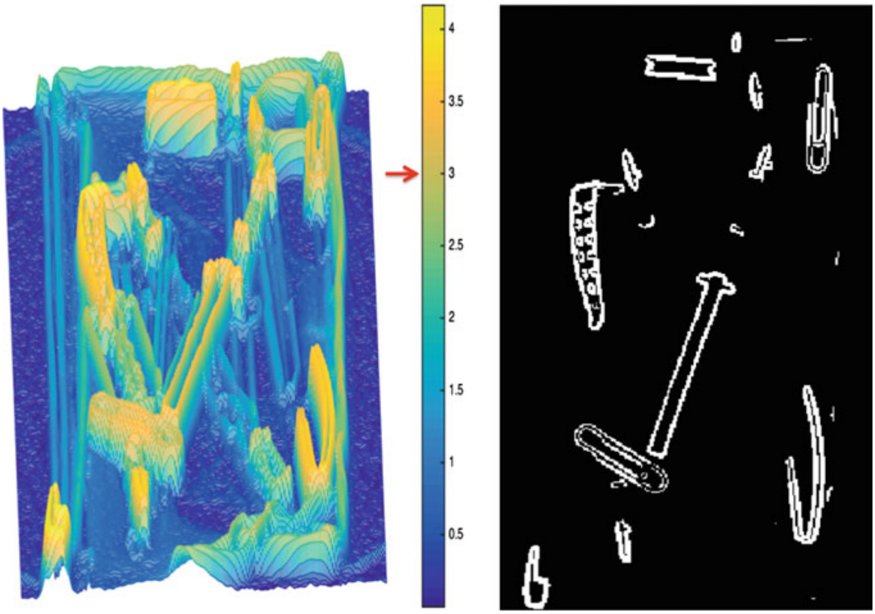


Fig. 4.17 Edge detection by thresholding a Gaussian gradient image of Fig. 4.16. The edges are detected for gradients greater than 3. In this representation, a logarithmical scale for the gray values was used (→ Example 4.5)

Matlab Example 4.5 In this example, we show the edge detection of an X-ray image of a pen-case using the gradient operators according to the method explained in this Sect. 4.5.1:

Listing 4.5 : Gradient with different masks

```
% PencaseGradient.m

close all
X   = Xloadimg('B',2,1);                             % input image
X   = imresize(X,0.25);                              % resize of input image
figure(1)
imshow(X); title('input image');

hs  = fspecial('sobel')';                            % sobel operator
hp  = fspecial('prewitt')';                          % prewitt operator
hg  = conv2(fspecial('gaussian',9,1),[-1 1],'same'); % gaussian operator

Gs  = Ximgrad(X,hs);                                 % Gradient for sobel
Gp  = Ximgrad(X,hp);                                 % ... prewitt
Gg  = Ximgrad(X,hg );                                % ... gaussian

G   = [Xlinimg(Gs) Xlinimg(Gp) Xlinimg(Gg)];         % output image
figure(2)
imshow(G); title('Original and Gradients (Sobel, Prewitt and Gaussian )')

figure(3)
Y = log(Gg+1);                                       % log representation
mesh(Y(5:end-5,end-5:-1:5)); title('3D representation of Gaussian output');
colorbar; view(-178,74); axis off

figure(4)
imshow(Y>3); title('Edge detection');                % edge detection
```

The output of this code is shown in Figs. 4.16 and 4.17. The code uses command Ximgrad (see Appendix B) of 𝕏vis Toolbox. □

4.4.2 *Laplacian-of-Gaussian*

In the previous section, we learned that the edges of a function can be located by detecting local maxima of the magnitudes of gradients. We know that the location of the maximal values of the gradient coincides with zero-crossing of the second derivative. In order to eliminate noisy zero-crossings, which do not correspond to high gradient values, this method uses a Gaussian low-pass filter (see Fig. 4.18). The method, known as Laplacian-of-Gaussian (LoG), is based on a kernel and a zero-crossing algorithm [6]. LoG-kernel involves a Gaussian low-pass filter (4.10), which is suitable for the pre-smoothing of the noisy X-ray images. LoG-kernel is defined as the Laplacian of a 2D-Gaussian function:

$$h_{LoG}(m,n) = \frac{1}{2\pi\sigma^4}\cdot\left(2-\frac{m^2+n^2}{\sigma^2}\right)\cdot e^{-\frac{m^2+n^2}{2\sigma^2}}. \tag{4.19}$$

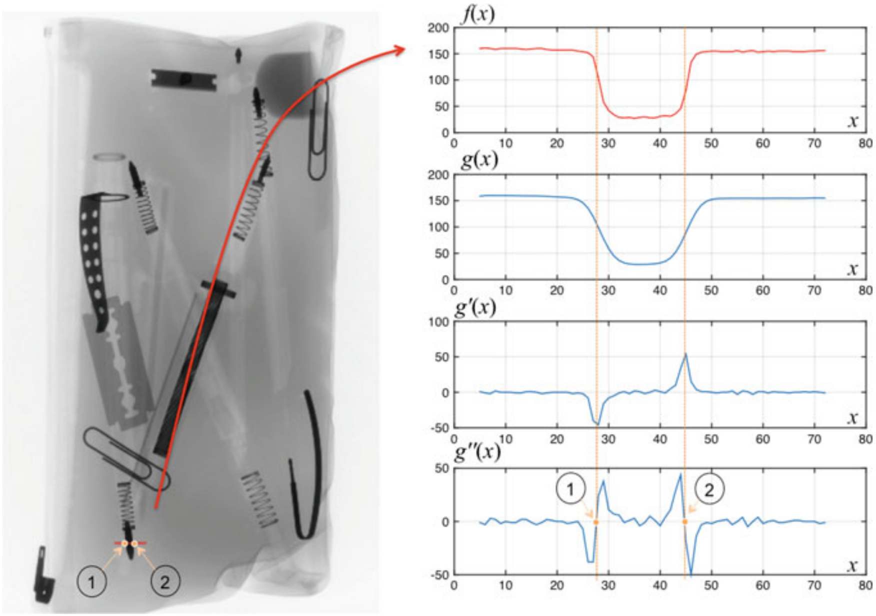


Fig. 4.18 Example of edge detection in 1D using LoG: The profile of the *red line* in an X-ray image is shown as $f(x)$. This function is filtered by a Gaussian low-pass filter obtaining $g(x)$. The gradient of $g(x)$, represented as $g'(x)$ shows the location of the maximal value (see *dashed orange lines*), that corresponds to the zero-crossing of the second derivative of $g(x)$. The edges '1' and '2' are then detected

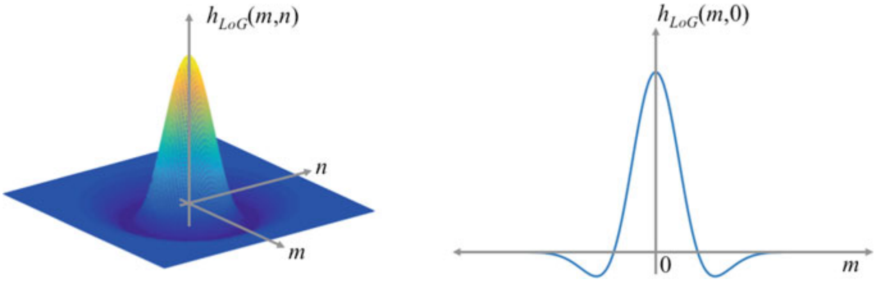


Fig. 4.19 LoG mask: *Left* representation of (4.19), *Right* profile for $n = 0$

LoG-kernel is shown in Fig. 4.19. The parameter σ defines the width of the Gaussian function and, thus, the amount of smoothing and the edges detected (see Fig. 4.20). Using (4.8) we can calculate an image $\mathbf{Y}$ in which the edges of the original image are located by their zero-crossing. After zero-crossing, the detected edges $\mathbf{Z}$ correspond to the maximal (or minimal) values of the gradient image. In order to eliminate weak edges, a threshold θ is typically used. Thus, all edge pixels in $\mathbf{Z}$ that are not strong enough are ignored. The higher the threshold, the less edges will

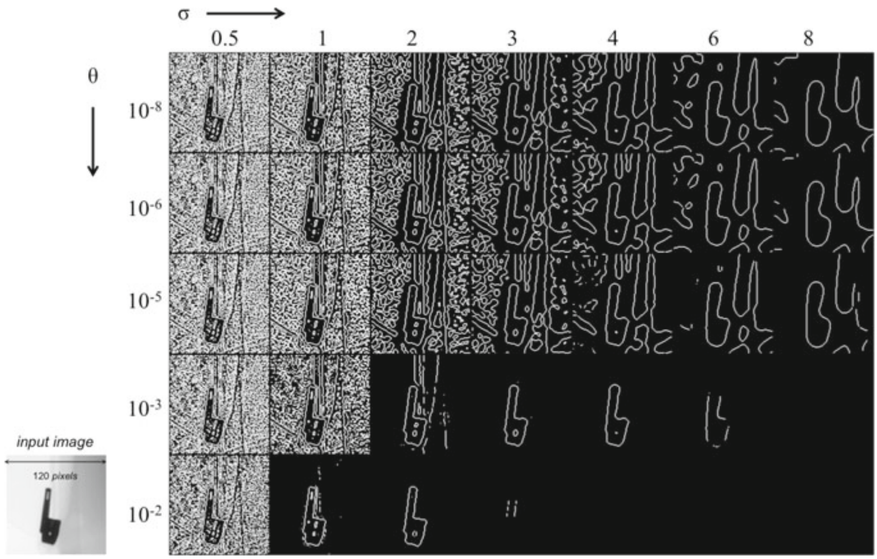


Fig. 4.20 Example of LoG-edge detection of a slider (see *bottom left* of X-ray image of the pen-case Fig. 4.18). Several values for σ and θ are presented. The smoothness of the edges is controlled by increasing σ. The reduction of noisy edges is controlled by increasing θ (→ Example 4.6)

be detected. On the other hand, if $\theta = 0$, i.e., all zero-crossings are included, the edge image has closed and connected contours. As we will see in Sect. 4.5.2, this property is required when segmenting a region of the image.

Matlab Example 4.6 In this example, we show the edge detection of the object of a pen-case (see Fig. 4.20) according to LoG algorithm explained in this Sect. 4.4.2:

Listing 4.6 : Edge detection using LoG

```
% PencaseLoG.m

close all
X = Xloadimg('B',2,1);                                  % input image
X = imresize(X,0.5);                                    % resize of input image
X = X(595:714,1:120);                                   % selected area
figure(1)
imshow(X); title('input image');

threshold = [1e-8 1e-6 1e-5 1e-3 1e-2]                  % different threshold values
sigma     = [0.5 1 2 3 4 6 8]                           % different sigma values

II = [];
for t = threshold                                       % for all thresholds
    JJ = [];
    for s=sigma                                         % for all simgas
        E = edge(X,'log',t,s);                          % edge detection
        JJ = [JJ E];                                    % row of edge images
    end
    II = [II;JJ];                                   % rows concatenation
end
figure(2)
imshow(II,[])
xlabel('sigma'); ylabel('threshold')
```

The output of this code—step by step—in Fig. 4.20. The code uses command `edge` of Image Processing Toolbox [3]. □

4.4.3 *Canny Edge Detector*

Canny proposes a 2D linear mask for edge detection based on an optimization approach [7], in which the following criteria are met:

- Good detection: The detection should respond to an edge (and not to noise).
- Good localization: The detected edge should be near the true edge.
- Single response: It should be one detected edge per true edge.

The optimal mask is similar to a derivative of a Gaussian. Thus, the idea is to use this mask to find the local maxima of the gradient of the image. The practical implementation uses adaptive thresholding of the gradient (to detect strong and weak edges) with hysteresis (weak edges are detected only if they are connected to strong edges).

The reader can change in Example 4.6 the code line `E=edge(X,'log',t,s)` by `E = edge(X,'canny',t,s)` in order to elucidate similarities and differences between both edge detectors.

4.5 Segmentation

Image segmentation is defined as the process of subdividing an image into disjointed regions [1]. A region is defined as a set of connected pixels that correspond to a certain *object of interest*. Obviously, these regions of interest depend on the application. For instance, in the inspection of aluminum castings with X-ray images, the idea of segmentation is to find regions with defects. Here, the object of interest is the defects. An example is shown Fig. 4.21, where the segmentation are the small spots that indicate defective areas.

Another example of segmentation in X-ray testing is weld inspection as illustrated in Fig. 4.22, where a weld seam with two regions is clearly distinguishable: the weld (*foreground*) and the base metal (*background*). In this example, the (first) object of interest is the weld because it is the region where defects can be present. The reader can clearly identify the defects in the weld (see small dark regions in the middle of the X-ray image). In this example, the defects, that have to be detected in a second segmentation stage, are our second object of interest. In this case, the background is the weld, and the foreground is the defects.

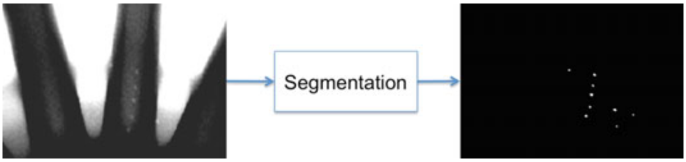


Fig. 4.21 Example of segmentation: detection of defects in an aluminum wheel (see details in Fig. 4.14) (→ Example 4.4)

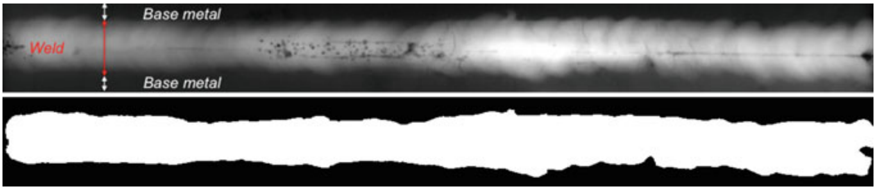


Fig. 4.22 Segmentation of a weld. *Top* original X-ray image. *Bottom* segmentation. The first step in weld inspection is the segmentation of the weld, i.e., the region where the defects can be present (see segmentation in Fig. 4.27). The second step is the detection of defects (→ Example 4.8)

Segmentation is one of the most difficult processes in image processing. Clearly, there are some simple applications in which certain segmentation techniques are very effective (e.g., separation between weld and metal base as shown in Fig. 4.22), however, in many other applications segmentation is far from being solved as the appearance of the object of interest can become very intricate. This is the case of baggage screening, where the segmentation of objects of interest inside a piece of luggage can be extremely difficult due to problems of (self-)occlusion, noise, and acquisition (see Fig. 4.23).

In image processing for X-ray testing, segmentation is used to detect (potential) regions that can be the objects of interest that we are looking for. As mentioned in previous examples, segmentation divides the X-ray image into two areas: foreground and background. Foreground means the pixels of the object(s) of interest. Background means the remaining pixels of the image. Usually, a *binary image* is the output of the segmentation process as we can see in Figs. 4.21 and 4.22: where a pixel equals to ‘1’ (white) is foreground, whereas ‘0’ (black) means background.

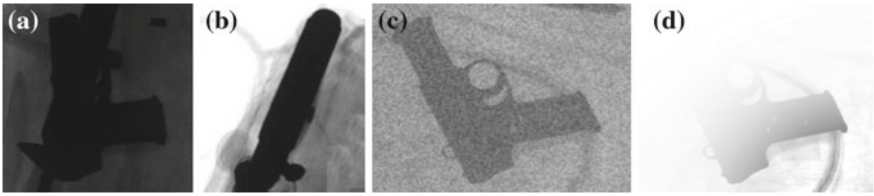


Fig. 4.23 Problems when detecting a gun. Detection can be a very complex task due to **a** occlusion, **b** self–occlusion, **c** noise, **d** wrong acquisition

We use the term ‘potential’ throughout to make it clear that a segmented region is not necessarily the final detected region. In many applications, the segmentation is just the first step of the whole detection process. In such cases, an additional step that analyzes the segmented region is required. This additional step can include multiple view analysis or a pattern recognition technique (see Fig. 1.21). The later extracts and classifies features of the segmented region in order to verify whether it corresponds to the object that we are detecting or it is a *false detection*.

Thus, segmentation basically acts as a focus of attention mechanism that filters the information that is fed to the following steps, as such a failure in the segmentation is catastrophic for the final performance. In this section, we will review some basic segmentation techniques that have been used in X-ray testing: thresholding (Sect. 4.5.1), region growing (Sect. 4.5.2) and maximally stable extremal regions (Sect. 4.5.3). Please note that more complex techniques based on computer vision algorithms will be addressed in the next sections.

4.5.1 Thresholding

In some X-ray images, we can observe that the background is significantly darker than the foreground. This is the case of an X-ray image of an apple placed on a uniform background as illustrated in Fig. 4.24. It is clear that the object of interest can be segmented using a very simple approach based on *thresholding*. In this section, we will explain a methodology based on two steps: (i) estimate of a global threshold using a statistical approach, and (ii) a morphological operation in order to fill the possible holes presented in the segmented binary image. This method was originally presented for color food images [8], however, it can be easily adapted for X-ray images.

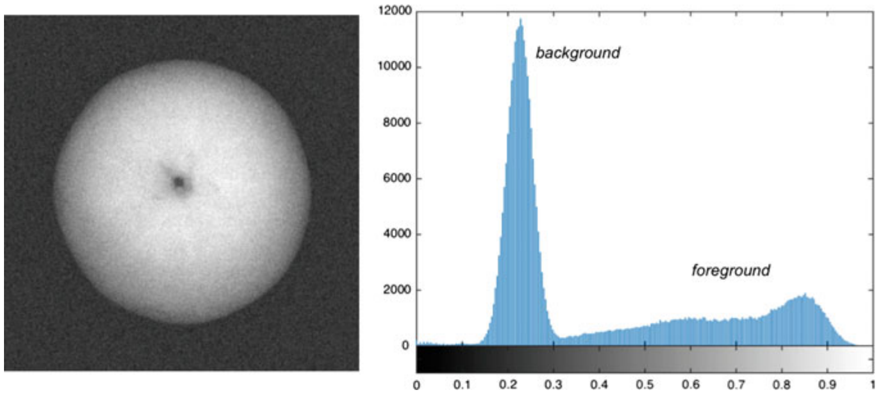


Fig. 4.24 X-ray image of an apple and its histogram

The X-ray image to be segmented is stored in matrix **I**. In order to enhance the contrast of the image, a linear transformation can be performed (see Sect. 4.2.2). Additionally, a linear or nonlinear filter can be used for noise removal (see Sect. 4.3). Here, after image enhancement and filtering, we obtain a new image **J** where $J_{\max} = 1$ and $J_{\max} = 0$. Image **J** has a bimodal histogram as shown in Fig. 4.24, where the left distribution corresponds to the background and the right to the food image. In this image, a first separation between foreground and background can be performed estimating a global threshold t. Thus, we define a binary image

$$K(i, j) = \begin{cases} 1 & \text{if} \quad J(i, j) > t \\ 0 & \text{else} \end{cases} \tag{4.20}$$

where '1' means foreground and '0' background, that define two classes of pixels in the image. Figure 4.25 illustrates different outputs depending on t. The problem is to determine the 'best' threshold t that separates the two modes of the histogram from each other. A good separation of the classes is obtained by ensuring (i) a small variation of the gray values in each class, and (ii) a large variation of the gray values in the image [9]. The first criterion is obtained by minimizing a weighted sum of the within-class variances (called *intraclass* variance $\sigma_W^2(t)$):

$$\sigma_W^2(t) = p_b(t)\sigma_b^2(t) + p_f(t)\sigma_f^2(t) \tag{4.21}$$

where the indices 'b' and 'f' denote respectively background and foreground classes, and p and σ^2 are, respectively, the probability and the variance for the indicated class. These values can be computed from the histogram.

The second criterion is obtained by maximizing the between-class variance (called *interclass* variance $\sigma_B^2(t)$):

$$\sigma_B^2(t) = p_b(\mu_b(t) - \mu)^2 + p_f(\mu_f(t) - \mu)^2 \tag{4.22}$$

where μ_b, μ_f and μ indicate the mean value of the background, foreground and the whole image respectively.

The best threshold t can be estimated by a sequential search through all possible values of t that minimizes $\sigma_W^2(t)$ (or maximizes $\sigma_B^2(t)$). Both criteria, however, lead to the same result because the sum $\sigma_W^2 + \sigma_B^2$ is a constant and corresponds to the variance of the whole image [9]. Matlab computes the global image threshold by minimizing the intraclass variance $\sigma_W^2(t)$. The threshold can be obtained with the function `graythresh` [3]. In our example, the obtained threshold is $t = 0.4824$, that is approximately 0.5 (see Fig. 4.25).

We can observe in Fig. 4.25 that the segmentation suffers from inaccuracy because there are many dark (bright) regions belonging to the foreground (background) that are below (above) the chosen threshold and therefore misclassified. For this reason, additional morphological processing must be obtained.

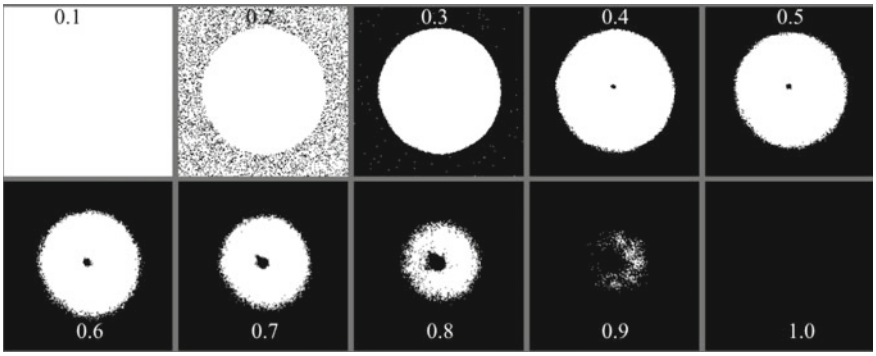


Fig. 4.25 Segmentation using threshold $t = 0.1, 0.2, \ldots 1.0$

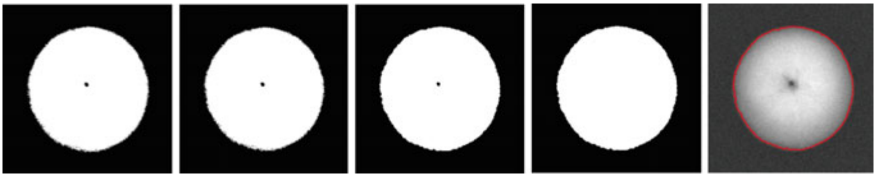


Fig. 4.26 Additional morphological operations. From *left* to *right*: **K**: binary image after thresholding, **A**: after removal of small objects, **C**: after closing process, **R**: after filling holes, and boundary superimposed onto the original image (→ Example 4.7)

The morphological operation is performed in three steps as shown in Fig. 4.26: (i) remove small objects, (ii) close the binary image and (iii) fill the holes.

In the first step, we remove from binary image **K** obtained from (4.20) all connected regions that have fewer than n pixels (see image **A** in Fig. 4.26).This operation is necessary to eliminate those isolated pixels of the background that have a gray value greater than the selected threshold. Empirically, we set $n = NM/100$, where $N \times M$ is the number of pixels of the image.

The second step *closes* the image, i.e., the image is *dilated* and then *eroded*. The dilation is the process that incorporates into the foreground the background pixels that touch it. On the other hand, erosion is the process that eliminates all the boundary pixels of the foreground. The closing process (dilation followed by erosion) fills small holes and thins holes in the foreground, connecting nearby regions, and smoothing the boundaries of the foreground without changing the area significantly [1] (see image **C** in Fig. 4.26). This operation is very useful in objects that have spots in the boundary.

Finally, the last operation fills the holes in the closed image (see image **R** in Fig. 4.26). We use this operation to incorporate into the foreground all pixels '0' that are inside of the region. The whole algorithm is implemented in command Xsegbimodal (see Appendix B) of 𝕏vis Toolbox. In the implementation, as suggested in [8], an offset p that modifies the threshold is used because there are dark zones in the boundary that are not well included in the original segmented region.

Matlab Example 4.7 In this example, we show the segmentation of an X-ray image of an apple (see Fig. 4.24) according to the method explained in this Sect. 4.5.1:

Listing 4.7 : Apple segmentation using global thresholding

```
% AppleSegmentation.m

close all
I = Xloadimg('N',5,9);                      % input image (a fruit)
figure(1)
imshow(I);title('Input image');

[R,E] = Xsegbimodal(I);                     % bimodal segmentation and boundary
figure(2)
imshow(R); title('Segmetation');

figure(3)
Xbinview(I,E,'r',2); title('Segmentation'); % boundary of the segmented region
```

The output of this code—step by step—in Fig. 4.26. The code uses command Xsegbimodal (see Appendix B) of 𝕏vis Toolbox. □

The above-mentioned methodology, based on a *global threshold*, does not segment appropriately when there is a large variation in the background or foreground intensity. For this reason, in certain cases, it is recommended to use an *adaptive threshold*. The idea is to divide the input image into partitions with some overlapping. Each partition is handled as a new image that is segmented by thresholding (using a global but an *ad hoc* threshold for each partition). The output image is a fusion of all segmented partitions, e.g., using logical OR operator. The next example shows an implementation that was used to segment the weld of Fig. 4.22. Since the weld area is horizontal, the proposed method uses vertical partitions that include background and foreground areas. The segmentation of each partition is performed by the same method used for the segmentation of the apple.

Matlab Example 4.8 This example shows the segmentation of a weld of Fig. 4.22 using adaptive thresholding. The approach is simple: the input image is divided into four partitions with an overlapping of 50 %. Each partition is segmented using command Xsegbimodal (see Appendix B) of 𝕏vis Toolbox. The obtained binary images of the segmentation are superimposed using logical OR operator:

Listing 4.8 : Weld segmentation using adaptive thresholding

```
% WeldSegmentation.m

close all
I  = Xloadimg('W',1,1);                         % input image
figure(1)
imshow(I);title('Input image');

R  = zeros(size(I));                            % initialization of segmentation

M  = size(I,2);                                 % width of the image
d1 = round(M/4);                                % 4 partitions
```

```
d2 = round(d1*1.5);                             % width of each partition
i1 = 1;                                         % first column of partition

while i1<M
    i2 = min([i1+d2 M]);                        % second column of partition
    Ii = I(:,i1:i2);                            % partition i
    Ri = Xsegbimodal(Ii);                       % segemntation of partition i
    R(:,i1:i2) = or(R(:,i1:i2),Ri);             % addition into whole segmentation
    i1 = i1+d1;                                 % update of first column
end
E = bwperim(R,4);                               % edge of segmentation
figure(2);
Xbinview(I,E,'r',5); title('Segmentation');     % boundary of the segmented region
```

The output of this code—step by step—is shown in the last row of Fig. 4.27. □

4.5.2 Region Growing

In region growing, we segment a region using an iterative approach. We start by choosing a seed pixel, as shown in Fig. 4.28. At this moment, our region is initialized and its size is one pixel only. We extract some feature of the region, e.g., the gray value. We extract the same feature of each neighboring pixel. In our example, there are four neighbors (up, down, right, and left), as we can see in third image of Fig. 4.28. We increase our region by adding similar neighboring pixels, i.e., those neighboring pixels that have a similar feature to the region. The whole process is continued, each added pixel is a new seed for the next iteration, until no more neighboring pixels can be added.

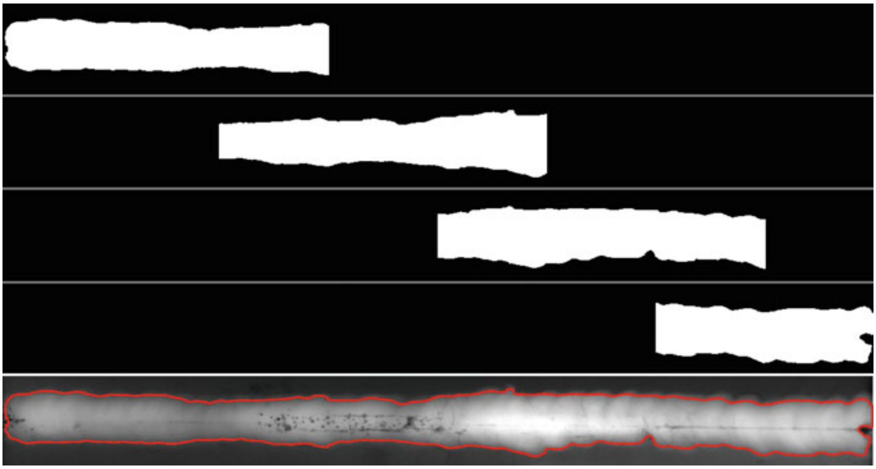


Fig. 4.27 Weld segmentation of Fig. 4.22 using adaptive thresholding of four partitions. The last image shows the segmentation after fusion the four individual segmentations using logical OR operator (→ Example 4.8)

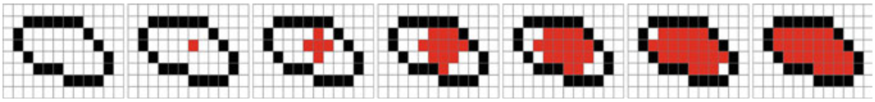


Fig. 4.28 Region growing: we start with a seed pixel that grows in each iteration in four directions until a boundary is found. The directions in this example are four: up, down, right, and left

In Fig. 4.28, we have a binary edge image. The feature that we use to establish the similarity is the value of the pixel. In our example, there are only two pixel values: ‘0’ for the edge pixels, and ‘1’ for the remaining pixels. That means, that the value of the pixel of the seed is ‘1’ and in each iteration, we can add only those neighboring pixels the value of which are ‘1’. As we can see, the red region grows up from 1 pixel to 5, 12, 16, 22, and finally 24 pixels. The output is the red region of the last step.

Region growing can be used directly in X-ray images as illustrated in Fig. 4.29. We start with a seed pixel, and neighboring pixels are added if they are similar enough.

Matlab Example 4.9 In this example, we show the performance of region growing in the segmentation of an object in an X-ray image of a pen-case (see Fig. 4.29). The seed is chosen at pixel (190,403). The seed grows by adding neighboring pixels with similar gray values. We use command Xregiongrowing (see Appendix B) of 𝕏vis Toolbox. In this implementation, the similarity between region and neighboring pixels is established if $|\bar{R} - r_n| \leq \theta$, where $\bar{R}$ is the average of the gray values of the region, r_n is the gray value of the neighboring pixel, and θ is a threshold. In this example, $\theta = 20$:

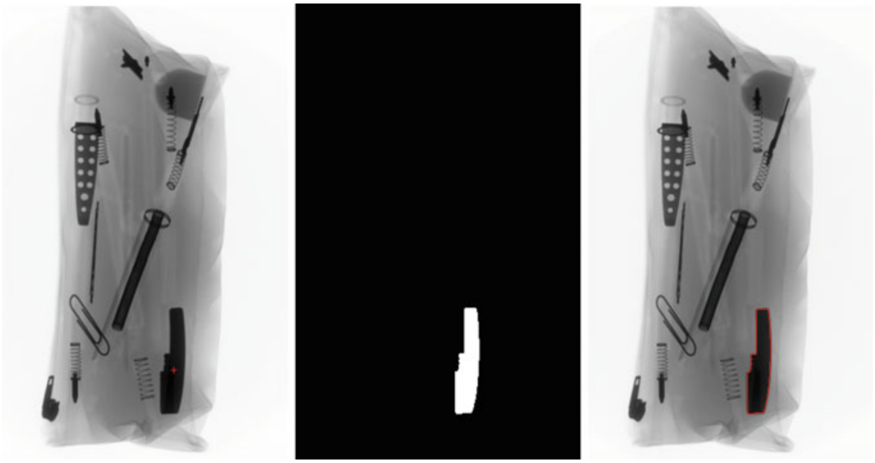


Fig. 4.29 Region growing in an X-ray image using a seed pixel in the object of interest. The region is well segmented as we can see in the binary image and in boundaries (→ Example 4.9)

Listing 4.9 : Region Growing

```
% PencaseRegionGrowing.m
close all
X = Xloadimg('B',3,4);              % input image
X = imresize(X,0.35);               % resize of input image
th = 40;                            % threshold
si = 403; sj = 190;                 % seed
Y  = Xregiongrowing(X,th,[sj si]); % segmentation of the selected region
figure(1)
imshow(X,[]); title('input image');
hold on
plot(sj,si,'r+');                   % seed pixel
figure(2)
imshow(Y);title('segmented region');
figure(3)
Xbinview(X,bwperim(Y));title('edges of the region');
```

The output of this code is shown in Fig. 4.29. □

Region growing can be used in X-ray testing in defect detection (see for example interesting approaches in aluminum castings [10] and welds [11]). The method is illustrated in Fig. 4.30. The method uses an edge detection algorithm to obtain an edge image with closed and connected contours around the real defects. Thus, we use region growing to isolate each region enclosed by edges. The idea is to extract features from this isolated region (e.g., area, average of gray value, contrast, etc.) that can be used in a classification strategy. In our example, a region is segmented using a very simple classifier (the features of a segmented region must be in certain ranges, e.g., $A_{\min} \leq \text{Area} \leq A_{\max}$). Obviously, more sophisticated features and classifiers can be used to improve the segmentation performance in more complex scenarios as we will see in the following chapters.

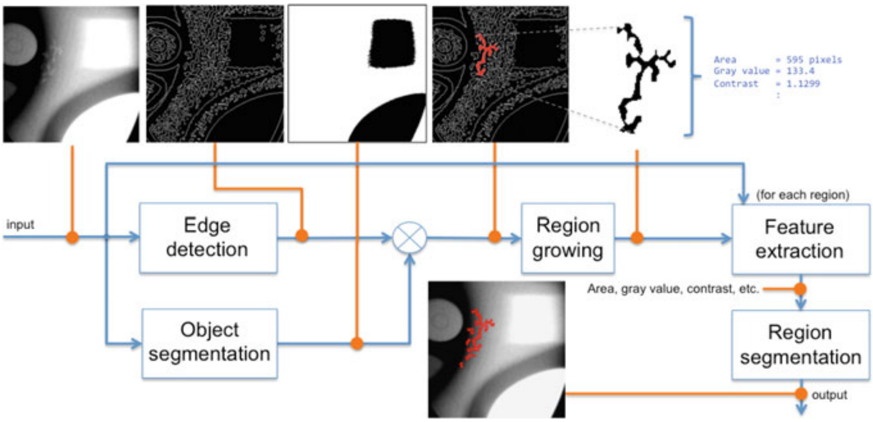


Fig. 4.30 Segmentation of defects in aluminum castings using region growing, edge detection, and some features. The size of the image in this example is 286 × 286 pixels (→ Example 4.10)

Matlab Example 4.10 In this example, we show how to segment defects in aluminum castings using binary images of potential defects and some simple features that can be extracted from each potential region. In this example, we segment all those regions the area of which is between 200 and 2000 pixels, the average of the gray value is less than 150, and the contrast is greater than 1.1:

Listing 4.10 : Region Growing

```
% SegmentationCastDefect.m
X = Xloadimg('C',31,19);                               % input image
X = X(1:2:572,1:2:572);
figure(1)
imshow(X); title('input image');

R = X<240;                                             % casting segmentation
figure(2);imshow(R);title('segmented object');

Amin  = 30;  Amax = 2000;                              % Area range
Gmin  = 0;   Gmax = 150;                               % Gray value range
Cmin  = 1.1; Cmax = 3;                                 % Contrast range
sigma = 2.5;                                           % sigma of LoG


Y = Xseglogfeat(X,R,[Amin Amax],[Gmin Gmax],...        % Segmentation
                    [Cmin Cmax],sigma);
figure(3)
Xbinview(X,bwperim(Y>0)); title('segmented regions')   % Edges of the segmentation
```

The output of this code is shown—step by step—in Fig. 4.30. We use command Xseglogfeat (see Appendix B) of 𝕏vis Toolbox. □

This method is very effective for regions of interest that have gray values significantly different from the background (the reader, for instance, can try to segment the objects of the pen-case of Fig. 4.29 using command Xseglogfeat (see Appendix B) of 𝕏vis Toolbox).

Nevertheless, the method may fail if the boundaries do not close a region of interest. This is the case in some defects of aluminum castings that are at an edge of a regular structure as illustrated in Fig. 4.31.[2] In this problem, we can see that the edges of LoG algorithm (and other edge detection algorithms like Sobel or Canny as well) cannot correctly find the defect's edge. Contrarily, it finds the regular structure's edge. To overcome this problem, we have to complete the remaining edges of these defects. A simple approach was suggested in [12] by thickening of the edges of the regular structure after LoG-edge detection: (i) The gradient of the original image is calculated. The gradient image is computed by taking the square root of the sum of the squares of the gradient in horizontal and in vertical directions. These are calculated by the convolution of the radioscopic image with the first derivative (in the corresponding direction) of the Gaussian low-pass filter used in the LoG filter. (ii) High gradient pixels are detected by thresholding. (iii) The resulting image is added to the LoG-edge detection image. Afterwards, each closed region is

[2] A video of this small defect can be watched at http://youtu.be/e3wDJhq2Tqg.

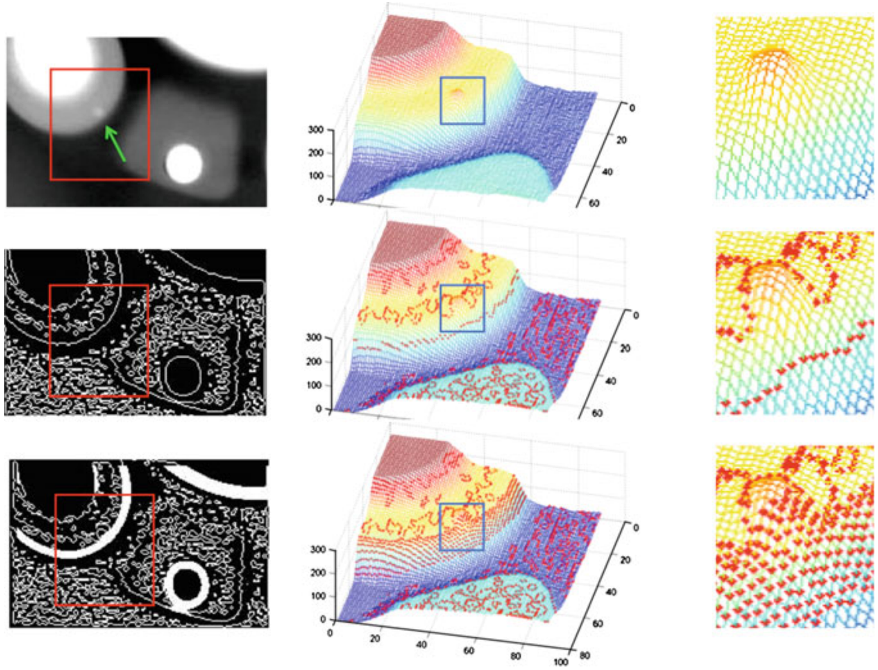


Fig. 4.31 X-ray image of an aluminum casting with a small defect at an edge (see defect pointed by *green arrow*). *First row* original image. *Second row* LoG. *Third row* LoG and high gradient pixels. *First column* image representation. *Second column* 3D representation of *red square*. *Third column* zoom of *blue square*. In this representation, the edge pixels are represented as *red points* superimposed onto the 3D surface. The output of this method is a binary image in which the real defects are closed by edges

segmented as a potential flaw. As can be observed the effectiveness of this method in Fig. 4.31, the defect on an edge of a regular structure could be satisfactory closed. Thus, the method of Fig. 4.30 can be used.

4.5.3 Maximally Stable Extremal Regions

In order to understand the MSER approach [13], the reader can imagine a simple video as follows. The video will have 256 frames. Frame t is defined as the binary image $\mathbf{I} < t$, where $\mathbf{I}$ is the input image we want to segment. If the binary image is black for '0' and white for '1', at the beginning our video will be very dark and at the end very bright. In the middle, we will have some regions depending on the threshold.[3] Thus, each region has an area $A(t)$, that depends on t. If the gray value

[3] The video can be found in http://youtu.be/tWdJ-NFE6vY.

of the region is very different from its background, the area of this region will be stable for some thresholds $t, t+1 \ldots t+p$, i.e., $A(t) \approx A(t+1) \cdots \approx A(t+p)$. The key idea of MSER, is to segment those regions which fulfill:

$$\frac{\Delta A}{\Delta t} < \theta, \tag{4.23}$$

where θ is a threshold. That means, those regions whose sizes remain approximately stable by varying the segmentation threshold t are to be detected.

Matlab Example 4.11 In this example, we show the segmentation of an X-ray image of a pen-case (see Fig. 4.15) according to MSER approach (see Sect. 4.5.3):

Listing 4.11 : Pencase segmentation using MSER algorithm

```
% PencaseSegmentation.m

close all
I    = Xloadimg('B',2,1);                            % input image
figure(1)
imshow(I); title('Input image');

[fr,sd,J] = Xsegmser(I,[15 20000 0.9 0.2 6 2 0]);    % MSER segmentation
E = imdilate(bwperim(J),ones(3,3));                  % edges
figure(2)
imshow(E); title('Edges');

L = bwlabel(J);
figure(3)
imshow(L,[]); title('Segmentation');
```

The output of this code—step by step—in Fig. 4.15. The code uses command Xsegmser (see Appendix B) of 𝕏vis Toolbox. This command uses VLFeat Toolbox [14]. □

4.6 Image Restoration

Image restoration involves recovering detail in severely blurred images. This process is more efficient when the causes of the imperfections are known a-priori [5]. This knowledge may exist as an analytical model, or as a-priori information in conjunction with knowledge (or assumptions) of the physical system that provided the imaging process in the first place. The purpose of restoration then is to estimate the best source image, given the blurred example and some a-priori knowledge.

In this section, we concentrate on the particular case of blur caused by uniform linear motion, which may be introduced by relative motion between detector and object. Early work on restoring an image degraded by blurring calculated the deblurring function as an inverse filtering. The inverse filtering evaluation of the blurring function h (or point spread function PSF) in the frequency domain tends to be very sensitive to noise [5]. The cause of this sensitivity is the low-pass nature of the PSF: its frequency response $H(\omega)$ contains very small values, and small noise in the frequency regions where $1/H(\omega)$ is very large, may be greatly emphasized. Sondhi [5], proposed a non-iterative algorithm to find a solution to the uniform-blurring case, but the computational load is extremely high in small motions. Another two non-iterative approaches are presented in [3]. In the first one, the matrix left division calculates the restored signal as a signal that has the fewest possible nonzero components. This solution differs strongly from the original signal because the original signal must not have necessarily many zero components. The second one, the Moore-Penrose pseudo-inverse of a matrix, finds a restored signal whose norm is smaller than any other solution. This solution is very good, but the estimation is based on Singular-Value Decomposition (SVD), whose computation load is very high. In this section, we address the above problems and reduce the computational times significantly using a new technique that minimize the norm between blurred and original.

A blurred X-ray image $g(x, y)$ that has been degraded by a motion in the vertical direction x and the horizontal direction y can be modeled by:

$$g(x, y) = \frac{1}{T}\int_0^T f(x - x_t(t), y - y_t(t))dt, \tag{4.24}$$

where f, T, $x_t(t)$ and $y_t(t)$ represent, respectively, the deterministic original X-ray image, the duration of the exposure, and the time-varying component of motion in the x and y directions. In this case, the total exposure is obtained by integrating the instantaneous exposure over the time interval during which the shutter is open. By rotation of the camera or by using a transformation that rotates the blurred image, a new system of coordinates is chosen in which $x_t(t)$ is zero. Considering that the original image $f(x, y)$ undergoes uniform linear motion in the horizontal direction y only, at a rate given by $y_t(t) = ct/T$, let us write (4.24), with $u = y - ct/T$, as:

$$g(y) = \frac{1}{T}\int_0^T f(y - ct/T)dt = \frac{1}{c}\int_{y-c}^{y} f(u)dt, \tag{4.25}$$

or as a digital that has been discretized in spatial coordinates by taking N samples $\Delta y = Y/N$ units apart:

$$g_k = \frac{1}{n}\sum_{i=0}^{n-1} f_{k+i}, \tag{4.26}$$

where

$$g_k = g\left(y_0 + (k-1)\frac{c}{n}\right), \quad f_k = f\left(y_0 + (k-1)\frac{c}{n} - c\right), \tag{4.27}$$

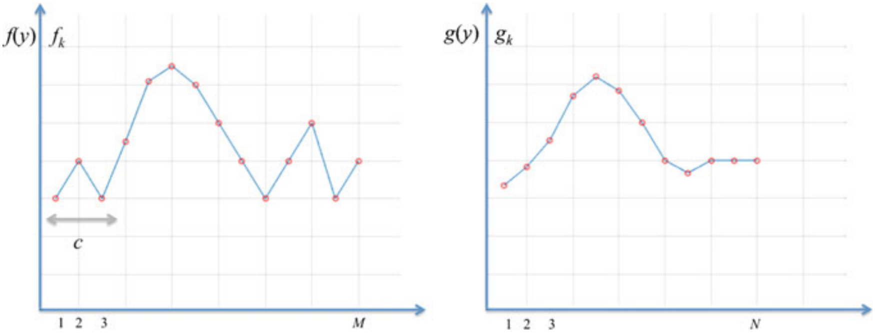


Fig. 4.32 Blurring process: *Left* original row f. *Right* Blurred row

with $n = c/\Delta y$. Figure 4.32 shows a row $\mathbf{f} = [f_1 \ \dots \ f_M]^\mathsf{T}$ of an original image and its corresponding row $\mathbf{g} = [g_1 \ \dots \ g_N]^\mathsf{T}$ of the blurred image for $n = 3$ pixels. Equation (4.26) describes an underdetermined system of N simultaneous equations (one for each element of vector $\mathbf{g}$) and $M = N + n - 1$ unknowns (one for each element of vector $\mathbf{f}$) with $M > N$. This process is carried out for each row of the image. The degradation of $\mathbf{f}$ can be modeled using a convolution of $\mathbf{f}$ with $\mathbf{h}$, where $\mathbf{h}$ is the PSF, a n-element vector defined as the impulse response of this linear system [5]. Thus, element g_i of vector $\mathbf{g}$ is calculated as a weighted sum of n elements of $\mathbf{f}$, i.e., $g_i = h_1 f_i + h_2 f_{i+1} + \cdots + h_n f_{i+n-1}$, for $i = 1, \dots, N$. Using a circulant matrix, the convolution can be written as $\mathbf{Hf} = \mathbf{g}$:

$$\mathbf{g} = \mathbf{f} * \mathbf{h} = \begin{bmatrix} h_1 \dots h_n & 0 & 0 & 0 & 0 \\ 0 & h_1 \dots h_n & 0 & 0 & 0 \\ \vdots & & & \vdots & \\ 0 & 0 & \dots & 0 & h_1 \dots h_n \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_M \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_N \end{bmatrix} \tag{4.28}$$

An example of a degradation process is shown in Fig. 4.33. In this example, we can see how the objects cannot be recognized when the degradation is severe.

If the PSF is not exactly known, but if we know that it corresponds to a uniform linear motion, the parameter n can be estimated from the spectrum of the blurred image. An example is shown in Fig. 4.34. The 2D-Fourier Transformation of a blurred test is represented in Fig. 4.34a, in this case a horizontal degradation took place with $n = 32$. The mean of its rows is illustrated in Fig. 4.34b. We can

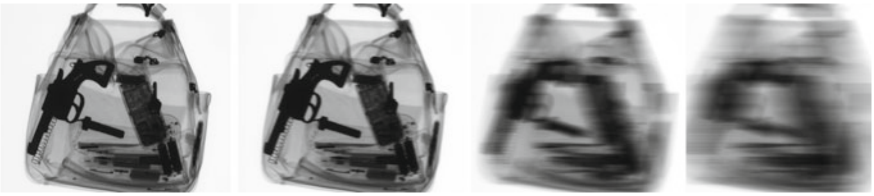


Fig. 4.33 Degradation of an X-ray image of 2208 × 2688 pixels: Original image and degraded images with n = 32, 256, and 512 pixels

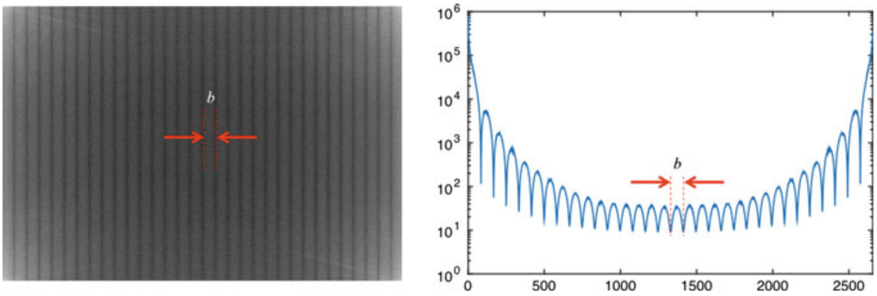


Fig. 4.34 Spectrum of a blurred image which was degraded by uniform linear motion with $n = 32$ pixels. *Left* 2D-Fourier Transformation of the original image of Fig. 4.33. *Right* Mean of the rows of the Fourier transformation. The size of the degraded image is 2208 × 2657 pixels. It can be demonstrated that b is approximately $2657/n$

observe that the period of this function is inversely proportional to the length of the blurring process in pixels.

The problem of restoring an X-ray that has been blurred by uniform linear motion consists of solving the underdetermined system (4.28). The objective is to estimate an original row per row ($\mathbf{f}$), given each row of a blurred ($\mathbf{g}$) and a-priori knowledge of the degradation phenomenon ($\mathbf{H}$). Since there is an infinite number of exact solutions for $\mathbf{f}$ in the sense that $\mathbf{Hf} - \mathbf{g} = \mathbf{0}$, an additional criterion that finds a sharp restored is required.

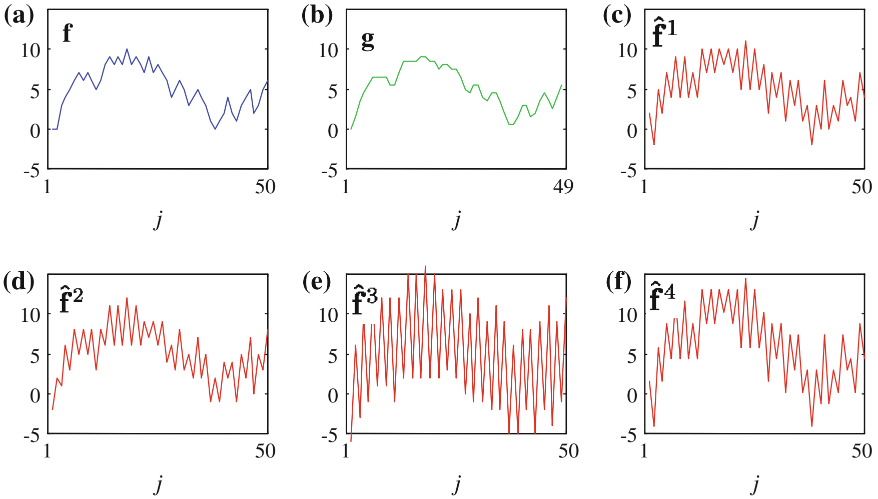


Fig. 4.35 Restoration of row $\mathbf{f}$: **a** original row, **b** degraded row with $n = 2$, **c**–**f** four possible solution that satisfy $\mathbf{H}\hat{\mathbf{f}} = \mathbf{g}$

We observed that most solutions for $\mathbf{f}$ strongly oscillate. Figure 4.35 shows an example in which four different solutions for $\mathbf{f}$ are estimated, all solutions satisfy Eq. (4.28): $\mathbf{Hf} = \mathbf{g}$. Although these solutions are mathematically right, they do not correspond to the original signal. By the assumption that the components of the higher frequencies of $\mathbf{f}$ are not so significant in the wanted solution, these oscillations can be reduced by minimization of the distance between f_k and g_k, i.e., we take a vector as a sharp solution of $\mathbf{Hf} = \mathbf{g}$ so, that this presents the smallest distance between original signal and blurred signal: we seek then to minimize the objective function

$$J(\mathbf{f}, \mathbf{g}) = \sum_{k=1}^{N} (f_k - g_k)^2 \to \min. \tag{4.29}$$

The application of criteria of the minimization of the norm between input and output (MINIO) does not mean that $\mathbf{f}$ is equal to $\mathbf{g}$, because this solution does not satisfy the system of equations (4.28) and the size of $\mathbf{f}$ and $\mathbf{g}$ are different. The solution also is defined as the vector in the solution space of the underdetermined system $\mathbf{Hf} = \mathbf{g}$ whose first N components has the minimum distance to the measured data, i.e., where the first N elements are of $\mathbf{f}$. We can express vector $\hat{\mathbf{f}} = \mathbf{Pf}$, with $\mathbf{f}$ a $N \times M$ matrix which projects the vector $\mathbf{f}$ on the support of $\mathbf{g}$:

$$\mathbf{P} = \begin{bmatrix} 1\,0 \ldots 0\,0 \ldots 0 \\ 0\,1 \ldots 0\,0 \ldots 0 \\ \quad\ \vdots \qquad\ \vdots\ \ 0 \\ 0\,0 \ldots 1\,0 \ldots 0 \end{bmatrix}. \tag{4.30}$$

The original optimization problem is now:

$$\hat{\mathbf{f}} = \underset{\mathbf{f}}{\operatorname{argmin}} \parallel \mathbf{Pf} - \mathbf{g} \parallel^2 \tag{4.31}$$

subject to the constraint $\parallel \mathbf{Pf} - \mathbf{g} \parallel^2 = \mathbf{0}$. Applying the technique of *Lagrange multipliers* this problem can be alternatively formulated as an optimization problem without constraints:

$$V(\mathbf{f}) = \lambda \parallel \mathbf{Hf} - \mathbf{g} \parallel^2 + \parallel \mathbf{Pf} - \mathbf{g} \parallel^2 \to \min, \tag{4.32}$$

if λ is large enough. The solution of this problem can be easily obtained by computing the partial derivative of criterion V with respect to the unknown $\mathbf{f}$:

$$\frac{\partial}{\partial \mathbf{f}} V(\mathbf{f}) = 2\lambda \mathbf{H}^\mathsf{T}(\mathbf{Hf} - \mathbf{g}) + 2\mathbf{P}^\mathsf{T}(\mathbf{Pf} - \mathbf{g}) = \mathbf{0}, \tag{4.33}$$

then is

$$\hat{\mathbf{f}} = \left[\lambda \mathbf{H}^\mathsf{T}\mathbf{H} + \mathbf{P}^\mathsf{T}\mathbf{P}\right]^{-1} [\lambda \mathbf{H} + \mathbf{P}]\, \mathbf{g}. \tag{4.34}$$

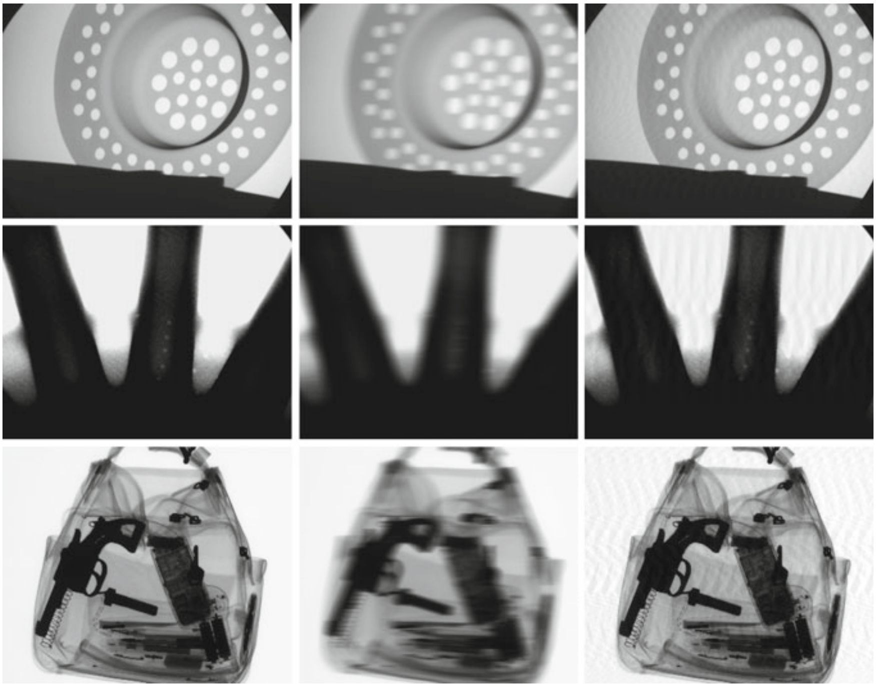


Fig. 4.36 Restoration in simulated degraded X-ray images. Each column shows the original, the degraded with n pixels and the restored images. The size of the images are respectively: 574×768, with $n = 30$; 574×768, with $n = 40$; and 2208×2688, with $n = 128$ (→ Example 4.12)

This solution for the example of Fig. 4.35b is almost identical to the original sharp input signal of Fig. 4.35a. Figure 4.36 shows three different restoration examples.

Matlab Example 4.12 In this example, we simulate an X-ray image that has been degraded by a horizontal motion. The image is restored using MINIO algorithm (4.34):

Listing 4.12 : X-ray image restoration.

```
% MinioRestauration.m
F  = Xloadimg('B',46,90);                        % original image
n  = 128;                                        % amount of blur in pixels
h  = ones(1,n)/n;                                % PSF
G  = conv2(double(F),h,'valid');                 % degradation
Fs = Xresminio(G,h);                             % restoration
figure(1)
imshow(F,[]) ;title('original image');
figure(2)
imshow(G,[]) ;title('degraded image');
figure(3)
imshow(Fs,[]);title('restored image');
```

The output of this code is shown in the last row of Fig. 4.36. Details of the baggage are not discernible in the degraded image, but are recovered in the restored image. In this code, we use function Xresminio (see Appendix B) of 𝕏vis Toolbox. This function computes MINIO restoration algorithm as defined in (4.34). □

The restoration quality is equally as good as the classical methods (see for example [5]), while the computation load is decreased considerably (see comparisons in [15]).

4.7 Summary

In this chapter, we covered the main techniques of image processing used in X-ray testing.

They are:

- Image preprocessing: Noise removal, contrast enhancement, and shading correction.
- Image Filtering: linear and nonlinear filtering.
- Edge detection: Gradient estimation, Laplacian-of-Gaussian, and Canny.
- Image segmentation: Thresholding, region growing, and maximally stable extremal regions.
- Image restoration: Minimization of the norm between input and output.

The chapter provided a good overview, presenting several methodologies with examples using real and simulated X-ray images.

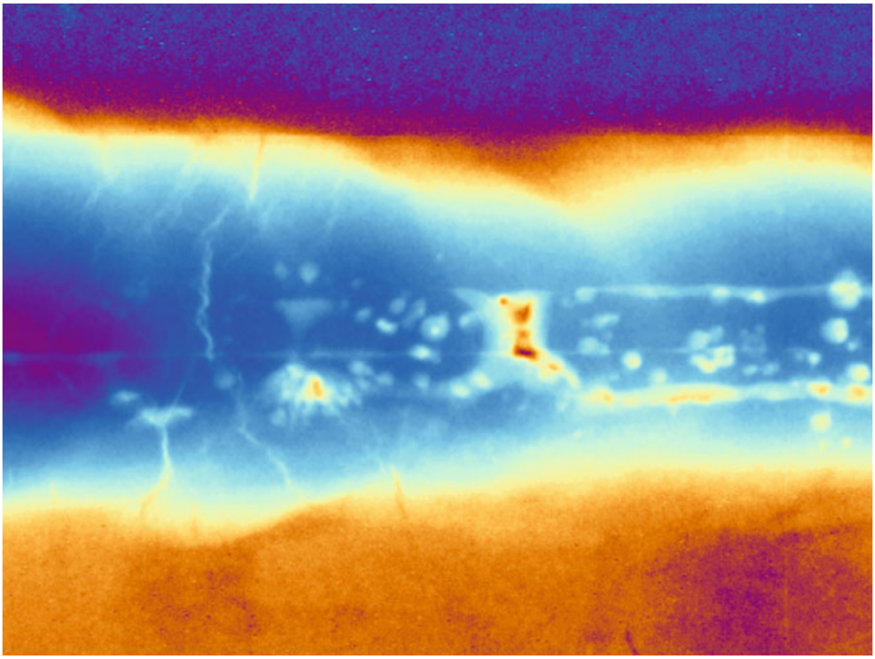
References

1. Castleman, K.: Digital Image Processing. Prentice-Hall, Englewood Cliffs (1996)
2. Boerner, H., Strecker, H.: Automated X-ray inspection of aluminum casting. IEEE Trans. Pattern Anal. Mach. Intell. **10**(1), 79–91 (1988)
3. MathWorks: Image Processing Toolbox for Use with MATLAB: User's Guide. The MathWorks Inc. (2014)
4. Heinrich, W.: Automated inspection of castings using X-ray testing. Ph.D. thesis, Institute for Measurement and Automation, Faculty of Electrical Engineering, Technical University of Berlin (1988). (in German)
5. Gonzalez, R., Woods, R.: Digital Image Processing, 3rd edn. Prentice Hall, Pearson (2008)
6. Faugeras, O.: Three-Dimensional Computer Vision: A Geometric Viewpoint. The MIT Press, Cambridge (1993)
7. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI–8**(6), 679–698 (1986)
8. Mery, D., Pedreschi, F.: Segmentation of colour food images using a robust algorithm. J. Food Eng. **66**(3), 353–360 (2004)
9. Haralick, R., Shapiro, L.: Computer and Robot Vision. Addison-Wesley Publishing Co., New York (1992)

10. Mery, D.: Crossing line profile: a new approach to detecting defects in aluminium castings. In: Proceedings of the Scandinavian Conference on Image Analysis (SCIA 2003), Lecture Notes in Computer Science vol. 2749, pp. 725–732 (2003)
11. Mery, D., Berti, M.A.: Automatic detection of welding defects using texture features. Insight-Non-Destr. Test. Cond. Monit. **45**(10), 676–681 (2003)
12. Mery, D., Filbert, D.: Automated flaw detection in aluminum castings based on the tracking of potential defects in a radioscopic image sequence. IEEE Trans. Robot. Autom. **18**(6), 890–901 (2002)
13. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide-baseline stereo from maximally stable extremal regions. Image Vis. Comput. **22**(10), 761–767 (2004)
14. Vedaldi, A., Fulkerson, B.: VLFeat: an open and portable library of computer vision algorithms. In: Proceedings of the International Conference on Multimedia, pp. 1469–1472. ACM (2010)
15. Mery, D., Filbert, D.: A fast non-iterative algorithm for the removal of blur caused by uniform linear motion in X-ray images. In: Proceedings of the 15th World Conference on Non-Destructive Testing (WCNDT-2000). Rome (2000)

Chapter 5
X-ray Image Representation

Abstract In this chapter we cover several topics that are used to represent an X-ray image (or a specific region of an X-ray image). This representation means that new features are extracted from the original image that can give us more information than the raw information expressed as a matrix of gray values. This kind of information is extracted as features or descriptors, i.e., a set of values, that can be used in pattern recognition problems such as object recognition, defect detection, etc. The chapter explains geometric and intensity features, and local descriptors and sparse representations that are very common in computer vision applications. Furthermore, the chapter addresses some feature selection techniques that can be used to chose which features are relevant in terms of extraction.



Cover image: *Welding defects (from X-ray image* `W0001_0001`, *well known as BAM5, colored with 'sinmap' colormap).*

D. Mery, *Computer Vision for X-Ray Testing*, DOI 10.1007/978-3-319-20747-6_5

5.1 Introduction

As we learned in the previous chapter, in image processing for X-ray testing, segmentation is used to detect (potential) regions that can be the objects of interest that we are looking for (see Sect. 4.5). As segmented potential regions frequently set off false detections, an analysis of the segmented regions can significantly improve the effectiveness of detection. Measuring certain characteristics of the segmented regions (*feature extraction*) can help us to distinguish the false detection, although some of the features extracted are either irrelevant or are not correlated. Therefore, a *feature selection* must be performed. Depending on the values returned for the selected features, we can try to classify each segmented potential region in one of the following two classes: *background* or *object of interest*.

In this chapter, we will explain several features that are normally used in image analysis and computer vision for X-ray testing. In our description, features will be divided into two groups: *geometric* and *intensity* features. Furthermore, we will cover some local descriptors and sparse representations that can be used in many X-ray testing applications. In this chapter we shall concentrate on the extraction and selection of features, whereas in the following chapter we will discuss the classification problem itself.

We will use Fig. 5.1 as our example in the description of features. In our example, we use an X-ray image of a circular defect. The segmentation is a binary image that gives information about the pixels that belongs to our object of interest (the defect). Geometric features are extracted from this binary image. Moreover, intensity features are extracted from the intensity image considering the pixels of the segmentation. Some intensity features consider only the gray values inside the segmented region, other ones take into account both gray values inside and outside the region (e.g., contrast).

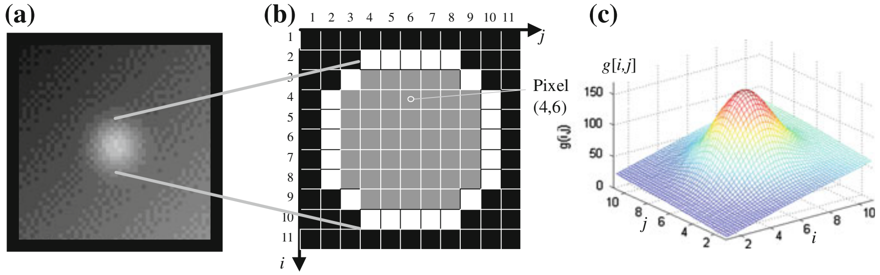


Fig. 5.1 Example of a region: **a** X-ray image, **b** segmented region (*gray* pixels), **c** 3D representation of the gray values

5.2 Geometric Features

These provide information on the location, size, and shape of the segmented region. Location and size features, such as center of mass, perimeter, height and width, are given in pixels. Shape features are usually coefficients without units. It is worth mentioning that we distinguish three different zones in the segmented image (see Fig. 5.1b), the segmented region (gray zone $\Re$), the boundary (white edge pixels ℓ) and the background (black zone).

5.2.1 Basic Geometric Features

In this section we will summarize basic geometric features that can be easily extracted.

Height and width
The eight and width of a region can be defined as:

$$h = i_{\max} - i_{\min} + 1 \qquad \text{and} \qquad w = j_{\max} - j_{\min} + 1 \tag{5.1}$$

where $i_{\max}$ and $i_{\min}$ is the maximal and minimal value that takes coordinate i in the region. The same is valid for $j_{\max}$ and $j_{\min}$ in j-direction. In our example of Fig. 5.1, $h = w = 7$ pixels.

Area and perimeter
We define the area A of a region as the number of pixels that belong to the region. On the other hand, the perimeter L is the number of pixels that belong to the boundary. In the region of Fig. 5.1, the area and the perimeter are $A = 45$ and $L = 24$ pixels, respectively. More accurate measurements for area and perimeter can also be estimated [1]: for instance, the boundary of the region can be fitted to a curve with known area and length (in our example the boundary can be fitted to a circle with radius $r = 4$ pixels, so $A = \pi r^2 = 50.26$ pixels and $L = 2\pi r = 25.13$ pixels), however, the computational time of such approaches can be extremely long if there are thousands of regions to be measured. Moreover, the shape of the region can be much more complex than a simple circle as shown in Fig. 4.30. We should remember, therefore, that the goal of feature extraction is not the accurate measurement, rather it is simply the extraction of features that can be used in a classification approach to separate our classes (objects of interest from background). Thus, it is not relevant that the measurement of the area of the region is just 45 pixels and not 50.26 pixels.

Center of mass
This provides information about the location of the region. It is computed as the average of coordinate i and coordinate j in pixels that belong to region $\Re$:

$$\bar{i} = \frac{1}{A}\sum_{i\in\Re} i \qquad \bar{j} = \frac{1}{A}\sum_{j\in\Re} j \tag{5.2}$$

where A is the area of the region, i.e., the number of pixels of the region.

Roundness

Shape features are usually attributed coefficients without units. An example is roundness that is defined as:

$$R = \frac{4\cdot A\cdot \pi}{L^2} \tag{5.3}$$

The roundness R is a value between 1 and 0. $R = 1$ means a circle, and $R = 0$ corresponds to a region without an area. In our example $R = 4\cdot 45\cdot \pi/24^2 = 0.98$.

Other basic features

There are some useful features that can be extracted employing the Image Processing Toolbox of Matlab using command `regionprops`. According to User's Guide of the Image Processing Toolbox [2] these are defined as follows:

- `EulerNumber`: The number of objects in the region minus the number of holes in those objects.
- `EquivDiameter`: The diameter of a circle with the same area as the region.
- `MajorAxisLength` and `MinorAxisLength`: The length (in pixels) of the major and minor axis of the ellipse that has the same normalized second central moments as the region.
- `Orientation`: The angle (in degrees ranging from -90 to $90°$) between the x-axis and the major axis of the ellipse that has the same second moments as the region.
- `Solidity`: The proportion of the pixels in the convex hull that are also in the region.
- `Extent`: The ratio of pixels in the region to pixels in the total bounding box.
- `Eccentricity`: The eccentricity of the ellipse that has the same second moments as the region.

All basic geometric features explained in this section can be extracted by command Xbasicgeo (see Appendix B) of $\mathbb{X}$vis Toolbox. An example is shown in Table 5.1, where the basic 15 geometric features (divided by 1000) are presented for 10 regions of Fig. 5.2: f_1: Center of mass in i direction. f_2: Center of mass in j direction. f_3: Height. f_4: Width. f_5: Area. f_6: Perimeter. f_7: Roundness. f_8: Euler Number. f_9: Equivalent Diameter. f_{10}: Major Axis Length. f_{11}: Minor Axis Length. f_{12}: Orientation. f_{13}: Solidity. f_{14}: Extent. f_{15}: Eccentricity.

Matlab Example 5.1 In this example, we show how to extract the basic geometric features of ten apples as segmented in Fig. 5.2.

Table 5.1 Basic geometric features of apples of Fig. 5.2 (→ Example 5.1)

k	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
1	0.3805	0.2954	0.1380	0.1520	17.2868	0.4795	0.0009	0.0010	0.1483	0.1574	0.1413	−0.0097	0.0010	0.0008	0.0004
2	0.3922	0.4499	0.1590	0.1530	18.6500	0.4940	0.0010	0.0010	0.1540	0.1624	0.1474	0.0696	0.0010	0.0008	0.0004
3	0.1571	0.4733	0.1490	0.1500	16.3525	0.4650	0.0010	0.0010	0.1442	0.1533	0.1371	−0.0512	0.0010	0.0007	0.0004
4	0.4659	0.5945	0.1430	0.1470	15.8229	0.4567	0.0010	0.0010	0.1418	0.1513	0.1343	−0.0445	0.0010	0.0008	0.0005
5	0.2433	0.6468	0.1480	0.1370	14.7376	0.4783	0.0008	0.0010	0.1369	0.1587	0.1207	−0.0619	0.0010	0.0007	0.0006
6	0.2210	1.4405	0.1670	0.1640	19.8205	0.5170	0.0009	0.0010	0.1588	0.1678	0.1528	−0.0326	0.0010	0.0007	0.0004
7	0.3853	1.4341	0.1510	0.1270	15.4304	0.4537	0.0009	0.0010	0.1401	0.1549	0.1277	−0.0836	0.0010	0.0008	0.0006
8	0.5513	1.4840	0.1740	0.1720	21.4611	0.5393	0.0009	0.0010	0.1652	0.1790	0.1536	−0.0415	0.0010	0.0007	0.0005
9	0.4136	1.5847	0.1500	0.1530	16.7374	0.4878	0.0009	0.0010	0.1459	0.1547	0.1416	0.0536	0.0010	0.0007	0.0004
10	0.2484	1.6098	0.1690	0.1580	19.5241	0.5112	0.0009	0.0010	0.1576	0.1673	0.1507	0.0734	0.0010	0.0007	0.0004

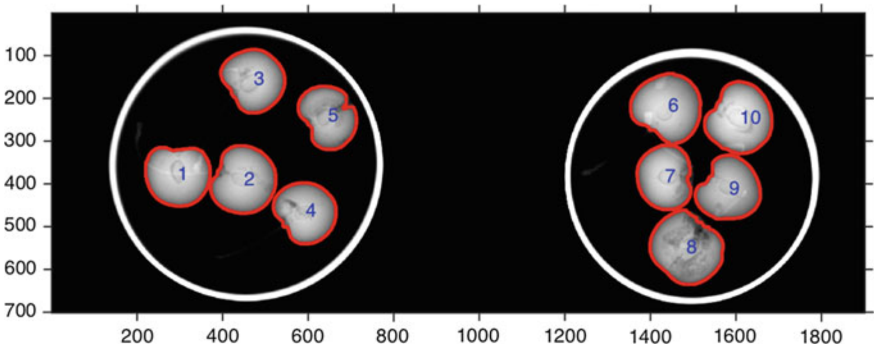


Fig. 5.2 X-ray image of 10 apples (→ Example 5.1)

Listing 5.1 : Basic geometric features

```
% AppleBasicGeoFeatures.m
I = Xloadimg('N',1,4)';                             % X–ray of apples
I = I(300:1000,100:2000);                           % input image
J = and(I>60,I<110);
K = imerode(bwfill(J,'holes'),ones(11,11));         % segmentation
[L,n]=bwlabel(K,4);                                 % regions

X = zeros(n,15);                                    % features
E = zeros(size(I));                                 % edges of the regions
imshow(I);hold on
for i=1:n
    Ri      = imdilate(L==i,ones(11,11));
    E       = or(E,bwperim(Ri));
    X(i,:)  = Xbasicgeo(Ri);                        % basic geo–features
    text(X(i,2),X(i,1),sprintf('%d',i),...
        'color','b','fontsize',12)                  % output
end
[ii,jj] = find(E==1); plot(jj,ii,'r.')              % display edges
X/1000                                              % features divided by 1000
```

The output of this code is shown in Fig. 5.2 and Table 5.1. The basic geometric features are extracted by command Xbasicgeo (see Appendix B) of 𝕏vis Toolbox. □

5.2.2 *Elliptical Features*

Elliptical features can be used to extract information about location, size, and shape of a region. They are extracted from a fitted ellipse to the boundary of the region [3]. From this ellipse we can extract the center, the length of the axes, the orientation and the eccentricity.

The pixels of the boundary are defined as (x_i, y_i) for $i = 1 \ldots L$. It is well known that an ellipse is defined as:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0, \tag{5.4}$$

that can be written as $\mathbf{a}^\mathsf{T}\mathbf{x} = 0$, where $\mathbf{a} = [a\ b\ c\ d\ e\ f]^\mathsf{T}$ is a vector that includes the parameters of the ellipse and $\mathbf{x} = [x^2\ xy\ y^2\ x\ y\ 1]^\mathsf{T}$ is a vector that includes the coordinates of a point (x, y) that lies on the ellipse.

If our region is elliptical, then for each point (x_i, y_i) we have $\mathbf{a}^\mathsf{T}\mathbf{x}_i = 0$ with $\mathbf{x}_i = [x_i^2\ x_iy_i\ y_i^2\ x_i\ y_i\ 1]^\mathsf{T}$. Nevertheless, in practice the regions are not perfectly elliptical, not only because real regions have different shapes, but also there is a discretization error when forming a digital image. For this reason, we look for a vector $\mathbf{a}$ so that $\mathbf{a}^\mathsf{T}\mathbf{x}_i \rightarrow \min$ for every point $i = 1 \ldots L$. That is, we can formulate the estimation of the parameters of the ellipse as an optimization problem as follows:

$$\| \mathbf{Xa} \| \rightarrow \min \tag{5.5}$$

where $\mathbf{X}$ is matrix with L rows whose ith row is $\mathbf{x}_i^\mathsf{T}$. Usually, a solution can be found by minimizing (5.5) subject to $\| \mathbf{a} \| = 1$. In this case, $\mathbf{a}$ is the last column of matrix $\mathbf{V}$, where $\mathbf{X} = \mathbf{USV}^\mathsf{T}$ is the singular value decomposition (SVD) of $\mathbf{X}$ [4].

The elliptical features can be extracted by writing (5.4) as follows:

$$\left(\frac{x - x_0}{a_e}\right)^2 + \left(\frac{y - y_0}{b_e}\right)^2 = 1 \tag{5.6}$$

where

$$a_e = \frac{1}{\sqrt{s\ a_p}}\ ,\ b_e = \frac{1}{\sqrt{s\ b_p}} \tag{5.7}$$

with

$$s = \frac{1}{v - f} \qquad v = \mathbf{t}^\mathsf{T}\mathbf{T}\mathbf{t}$$

$$\mathbf{T} = \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix} \qquad \mathbf{t} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \frac{1}{2}\mathbf{T}^{-1}\begin{bmatrix} d \\ e \end{bmatrix}$$

$$\begin{aligned} a_p &= a\cos^2(\alpha) + b\cos(\alpha)\sin(\alpha) + c\sin^2(\alpha) \\ b_p &= a\sin^2(\alpha) - b\cos(\alpha)\sin(\alpha) + c\cos^2(\alpha) \end{aligned}$$

and

$$\alpha = \frac{1}{2}\arctan\left(\frac{b}{a - c}\right) \tag{5.8}$$

The axes of the ellipse are defined by a_e and b_e, the center of the ellipse is located on (x_0, y_0) and the orientation is α. Thus, the eccentricity is defined by

$$e_x = \frac{\min(a_e, b_e)}{\max(a_e, b_e)} \tag{5.9}$$

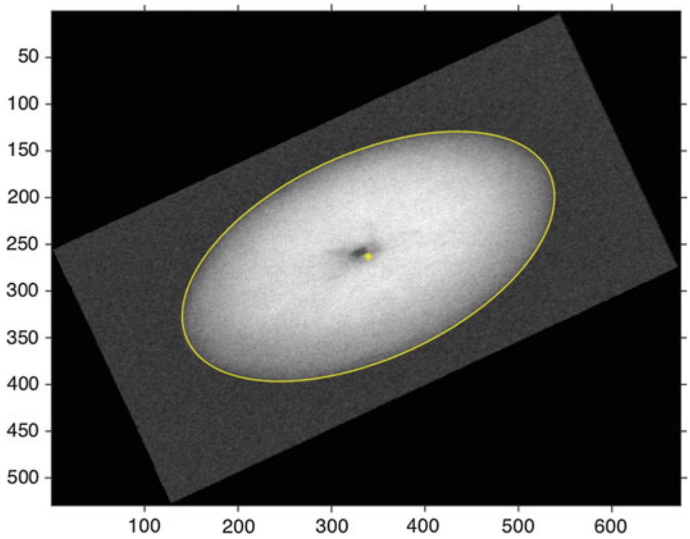


Fig. 5.3 Elliptical features of a fruit. In this example, the coordinates of the center of the ellipse correspond to the *yellow cross* ($i = 262.99$, $j = 339.42$). The estimated length of each axis are 109.71 and 213.39 pixels. The orientation (with respect to vertical axis in a counterclockwise direction) is 1.1396 rad, i.e., 65.30°. The eccentricity is 0.5141 (→ Example 5.2) (Color figure online)

For circular shapes, the eccentricity as the roundness (5.3), takes values between 0 and 1, where 1 means a perfect circle.

Matlab Example 5.2 In this example, we show how to extract elliptical features of a shape. We test this approach on an X-ray of an apple with a circular shape that was transformed and rotated as shown in Fig. 5.3.

Listing 5.2 : Elliptical boundary of a fruit

```
% EllipticalBoundary.m
I = double(Xloadimg('N',5,9));       % input image
I = imrotate(I(1:2:end,:),25);       % shape transformation and rotation
R = Xsegbimodal(I);                  % segmentation
[X,Xn] = Xfitellipse(R);             % ellipse features
Xprintfeatures(X,Xn)                 % features
imshow(I,[]); hold on
Xdrawellipse(X,'y')                  % ellipse drawing
plot(X(2),X(1),'y*')                 % center of ellipse
```

The output of this code is shown in Fig. 5.3. The elliptical features are extracted by command Xfitellipse (see Appendix B) of 𝕏vis Toolbox. □

5.2.3 Fourier Descriptors

Shape information—invariant to scale, orientation and position—can be measured using *Fourier descriptors* [5–7]. The coordinates of the pixels of the boundary are arranged as a complex number $i_k + j \cdot j_k$, with $j = \sqrt{-1}$ and $k = 0, \ldots, L-1$, where L is the perimeter of the region, and pixel k and $k+1$ are connected. The complex boundary function can be considered as a periodical signal of period L. The Discrete Fourier Transformation [8] gives a characterization of the shape of the region. The Fourier coefficients are defined by:

$$F_n = \sum_{k=0}^{L-1} (i_k + j \cdot j_k) e^{-j\frac{2\pi kn}{L}} \qquad \text{for } n = 0, \ldots, L-1. \tag{5.10}$$

The Fourier descriptors correspond to the coefficients F_n for $n > 0$. The Fourier coefficient F_0 is not used because it gives information about the location of the region. The magnitude and phase of Fourier descriptors give information about orientation and symmetry of the region. In general, only the magnitude $|F_n|$ is used. Fourier descriptors are invariant under rotation. The Fourier descriptors of our example in Fig. 5.1a are illustrated in Fig. 5.4. The first pixel of the periodic function is $(i_0, j_0) = (6, 10)$. In case the region is a perfect circle, $|F_n| = 0$ for $1 < n < L$ because (i_k, j_k) represent a perfect sinusoid. In our example, the region is not a perfect circle, however, as we can see the Fourier descriptors are very small for $2 < n < L$.

Fourier descriptors can be extracted using command Xfourierdes (see Appendix B) of 𝕏vis Toolbox.

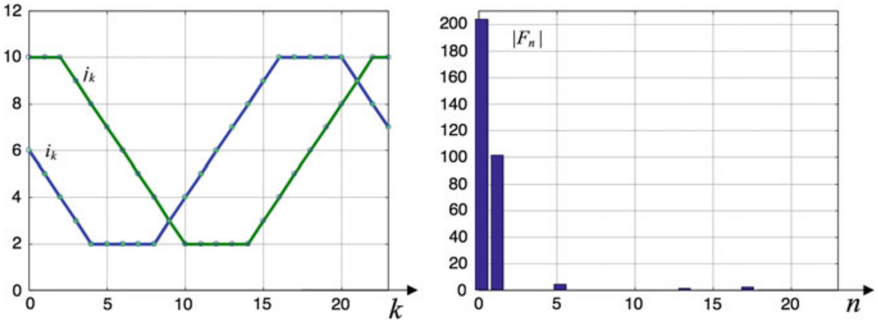


Fig. 5.4 Coordinates of the boundary of region of Fig. 5.1 and the Fourier descriptors

5.2.4 Invariant Moments

The statistical moments are defined by:

$$m_{rs} = \sum_{i,j\in\Re} i^r j^s \qquad \text{for } r, s \in \mathbb{N} \tag{5.11}$$

where $\Re$ is the set of pixels that belong to the region (see gray pixels in Fig. 5.1b). In this example, pixel $(i = 4, j = 6) \in \Re$. The parameter $r + s$ corresponds to the order of the moment. The reader can demonstrate that the zeroth moments m_{00} is equal to the area A of the region. Moreover, the center of mass of the region is easily defined by:

$$\bar{i} = \frac{m_{10}}{m_{00}} \qquad \bar{j} = \frac{m_{01}}{m_{00}} \tag{5.12}$$

The reader can compare this definition with (5.2). The coordinates of the center of mass can be computed using command Xcentroid (see Appendix B) of $\mathbb{X}$VIS Toolbox.

The center of mass and statistical moments of higher order, however, are not invariant to the location of the region. This can be useful for detecting objects that must be in certain locations. Nevertheless, when objects of interest may be everywhere in the image we must use features that are invariant to the position. Using the center of mass, the central moments are defined. They are invariant to the position:

$$\mu_{rs} = \sum_{i,j\in\Re} (i - \bar{i})^r (j - \bar{j})^s \qquad \text{for } r, s \in \mathbb{N}. \tag{5.13}$$

Other known moments that can be used are the well-known Hu moments [9, 10]. These were developed using the central moments as follows:

$$\begin{aligned}
\phi_1 &= \eta_{20} + \eta_{02}\\
\phi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2\\
\phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2\\
\phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2\\
\phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] +\\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]\\
\phi_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] +\\
&\quad 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\\
\phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] -\\
&\quad (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned} \tag{5.14}$$

with

$$\eta_{rs} = \frac{\mu_{rs}}{\mu_{00}^t} \qquad t = \frac{r + s}{2} + 1.$$

Hu moments are invariant to translation, rotation, and scale. That means that regions that have the same shape, but have a different size, location and orientation, will have similar Hu moments.

In addition, there are similar invariant features, called Gupta moments, that are derived from the pixels of the boundary (instead of the region) [11]. They are invariant to translation, rotation, and scale.

Sometimes, it is necessary to have features that are invariant to affine transformation as well (see Sect. 3.2.2). For this reason Flusser moments, i.e., features invariant to translation, rotation, scale, and affine transformation were derived from second and third order central moments [12, 13]:

$$
\begin{aligned}
I_1 &= \frac{\mu_{20}\mu_{02} - \mu_{11}^2}{\mu_{00}^4} \\
I_2 &= \frac{\mu_{30}^2\mu_{03}^2 - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} + 4\mu_{30}\mu_{12}^3 + 4\mu_{21}^3\mu_{03} - 3\mu_{21}^2\mu_{12}^2}{\mu_{00}^{10}} \\
I_3 &= \frac{\mu_{20}(\mu_{21}\mu_{03} - \mu_{12}^2) - \mu_{11}(\mu_{30}\mu_{03} - \mu_{21}\mu_{12}) + \mu_{02}(\mu_{30}\mu_{12} - \mu_{21}^2)}{\mu_{00}^7} \qquad (5.15) \\
I_4 &= (\mu_{20}^3\mu_{03}^2 - 6\mu_{20}^2\mu_{11}\mu_{12}\mu_{03} - 6\mu_{20}^2\mu_{02}\mu_{21}\mu_{03} + 9\mu_{20}^2\mu_{02}\mu_{12}^2 \\
&\quad +12\mu_{20}\mu_{11}^2\mu_{21}\mu_{03} + 6\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{03} - 18\mu_{20}\mu_{11}\mu_{02}\mu_{21}\mu_{12} \\
&\quad -8\mu_{11}^3\mu_{30}\mu_{03} - 6\mu_{20}\mu_{02}^2\mu_{30}\mu_{12} + 9\mu_{20}\mu_{02}^2\mu_{21} \\
&\quad +12\mu_{11}^2\mu_{02}\mu_{30}\mu_{12} - 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{21} + \mu_{02}^3\mu_{30}^2)/\mu_{00}^{11}
\end{aligned}
$$

Matlab Example 5.3 In this example, we show how to measure invariant moments that can be used as a shape feature of objects of interest. We tested this approach on an X-ray containing 10 apples. We superimpose onto this image 6 rectangles the size of which is $a \times b$ pixels (where $b = 2a$). The rectangles are located in horizontal and vertical directions as shown in Fig. 5.5. Thus, we can simulate an input X-ray image containing apples and rectangles. The idea is to separate them. We see that the first Hu moment can be used to effectively discriminate apples from rectangles.

Listing 5.3 : Detection using invariant moments

```
% AppleMoments.m
I = Xloadimg('N',1,4);                                    % X-ray of apples
I(  50: 249,1500:1599) = 90;
I(1500:1599,1550:1749) = 90;
I(  50: 449,1700:1899) = 90;
I(1800:1899,2050:2249) = 90;
I( 350: 749,2000:2199) = 90;
I(1050:1249,2000:2099) = 90;
```

```
imshow(I); hold on
J = and(I>60,I<110);
K = imerode(bwfill(J,'holes'),ones(11,11));                    % segmentation
[L,n]=bwlabel(K,4);                                            % regions
X = zeros(n,7);
for i=1:n
    Ri      = imdilate(L==i,ones(11,11));
    c       = Xcentroid(Ri);                                   % mass center
    X(i,:)  = Xhugeo(Ri);                                      % Hu moments
    text(c(2)-55,c(1),sprintf('%4.0f',X(i,1)*1000),...
        'color','b','fontsize',12)                             % output
end
E = bwperim(K);
[ii,jj] = find(E==1); plot(jj,ii,'y.')                         % display edges
```

The output of this code is shown in Fig. 5.5. In this example, command Xcentroid (see Appendix B) was used to compute the center of mass of each segmented object. The coordinates were used to print the first Hu moment (× 1000) in blue. The Hu moments are extracted by command Xhugeo (see Appendix B) of 𝕏vis Toolbox.

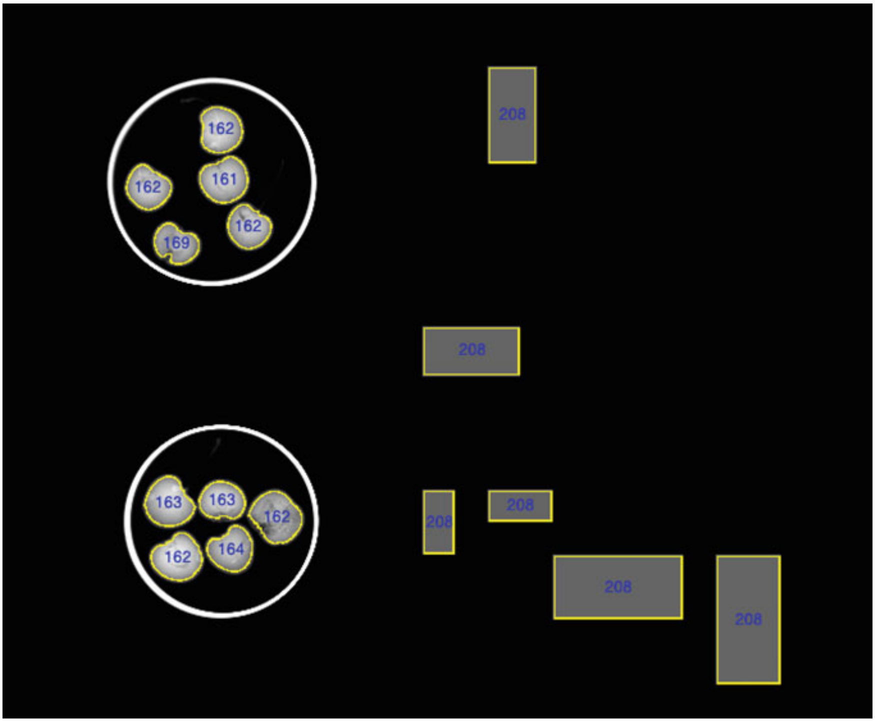


Fig. 5.5 First Hu moment (ϕ_1) of apples and *rectangles*. Since ϕ_1 for apples is approximately 163, and for these *rectangles* is 208, it is evident that this feature can be used to discriminate them from each other (→ Example 5.3)

The reader can test Flusser and Gupta moments using commands Xflusser and Xgupta (see Appendix B). □

5.3 Intensity Features

These provide information about the intensity of a region. For gray value images, e.g., X-ray images, there is only one intensity channel. The following features are computed using the gray values in the image, where $x(i, j)$ denotes the gray value of pixel (i, j).

5.3.1 Basic Intensity Features

In this section we summarize basic intensity features that can be easily extracted.

Mean gray value
The mean gray value of the region is computed as:

$$G = \frac{1}{A} \sum_{i,j \in \Re} x(i, j) \tag{5.16}$$

where $\Re$ is the set of pixels of the region and A the area. A 3D representation of the gray values of the region and its neighborhood of our example is shown in Fig. 5.1. In this example $G = 121.90$ ($G = 0$ means 100 % black and $G = 255$ corresponds to 100 % white).

Mean gradient in the boundary
This feature gives information about the change of the gray values in the boundary of the region. It is computed as:

$$C = \frac{1}{L} \sum_{i,j \in \ell} x'(i, j) \tag{5.17}$$

where $x'(i, j)$ means the gradient of the gray value function in pixel (i, j) (see Sect. 4.4.1) and ℓ the set of pixels that belong to the boundary of the region. The number of pixels of this set corresponds to L, the perimeter of the region. Using a Gaussian gradient operator in our example in Fig. 5.1, we obtain $C = 35.47$.

Mean second derivative
This feature is computed as:

$$D = \frac{1}{A} \sum_{i,j \in \Re} x''(i, j) \tag{5.18}$$

where $x''(i, j)$ denotes the second derivate of the gray value function in pixel (i, j). The Laplacian-of-Gauss (LoG) operator can be used to calculate the second derivate of the image. If $D > 0$ we have a region that is darker than its neighborhood as shown in Fig. 4.18.

Other basic features
A simple texture feature is the local variance [14]. This is given by:

$$\sigma_g^2 = \frac{1}{4hw + 2h + 2w} \sum_{i=1}^{2h+1} \sum_{j=1}^{2w+1} (g(i, j) - \bar{g})^2 \tag{5.19}$$

where $\bar{g}$ denotes the mean gray value in the zone, and h and w are the height and width as expressed in (5.1).

Other basic intensity features such as kurtosis and skewness can be computed as (5.19). All intensity geometric features explained in this section can be extracted by command Xbasicint (see Appendix B) of 𝕏vis Toolbox. An example is shown in Table 5.2, where the basic 6 intensity features are presented for 10 regions of Fig. 5.2: f_1: Intensity mean. f_2: Intensity standard deviation. f_3: Intensity kurtosis. f_4: Intensity skewness. f_5: Mean Laplacian. f_6: Mean boundary gradient.

Table 5.2 Basic intensity features of apples of Fig. 5.2 (→ Example 5.4)

k	f_1	f_2	f_3	f_4	f_5	f_6
1	158.6886	41.5743	2.2734	−0.5948	−0.3466	11.7224
2	151.1139	39.7086	2.1362	−0.4942	−0.3008	10.4349
3	150.1203	39.7638	2.1516	−0.5286	−0.3117	10.0269
4	148.1976	39.3551	2.0933	−0.4574	−0.3213	10.2263
5	136.1336	34.4244	2.1840	−0.4309	−0.3156	9.0255
6	156.5436	43.7055	2.0797	−0.3863	−0.3287	10.3313
7	152.4726	44.2841	1.9045	−0.3172	−0.3046	9.0481
8	132.3472	33.1643	2.5448	−0.5122	−0.2555	8.8948
9	143.1046	38.3128	2.0296	−0.3942	−0.2908	9.0821
10	166.2491	46.3513	2.2428	−0.5072	−0.3114	11.1117

Matlab Example 5.4 In this example, we show how to extract basic intensity features of ten apples as segmented in Fig. 5.2.

Listing 5.4 : Basic intensity features

```
% AppleBasicIntFeatures.m
close all
I = Xloadimg('N',1,4)';                             % X–ray of apples
I = I(300:1000,100:2000);                           % input image
J = and(I>60,I<110);
K = imerode(bwfill(J,'holes'),ones(11,11));         % segmentation
[L,n]=bwlabel(K,4);                                 % regions

X = zeros(n,6);                                     % features
E = zeros(size(I));                                 % edges of the regions
imshow(I);hold on
op.mask = 15; op.show = 0;
for i=1:n
    Ri      = imdilate(L==i,ones(11,11));
    E       = or(E,bwperim(Ri));
    X(i,:)  = Xbasicint(I,Ri,op);                   % basic int–features
    c       = Xcentroid(Ri);                        % centroid
    text(c(2),c(1),sprintf('%d',i),...
        'color','b','fontsize',12)                  % output
end
[ii,jj] = find(E==1); plot(jj,ii,'r.')              % display edges
X                                                   % features
```

The output of this code is shown in Fig. 5.2 and Table 5.2. The basic geometric features are extracted by command Xbasicint (see Appendix B) of 𝕏vis Toolbox. □

5.3.2 Contrast

The contrast gives a measure of the difference in the gray value between region and its neighborhood. The smaller the gray value difference, the smaller the contrast. In this work, region and neighborhood define a zone. The zone is considered as a window of the image:

$$g(i,j) = x(i + i_r, j + j_r) \tag{5.20}$$

for $i = 1, \ldots, 2h + 1$ and $j = 1, \ldots, 2w + 1$, where h and w are the height and width as expressed in (5.1). The offsets i_r and j_r are defined as $i_r = \bar{i} - h - 1$ y $j_r = \bar{j} - b - 1$, where $(\bar{i}, \bar{j})$ denotes the center of mass of the region as computed in (5.12).

Contrast is a very important feature in fault detection, as the differences in the gray values are good for distinguishing a region from its neighborhood. The smaller the gray value difference, the smaller the contrast. In order to visualize the contrast we can use a 3D representation with three coordinates (x, y, z), where (x, y) are used to represent the location of a pixel (i, j), and z is used for the representation of the gray value. An example is illustrated in Fig. 5.1c that shows the 3D representation of Fig. 5.1a. The reader can observe in this example a high contrast region.

There are many definitions of contrast. A common definition of contrast is given using texture features (as explained in Sect. 5.3.5). Other simple definitions of contrast are given in [12, 15]:

$$K_1 = \frac{G - G_e}{G_e}, \qquad K_2 = \frac{G - G_e}{G + G_e} \qquad \text{y} \qquad K_3 = \ln(G/G_e), \tag{5.21}$$

where G an G_e denote the mean gray value in the region and in the neighborhood respectively.

Two further definitions of contrast are given in [16] where new contrast features are suggested. According to Fig. 5.6 these new features can be calculated in four steps: (i) we take a profile in i direction and in j direction centered in the mass center of the region (see P_1 and P_2 respectively); (ii) we calculate the ramps R_1 and R_2 that are estimated as a first-order function that contains the first and last point of P_1 and P_2; (iii) new profiles without background are computed as $Q_1 = P_1 - R_1$ and $Q_2 = P_2 - R_2$ (they are stored together as $Q = [Q_1\ Q_2]$); (iv) the new contrast features are given by:

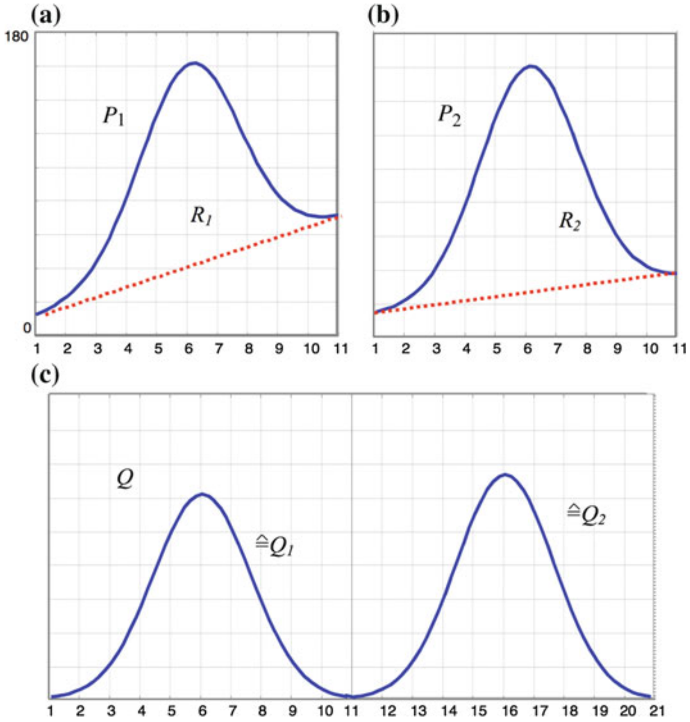


Fig. 5.6 Computation of Q for contrast features for region of Fig. 5.1: **a** Profile in i direction, **b** profile in j direction, **c** fusion of profiles: $Q = [Q_1\ Q_2]$

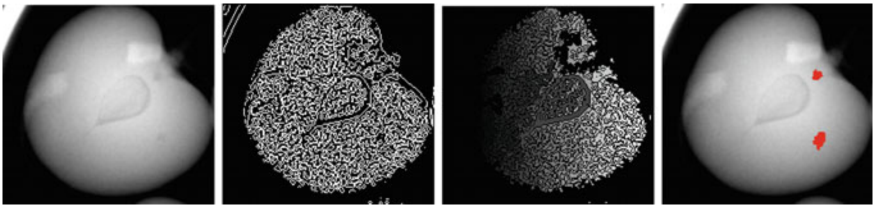


Fig. 5.7 Detection of small defects in apples using area contrast features: input image, edge detection, labeled regions and detection (→ Example 5.5)

$$K_\sigma = \sigma_Q \qquad \text{and} \qquad K = \ln(Q_{max} - Q_{min}). \tag{5.22}$$

Another definition of contrast can be found in [17], where the contrast is given by the mean of absolute differences between pixel values and mean of adjacent (e.g., 8-adjacent pixels):

$$K_c = \frac{1}{A_T} \sum_{(i,j)\in\mathbb{T}} |g(i,j) - \mu_{A(i,j)}|. \tag{5.23}$$

where A_T is the area of the region and its neighborhood and $\mu_{A(i,j)}$ is the mean value of pixels locations adjacent of pixel (i, j).

The contrast features explained in this section can be computed using command Xcontrast (see Appendix B) of 𝕏vis Toolbox.

Matlab Example 5.5 In this example, we show how to detect small defects in an X-ray image of an apple (see Fig. 5.7) using area and contrast features. We follow the general block diagram of Fig. 4.30. Here, area and contrast features are extracted for each region as defined by enclosed edges. The detection is performed if the size of the region is between some thresholds and the contrast is high enough.

Listing 5.5 : Defects detections using area and contrast features

```
% ContrastDefects.m
close all
I            = Xloadimg('N',1,4);               % X−ray of apples
X            = I(1450:1629,420:609);            % input image (one apple)
figure(1)
imshow(X); title('input image');

E            = and(edge(X,'log',1e−10,1),X>40);  % edge detection

[F,m]        = bwlabel(not(E),4);               % labels of the regions
op.neighbor  = 2;                               % neigborhood is imdilate
op.param     = 5;                               % with 5x5 mask
op.show      = 0;
R            = zeros(size(X));                  % initialization of detection
for i=1:m                                       % for each region
    Ri     = F==i;                              % region i
    Area   = sum(Ri(:));
    K      = Xcontrast(X,Ri,op);                % contrast features
    if (Area>50) && (Area<150) && ...
```

```
        K(2)<-0.01 && K(5) > 2.9                 % detection
        R = or(R,Ri);
    end
end
figure(2)
Xbinview(X,imclose(R,ones(5,5)));                % output image
title('small defects');
```

The output of this code is shown in Fig. 5.7. In this example, the contrast features are extracted using command Xcontrast (see Appendix B) of 𝕏vis Toolbox. In this example we use features K_2 from (5.21) and K from (5.22). □

5.3.3 Crossing Line Profiles

An approach based on *crossing line profiles* (CLP) was originally developed to detect aluminum casting defects [18], however, it can be used to detect spots in general, or regions that have some gray value difference with their neighborhood. As the contrast between a defect and a defect-free neighborhood is distinctive, the detection is usually performed by thresholding this feature (as we already learned in Sect. 5.3.2). Nevertheless, this measurement suffers from accuracy error when the neighborhood is not homogeneous, for example when a defect is at an edge of a regular structure of the test object (see Fig. 4.31). For this reason, many approaches use a priori information about the location of regular structures of the test piece. CLP is able to detect those defects without a priori knowledge using *crossing line profiles*, i.e., the gray level profiles along straight lines crossing each segmented potential region in the middle. The profile that contains the most similar gray levels in the extremes is selected. Hence, the homogeneity of the neighborhood is ensured. Features from the selected profile are extracted.

In this approach, we follow a simple automated segmentation approach based on Figs. 4.30 and 4.31. The steps of detection based on CLP are shown in Fig. 5.8. First, a Laplacian-of-Gaussian (LoG) kernel and a zero crossing algorithm are used to detect the edges of the X-ray images. The LoG operator involves a Gaussian low-pass filter which is a good choice for the pre-smoothing of our noisy images that are obtained without frame averaging. The resulting binary edge image should produce at real defects closed and connected contours which demarcate *regions*. However, a region of interest may not be perfectly enclosed if it is located at an edge of a regular structure as shown in Fig. 5.8c. In order to complete the remaining edges of these defects, a thickening of the edges of the regular structure is performed as follows: (a) the gradient of the original image is calculated (see Fig. 5.8d); (b) by thresholding the gradient image at a high gray level a new binary image is obtained; and (c) the resulting image is added to the zero-crossing image (see Fig. 5.8e). Afterwards, each closed region is segmented as a potential flaw. For details see a description of the method in [19].

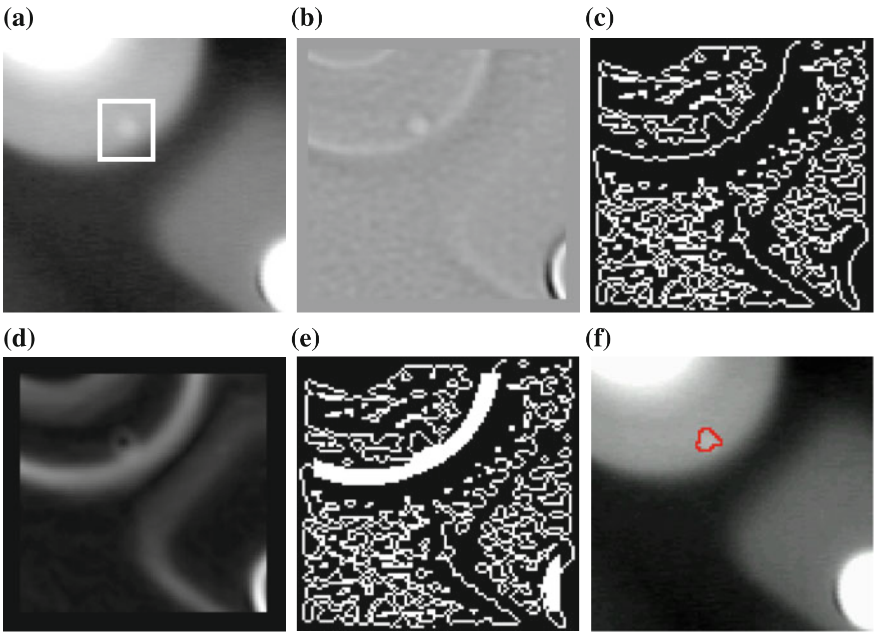


Fig. 5.8 Detection of flaws: **a** radioscopic image with a small flaw at an edge of a regular structure, **b** Laplacian-filtered image with σ = 1.25 pixels (kernel size = 11 × 11), **c** zero crossing image, **d** gradient image, **e** edge detection after adding high gradient pixels, and **f** detected flaw using feature F_1 extracted from a crossing line profile (→ Example 5.6)

This is a very simple detector of potential regions with a large number of false detections flagged erroneously. However, the advantages are as follows: (i) it is a single detector (it is the same detector for each image), (ii) it is able to identify potential defects independent of the placement and the structure of the specimen, i.e., without a priori information of the design structure of the test piece, and (iii) the detection rate of real flaws is very high (approximately 90 %). In order to reduce the number of the false positives, the segmented regions must be measured and classified.

A segmented potential region is defined as a region enclosed by edges of the binary image obtained in the edge detection (see connected black pixels in Fig. 5.8e). For each segmented region, a window g is defined from the X-ray image x as: $g(i, j) = x(i + i_r, j + j_r)$ for $i = 1 \ldots 2h + 1$, and $j = 1 \ldots 2w + 1$, where h and w are the height and width of the region as defined in (5.1). The offsets i_r and j_r are defined as $i_r = \bar{i} - h - 1$ and $j_r = \bar{j} - w - 1$ where $(\bar{i}, \bar{j})$ denotes the coordinates of the center of mass of the region (5.12), rounded to the nearest integers. Hence, g is a window of size $(2h + 1) \times (2w + 1)$, in which the middle pixel corresponds to the center of mass of the segmented potential flaw, i.e., $g(h + 1, w + 1) = x(\bar{i}, \bar{j})$.

Now, we define the crossing line profile P_θ as the gray level function along a straight line of window g through the middle pixel $(h+1, w+1)$ forming an angle θ with i-axis. In Sect. 5.3.2, P_0 and $P_{\pi/2}$ were analyzed together in order to obtain two

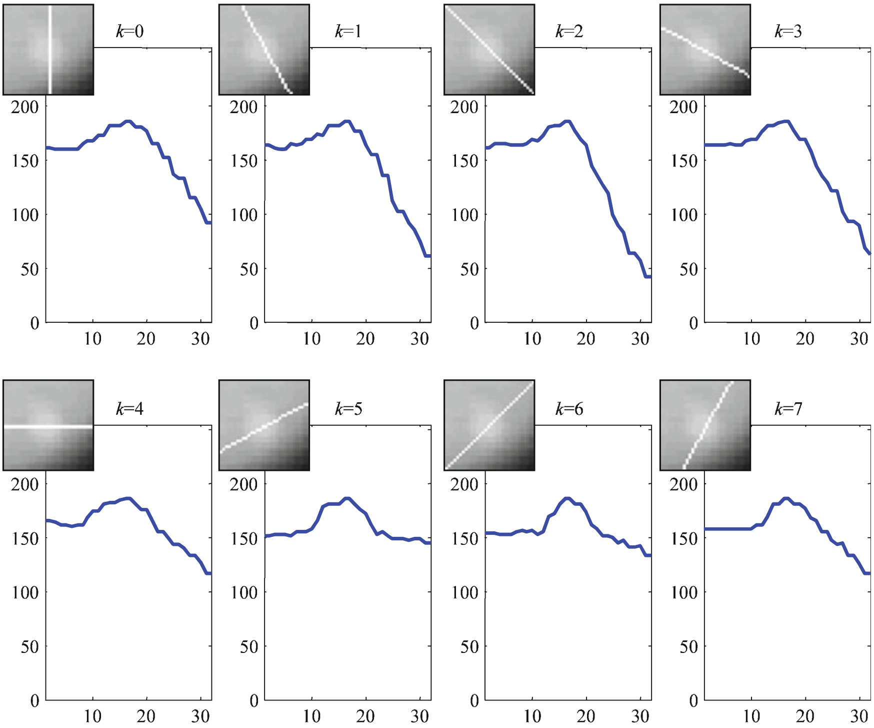


Fig. 5.9 Crossing line profiles for the window shown in Fig. 5.8a (→ Example 5.6)

features, K and K_σ, that give a measurement of the difference between maximum and minimum, and the standard deviation of both crossing line profiles. However, the analysis does not take into account that the profiles could include a nonhomogeneous area. For example, if a non-defect region is segmented at an edge of a regular structure, it could be that P_0 (or $P_{\pi/2}$) includes a significant gray level change of the regular structure. In this case, the variation of the profile will be large and therefore the region will be erroneously classified as defect.

In order to avoid this problem, we suggest an individual analysis of eight crossing line profiles P_θ, at $\theta = k\pi/8$, for $k = 0, \ldots, 7$, as illustrated in Fig. 5.9. In this analysis, the crossing line profile that contains the most similar gray levels in the extremes is selected. Hence, the attempt is made to ensure the homogeneity of the neighborhood filtering out those profiles that present a high gray level change in the edge of the regular structure. In the example of Fig. 5.9, the selected profile is obtained for $k = 5$ where the gray values of the extremes are both approximately equal to 150. We can observe that the selected crossing line is approximately perpendicular to the direction of the gradient of the X-ray image without defect. This coincides with one of the criteria used by approaches with a priori knowledge: the selected pixels of the defect-free area are located perpendicular to the direction of the gradient of the piece's contour [20].

Before the features are extracted, a preprocessing of the selected crossing line profile is performed as follows: (1) The selected profile is resized to size $n = 32$ using a nearest neighbor interpolation. The resized profile will be denoted by P. (2) In order to obtain a defect profile without the background of the regular structure, P is linearly transformed by $Q_i = mP_i + b$, for $i = 1, \ldots, n$, where m and b are so chosen that $Q_1 = Q_n = 0$.

Finally, the proposed features are extracted from the normalized profile Q. They are defined as follows:

$$\begin{aligned}
\bar{Q} &= \text{mean}(Q) \\
\sigma_Q &= \text{std}(Q) \\
\Delta_Q &= \max(Q) - \min(Q) \\
F_i &= \textstyle\sum_{k=0}^{n-1} Q_{k+1} e^{-j\frac{2\pi ki}{n}} \qquad \text{for } i = 1, \ldots, 4.
\end{aligned} \tag{5.24}$$

That is $\bar{Q}$: mean of Q; σ_Q: standard deviation of Q; Δ_Q: difference between maximum and minimum of Q; and F_i: magnitude of the ith harmonic of the Discrete Fourier Transform of Q for $i = 1, \ldots 4$.

Matlab Example 5.6 In this example, we show how to detect a very small casting defect that is located at the edge of a regular structure as illustrated in Fig. 5.8 using area and CLP features. We follow the general block-diagram of Fig. 4.30. That is area and contrast features are extracted for each region defined by enclosed edges. The detection is performed if the size of the region is between some thresholds and a CLP feature is high enough.

Listing 5.6 : Defects detections using area and CLP features

```
% DetectionCLP.m
close all
X = imread('small_wheel.png');
[N,M] = size(X);
figure(1);
imshow(X); title('Input image');                % input image

D = Xgradlog(X,1.25,4);                         % edge detection
figure(2)
imshow(D,[]);title('or(LoG,High gradient)')

[F,m]   = bwlabel(not(D),4);                    % labels of the regions
op.ng   = 32;                                   % size of CLP window
op.show = 0;                                    % do not display results
R           = zeros(N,M);                       % initialization of detection
for i=1:m                                       % for each region
    Ri    = F==i;                               % region i
    Area  = sum(Ri(:));                         % area of region
    CLP   = Xclp(X,Ri,op);                      % CLP features
    if (Area>10) && (Area<40) && CLP(6)>0.8     % detection
        R       = or(R,Ri);
        op.show = 1;
        CLP     = Xclp(X,Ri,op);                % display results
        op.show = 0;
```

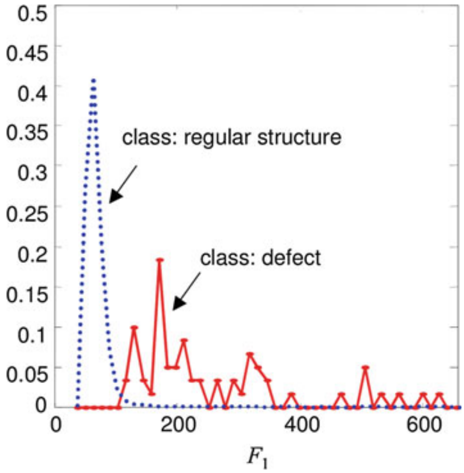


Fig. 5.10 Class distribution of CLP feature F_1 in detection of casting defects

```
        pause(1)
    end
end
figure(3)
Xbinview(X,imclose(R,ones(5,5)));                 % output image
title('Casting defects');
```

The output of this code is shown in Figs. 5.8 and 5.9. In this example, the edges are detected using command Xgradlog (see Appendix B) of 𝕏vis Toolbox, that computes the logical OR of edge detection using LoG and edge detection by thresholding the gradient. The contrast features are extracted using command Xclp (see Appendix B) of 𝕏vis Toolbox. In this example we use features F_1 from (5.24). □

CLP features were tested on detecting casting defects. In this experiment, 50 X-ray images of aluminum wheels were analyzed. In the segmentation, approximately 23,000 potential flaws were obtained, in which there were 60 real defects. Some of these were existing blow holes. The other defects were produced by drilling small holes in positions of the casting which were known to be difficult to detect. In the performance analysis, the best result was achieved by our feature F_1 (5.24). The class distribution between class ‘defect’ and ‘non-defect’ (or regular structure) is illustrated in Fig. 5.10. The reader can observe the effectiveness of the separation clearly. For more details see [18].

5.3.4 Intensity Moments

In intensity moments, we use statistical moments (5.11) including gray value information [12]:

$$m'_{rs} = \sum_{i,j \in \Re} i^r j^s x(i,j) \qquad \text{for } r, s \in \mathbb{N}. \tag{5.25}$$

The summation is computed over the pixels (i, j) of the region $\Re$ only. Thus, it is possible to compute Hu, Flusser and Gupta moments, as explained in Sect. 5.2.4 using the gray value information of the region. Hu moments with intensity information can be computed by Xhuint (see Appendix B) of $\mathbb{X}$vis Toolbox.

5.3.5 Statistical Textures

These features provide information about the distribution of the gray values in the image. In this work, however, we restrict the computation of the texture features for a zone only defined as region and neighborhood (see Eq. 5.20).

Statistical texture features can be computed using the co-occurrence matrix $\mathbf{P}_{kl}$ [21]. The element $P_{kl}(i, j)$ of this matrix for a zone is the number of times, divided by N_T, that gray levels i and j occur in two pixels separated by that distance and direction given by the vector (k, l), where N_T is the number of pixels pairs contributing to build matrix $\mathbf{P}_{kl}$. In order to decrease the size $N_x \times N_x$ of the co-occurrence matrix the gray scale is often reduced to 8 gray levels. From the co-occurrence matrix several texture features can be computed. Haralick in [21] proposes (here $p(i, j) := P_{kl}(i, j)$):

$$
\begin{array}{ll}
\text{Angular second moment:} & f_1 = \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} [p(i, j)]^2 \\
\text{Contrast:} & f_2 = \sum_{n=0}^{N_x-1} n^2 \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} p(i, j) \\
 & \quad \text{for } |i - j| = n \\
\text{Correlation:} & f_3 = \frac{1}{\sigma_x \sigma_y} \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} \\
 & \quad \left[ij \cdot p(i, j) - \mu_x \mu_y \right]^2 \\
\text{Sum of squares:} & f_4 = \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} (i - j)^2 p(i, j) \\
\text{Inverse difference moment:} & f_5 = \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} \frac{p(i,j)}{1+(i-j)^2} \\
\text{Sum average:} & f_6 = \sum_{i=2}^{2N_x} i \cdot p_{x+y}(i) \\
\text{Sum variance:} & f_7 = \sum_{i=2}^{2N_x} (i - f_8) \cdot p_{x+y}(i) \\
\text{Sum entropy:} & f_8 = -\sum_{i=2}^{2N_x} p_{x+y}(i) \cdot \log(p_{x+y}(i)) \\
\text{Entropy:} & f_9 = -\sum_{i=1}^{N_x} \sum_{j=1}^{N_x} p(i, j) \log(p(i, j)) \\
\text{Difference variance:} & f_{10} = \text{var}(\mathbf{p}_{x+y}) \\
\text{Difference entropy:} & f_{11} = -\sum_{i=0}^{N_x-1} p_{x-y}(i) \cdot \log(p_{x-y}(i)) \\
\text{Information measures of correlation 1:} & f_{12} = \frac{f_9 - HXY1}{\max\,(HX, HY)} \\
\text{Information measures of correlation 2:} & f_{13} = \sqrt{1 - \exp\left(-2(HXY2 - HXY)\right)} \\
\text{Maximal correlation coefficient:} & f_{14} = \sqrt{\lambda_2}
\end{array}
\tag{5.26}
$$

where μ_x, μ_y, σ_x and σ_y are the means and standard deviations of p_x and p_y respectively with

$$\begin{aligned}
p_x &= \textstyle\sum_{j=1}^{N_x} p(i, j)\\
p_y &= \textstyle\sum_{i=1}^{N_x} p(i, j)\\
p_{x+y}(k) &= \textstyle\sum_{i=1}^{N_x} \sum_{j=1 i+j=k}^{N_x} p(i, j) \ \text{ for } k = 2, 3, \dots 2N_x\\
p_{x-y}(k) &= \textstyle\sum_{i=1}^{N_x} \sum_{j=1 |i-j|=k}^{N_x} p(i, j) \ \text{ for } k = 0, 1, \dots N_x - 1,
\end{aligned}$$

and

$$\begin{aligned}
HX &= -\textstyle\sum_{i=1}^{N_x} p_x(i) \log(p_x(i))\\
HY &= -\textstyle\sum_{j=1}^{N_x} p_y(j) \log(p_y(j))\\
HXY1 &= -\textstyle\sum_{i=1}^{N_x} \sum_{j=1}^{N_x} p(i, j) \log(p_x(i) p_y(j))\\
HXY2 &= -\textstyle\sum_{i=1}^{N_x} \sum_{j=1}^{N_x} p_x(i) p_y(j) \log(p_x(i) p_y(j)).
\end{aligned}$$

In f_{14}, λ_2 is the second largest eigenvalue of Q defined by

$$Q(i, j) = \textstyle\sum_{k=1}^{N_x} \frac{p(i,k)p(j,k)}{p_x(i)p_y(k)}$$

The texture features are extracted for four directions (0°–180°, 45°–225°, 90°–270° and 135°–315°) in different distances $d = \max(k, l)$. That is, for a given distance d we have four possible co-occurrence matrices: P_{0d}, P_{dd}, P_{d0} and P_{-dd}. For example, for $d = 1$, we have (k, l) = (0, 1); (1, 1); (1, 0); and (−1, 1). After Haralick, 14 texture features using each co-occurrence matrix are computed (5.26), and the mean and range for each feature are calculated, i.e., we obtain 14 × 2 = 28 texture features for each distance d. The features will be denoted as $\bar{f_i}$ for the mean and f_i^Δ for the range, for $i = 1 \dots 14$.

The texture features after Haralick can be computed using command Xharalick (see Appendix B) of $\mathbb{X}$VIS Toolbox.

5.3.6 Gabor

The Gabor functions are Gaussian shaped band-pass filters, with dyadic treatment of the radial spatial frequency range and multiple orientations, which represent an appropriate choice for tasks requiring simultaneous measurement in both space and frequency domains. The Gabor functions are a complete (but a nonorthogonal) basis set given by:

$$f(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right) \tag{5.27}$$

where σ_x and σ_y denote the Gaussian envelope along the x and y-axes, and u_0 defines the radial frequency of the Gabor function. Examples of Gabor functions

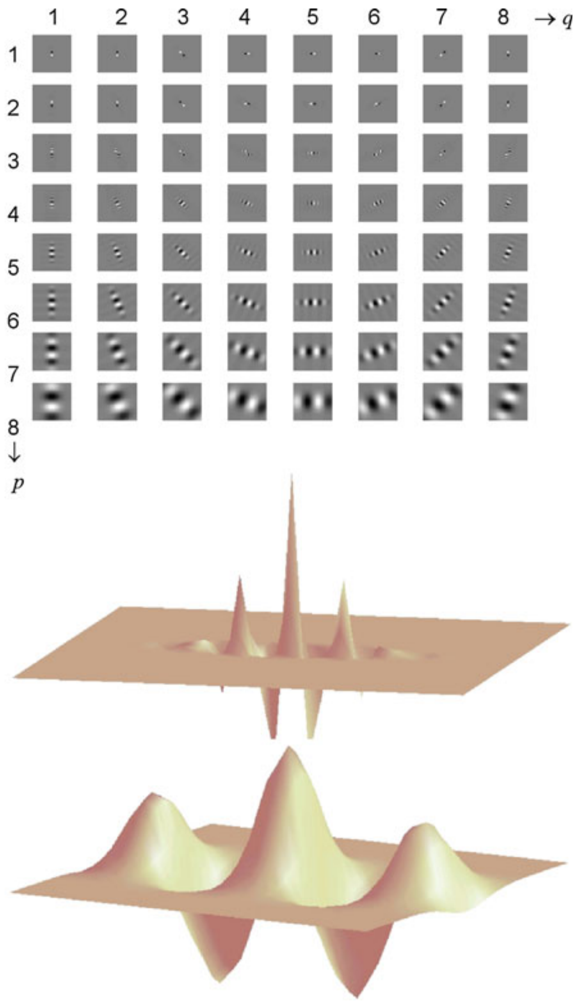


Fig. 5.11 Example of Gabor functions in spatial domain: *top* imaginary components of self-similar filter bank by using $p = 1 \ldots 8$ scales and $q = 1 \ldots 8$ orientations, *bottom* 3D representations of two of them

are illustrated in Fig. 5.11. In this case a class of self-similar functions are generated by rotation and dilation of $f(x, y)$.

Each Gabor filter has a real and an imaginary component that are stored in $M \times M$ masks, called $\mathbf{R}_{pq}$ and $\mathbf{I}_{pq}$ respectively, where $p = 1 \ldots S$, denotes the scale, and $q = 1 \ldots L$, denotes the orientation (for details see [22]). Usually, $S = 8$ scales, and $L = 8$ orientations as shown in Fig. 5.11, with $M = 27$.

The Gabor filters are applied to each segmented window $\mathbf{W}$, that contains the segmented region and its surrounding (see Fig. 5.1). The filtered windows $\mathbf{G}_{pq}$ are computed using the 2D convolution (4.9) of the window $\mathbf{W}$ of the X-ray image with the Gabor masks as follows:

$$\mathbf{G}_{pq} = \left[(\mathbf{W} * \mathbf{R}_{pq})^2 + (\mathbf{W} * \mathbf{I}_{pq})^2\right]^{1/2} \tag{5.28}$$

The Gabor features, denoted by g_{pq}, are defined as the average output of $\mathbf{G}_{pq}$, i.e., it yields $S \times L$ Gabor features for each segmented window:

$$g_{pq} = \frac{1}{n_w n_w} \sum_{i=1}^{n_w} \sum_{j=1}^{m_w} G_{pq}(i, j) \tag{5.29}$$

where the size of the filtered windows $\mathbf{G}_{pq}$ is $n_w \times m_w$.

Three additional Gabor features can be extracted: (i) maximum of all Gabor features: $g_{\max} = \max(\mathbf{g})$, (ii) minimu of all Gabor features: $g_{\min} = \min(\mathbf{g})$, and (iii) range of all Gabor features: $g_{\Delta} = g_{\max} - g_{\min}$. These features are very useful because they are rotation invariant.

The Gabor features can be computed using command Xgabor (see Appendix B) of $\mathbb{X}$VIS Toolbox.

5.3.7 Filter Banks

Filter banks can be used to extract texture information [23]. They are used in image transformations like Discrete Fourier Transform (DFT) (magnitude and phase), Discrete Cosine Transform (DCT) [24], and Wavelets as Gabor features based on 2D Gabor functions (see Sect. 5.3.6).

For an image $\mathbf{X}$ of $N \times N$ pixels, the Discrete Fourier Transformation in 2D is defined as follows:

$$F(m, n) = \sum_{i=1}^{N} \sum_{k=1}^{N} X(i, k) e^{-2\pi j \left(\frac{(m-1)(i-1)}{N} + \frac{(n-1)(k-1)}{N} \right)} \tag{5.30}$$

where $j = \sqrt{-1}$. $F(m, n)$ is a complex number. That means magnitude and phase can be used as features. Fourier features can be computed using command Xfourier (see Appendix B) of $\mathbb{X}$VIS Toolbox.

Discrete Cosine Transform in 2D is defined as:

$$D(m, n) = \alpha_m \alpha_n \sum_{i=1}^{N} \sum_{k=1}^{N} X(i, k) \cos\left(\frac{\pi(2i-1)(m-1)}{2N} \right) \cos\left(\frac{\pi(2k-1)(n-1)}{2N} \right) \tag{5.31}$$

where $\alpha_1 = 1/\sqrt{N}$ and $\alpha_m = \sqrt{2/N}$, for $m = 2 \ldots N$. DCT features are real numbers instead of complex number such as Fourier features. DCT features can be computed using command Xdct (see Appendix B) of $\mathbb{X}$VIS Toolbox.

It is worth mentioning that these features are not rotation invariant, however, we can extract rotation-invariant features if we use maximum, minimum, and a range of them as we did for the Gabor features in Sect. 5.3.6.

5.4 Descriptors

Descriptors have been very relevant on computer vision applications [25]. This is because they are able to provide highly distinctive features, and can be used in applications such as multiple view analysis, in object recognition, texture recognition, and others. In this section we provide some descriptors that are very useful in X-ray testing.

5.4.1 *Local Binary Patterns*

LBP, *Local Binary Patterns* was proposed as a texture feature [26]. The idea is to extract texture information from occurrence histogram of local binary patterns computed from the relationship between each pixel intensity value with its eight neighbors. The LBP features are the frequencies of each one of the histogram bins. LBP is computed in three steps: (i) coding, (ii) mapping, and (iii) histogram.

Coding

Each pixel (i, j) of the input image has a set of neighbors. Typically, the set of eight neighbors defined by the 8-connected pixels is used. However, more neighbors for different distances can be defined as well. For 8 connected pixels, the locations are $(i-1, j-1)$; $(i-1, j)$; $(i-1, j+1)$; $(i, j+1)$; $(i+1, j+1)$; $(i+1, j+1)$; $(i+1, j)$ and $(i+1, j-1)$, respectively, as shown in Fig. 5.12. The central pixel has a gray value q, and the neighbors have gray values p_i, for $i = 0 \ldots 7$. The code is computed by:

$$y = \sum_{i=0}^{7} t_i 2^i, \tag{5.32}$$

where $t_i = 1$ if $p_i \geq q$ otherwise $t_i = 0$. That means, a pixel q with its neighbors can be coded as a number $y \in \{0 \ldots 255\}$. The code can be represented as a string of bits as shown in Fig. 5.12.

Mapping

We can observe that the code generated by the previous step can be categorized according to number of changes (from '1' to '0', or from '0' to '1') in a cycle. For instance, in the example of Fig. 5.12, where the code is 01100111, we define a cycle with eight transitions as: $0 \rightarrow 1 \rightarrow 1 \rightarrow 0 \rightarrow 0 \rightarrow 1 \rightarrow 1 \rightarrow 1 \rightarrow 0$ (the last bit is the repetition of the first one because it is a cycle). The number of changes is $U = 4$. Thus, we can have codes with $U = 0, 2, 4, 6$ and 8 as illustrated in Fig. 5.13. After the authors, there are *uniform* and *nonuniform* patterns. The first ones ($U = 0$ and 2)

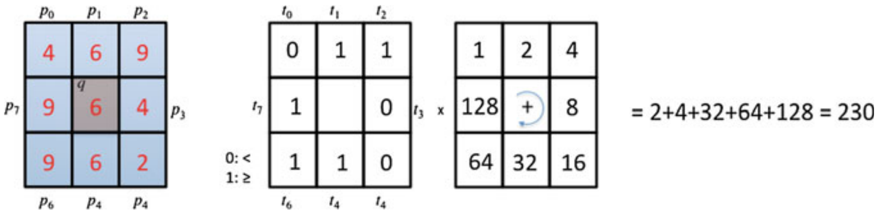


Fig. 5.12 LBP coding: a central pixel q the gray value of which is 6 has 8 neighbors with gray values $p_0 = 4$, $p_1 = 6$, $p_2 = 9$, $p_3 = 4$, $p_4 = 2$, $p_5 = 6$, $p_6 = 9$, and $p_7 = 9$. A new mask with 8 bits is built where $t_i = 1$ if $p_i \geq q$ otherwise $t_i = 0$. The LBP code is computed as $\sum_i t_i 2^i$, in this example the code is 230

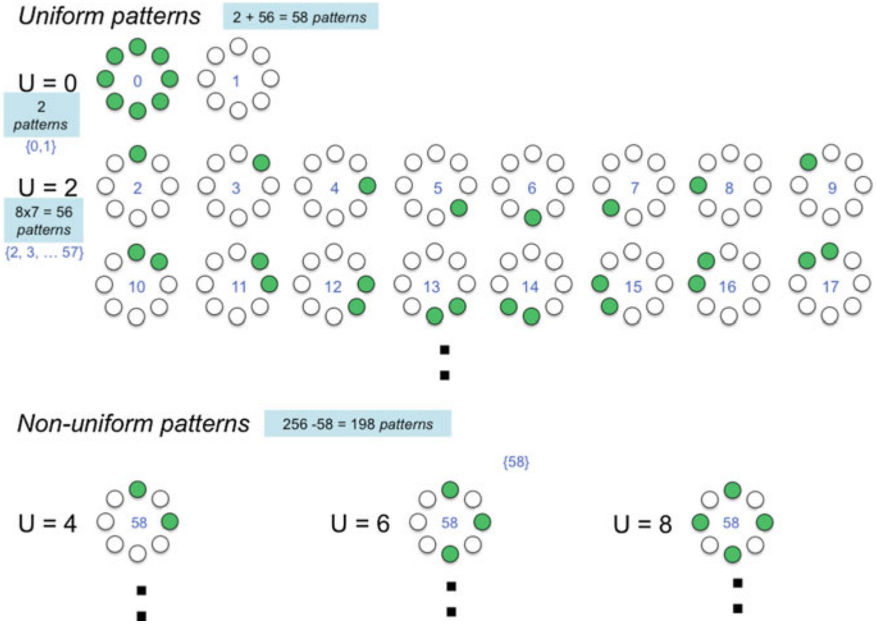


Fig. 5.13 LBP mapping for 8 neighbors. Each *small circle* represents a bit t_i of the code, *green* means '1', *white* means '0'. U is the number of changes from '1' to '0', or from '0' to '1' in one cycle. For a small number of changes, i.e., $U = 0$ and 2, the codes represent *uniform patterns*, for $U > 2$ the patterns are *nonuniform* (Color figure online)

correspond to textures with a low number of changes, the last ones ($U > 2$) can be interpreted as noise because there are many changes in the gray values. There are 58 uniform patterns and 198 nonuniform patterns. Each uniform code is mapped as a number from 0 to 57 as illustrated in Fig. 5.13, whereas all nonuniform codes are mapped as number 58. This descriptor is known as LBP-u2.

LBP-u2 mapping correspond to a mapping that varies with the orientation of the image, i.e., it is not rotation invariant. In order to build a rotation-invariant LBP descriptor, all patterns that have the same structure but with different rota-

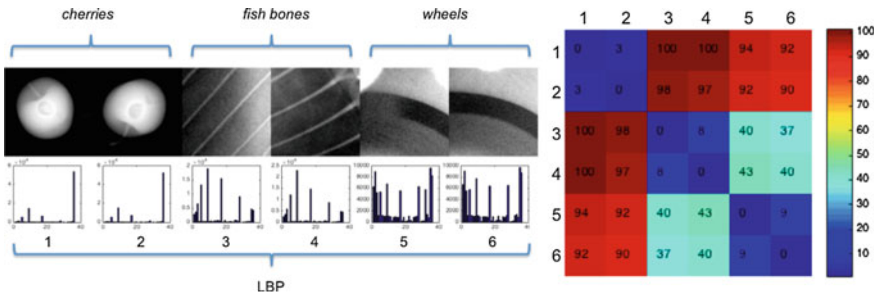


Fig. 5.14 Comparison of six textures using LBP-ri descriptor. It is clear that descriptors of the same texture are very similar, and descriptors from different textures are very different. A measurement of the Euclidean distance between all six descriptors is shown in the right color matrix

tions are mapped as an unique number. For instance, all patterns of the second row of Fig. 5.13 are mapped with the same number. The same is valid for the third row. In this mapping, we have 36 different numbers. This descriptor is known as LBP-ri.

Histogram

The process of coding and mapping is performed at each pixel of the input image. Thus, each pixel is converted into a number from 0 to $M - 1$, with a mapping of M numbers. Afterwards, a histogram of M bins of this image is computed. The LBP descriptor of the image is this histogram.

LBP is very robust in terms of grayscale and rotation variations [26]. An example is shown in Fig. 5.14. Other LBP features like *semantic* LBP (sLBP) [27] can be used in order to bring together similar bins. LBP is implemented in function Xlbp (see Appendix B) of 𝕏vis Toolbox.

The reader can find a descriptor with similar properties in [28], where LPQ (from *local phase quantization*) is proposed.

5.4.2 Binarized Statistical Image Features

BSIF, *binarized statistical image features*, was proposed as a texture descriptor [29]. As LBP and LPQ, it computes a binary code for each pixel of the input image. Thus, a histogram that encodes texture information is built by counting the frequency of each code.

In BSIF, the input image is filtered using a set of linear filters. The linear filters are learned from a training set of natural image patches ensuring statistical independence of the filter responses. BSIF computes the bits of the binary code by thresholding the response of the linear filters.

Therefore, instead of manually predefined sets of filters (like LBP or LPQ), BSIF uses filters based on statistics of natural images. After the authors, this improves its modeling capacity and the accuracy in texture recognition. BSIF is implemented in function Xbsif (see Appendix B) of $\mathbb{X}$VIS Toolbox. The reader can use this function to obtain similar results to those obtained by LBP in Fig. 5.14.

5.4.3 Histogram of Oriented Gradients

HOG, *histogram of oriented gradients*, was originally proposed as a descriptor that is able to detect pedestrians [30], however, the powerful of this descriptor can be used in many computer vision problems that require local object appearance and shape information. The key idea of HOG is to compute the distribution of intensity gradients in uniformly spaced cells arranged in a grid manner. A cell is typically defined a as squared region of the image.

In HOG, the gradient of the input image in both directions G_i and G_j is computed (see Sect. 4.4.1). Thus for each pixel, we have the magnitude $G(i, j)$ and the angle $A(i, j)$ using (4.15) and (4.15) respectively.[1] In order to compute the cell histogram, we define n bins, where bin k corresponds to the orientation between θ_k and θ_{k+1}, with $\theta_{k+1} = \theta_k + \Delta\theta$, for $k = 1 \dots n$. For example, for $n = 9$ bins we could define $\Delta\theta = 360°/9 = 40°$ and $\theta_1 = -\Delta\theta/2 = -20°$, so the first bin will be for orientations from $-20°$ to $+20°$, the second from $+20°$ to $+60°$ and so on. Therefore, a pixel (i, j) of the cell whose orientation is $\theta_k < A(i, j) \leq \theta_{k+1}$, registers a weighted vote in bin k based on its gradient value $G(i, j)$. This operation is repeated for every pixel of the cell.

In order to improve the performance of HOG descriptor and make it robust against changes in illumination and contrast, the authors used a dense grid of cells and an overlapping local contrast normalization [30]. That means, normalized cells are grouped together into connected blocks. Then, the descriptor is a concatenation of the normalized cell histograms of all blocks. HOG is implemented in function Xhog (see Appendix B) of $\mathbb{X}$VIS Toolbox. An example is illustrated in Fig. 5.15.

5.4.4 Scale-Invariant Feature Transform

The *Scale-invariant feature transform*, SIFT, was proposed in [31] to detect and describe *keypoints*. A keypoint is a distinguishable point in an image, i.e., it represents a salient image region that can be recognized by changing its viewpoint, orientation, scale, etc. In SIFT methodology, each keypoint is described using a 128-element vector called *SIFT-descriptor*. SIFT-descriptor is:

- Scale invariant
- Rotation invariant

[1] Sometimes the magnitude of the angle is used.

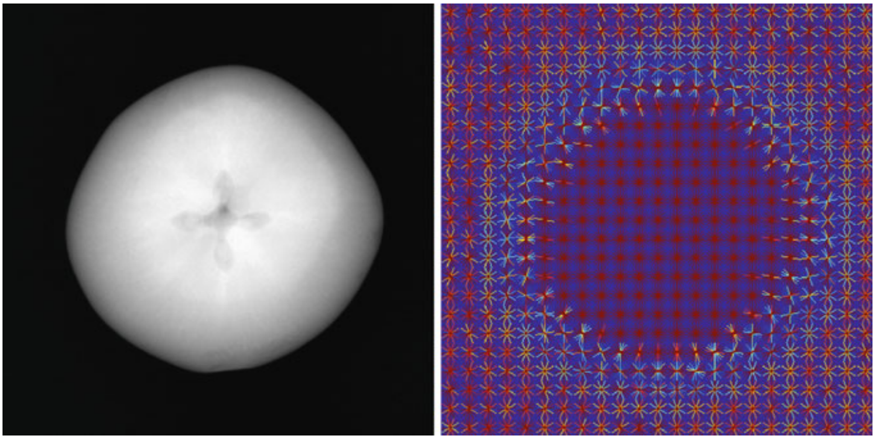


Fig. 5.15 Computation of HOG descriptors of an X-ray image of a fruit. The descriptors give information about shape and appearance

- Illumination invariant
- Viewpoint invariant

SIFT-descriptor can be used as a ‘signature’ and it is highly distinctive, i.e., SIFT-descriptors of corresponding points (in different images) are very similar, and SIFT-descriptors of different points are very different. SIFT has two main stages: (i) keypoint detection and (i) keypoint description. In the following, these stages are presented in further details.

Keypoint detection

Keypoints are then taken as maxima/minima of the Difference of Gaussians (DoG) that occur at multiple scales. Keypoints can be detected in four steps (see Fig. 5.16):

1. We define two Gaussian masks: $G(x, y, \sigma)$ and $G(x, y, k\sigma)$ from (4.10) at scales σ and $k\sigma$.
2. The input image $I(x, y)$ is convolved with both Gaussian filters obtaining $L(x, y, \sigma)$ and $L(x, y, k\sigma)$ respectively.
3. The Difference of Gaussians (DoG) is computed as:

$$D(x, y, \sigma) = L(x, y, \sigma) - L(x, y, k\sigma). \tag{5.33}$$

4. Keypoints are found as maxima of $|D(x, y, \sigma)|$ that can occur at different values of σ. We compare each pixel in the DoG images to its 26 neighbors (8 at the same scale and 9 from the next scale and 9 from the previous scales). If the pixel value is the maximum or minimum among all compared pixels, it is selected as a candidate keypoint.

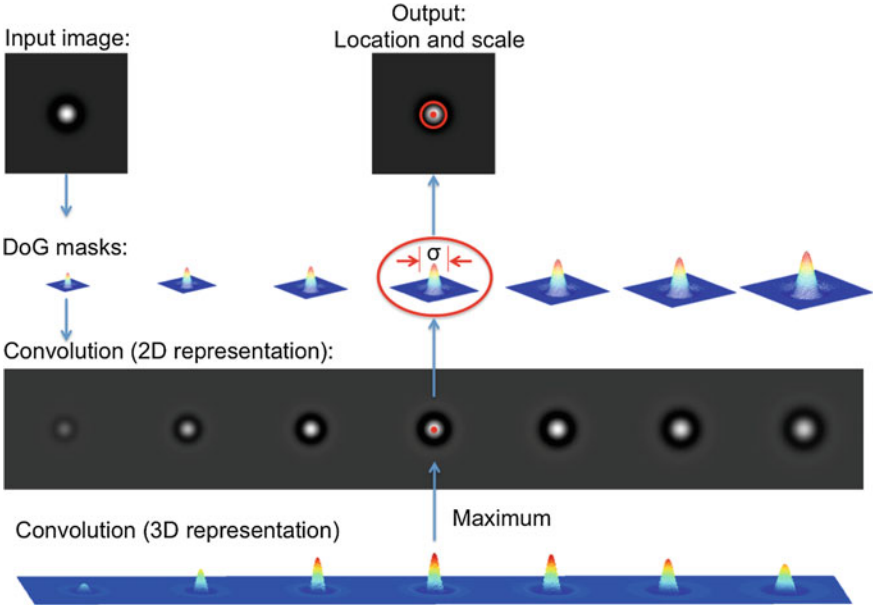

Input image:
Output:
Location and scale
DoG masks:
σ
Convolution (2D representation):
Convolution (3D representation)
Maximum

Fig. 5.16 Detection of a keypoint in a synthetic image. The image is convolved with several DoG masks. The maximal response defines the location (x, y) of the keypoint. The used mask for the convolution defines the scale σ

Keypoint description

For each keypoint we need a description. A keypoint is defined by its location (x, y) and its scale σ. The descriptor is computed in seven steps (see Fig. 5.17):

1. We define a window of size 1.5σ centered in (x, y).
2. The window is rotated $-\theta$, where θ is the orientation of the gradient in (x, y).
3. The rotated window is divided into $4 \times 4 = 16$ regular cells distributed in a grid manner.
4. For each cell, the histogram of gradients is computed using 8 bins.
5. All 16 histograms with 8 bins are concatenated, i.e., we obtain a descriptor of $16 \times 8 = 128$ elements.
6. Finally, the descriptor is normalized to unit length.

We can observe that SIFT descriptor is invariant to scale because the size of the window in step 1 depends on scale factor σ. SIFT descriptor is invariant to rotation, because the window is rotated according to the orientation of the gradient (see step 2). Thus, if an image is resized and rotated it will have the same window after these two steps. The SIFT descriptor is invariant to illumination because the descriptor is normalized to unit length. SIFT has been proven to be robust against perspective distortions and viewpoint changes when the rotation of the 3D object is less than 30° rotation. An example of this can be found in Fig. 5.18.

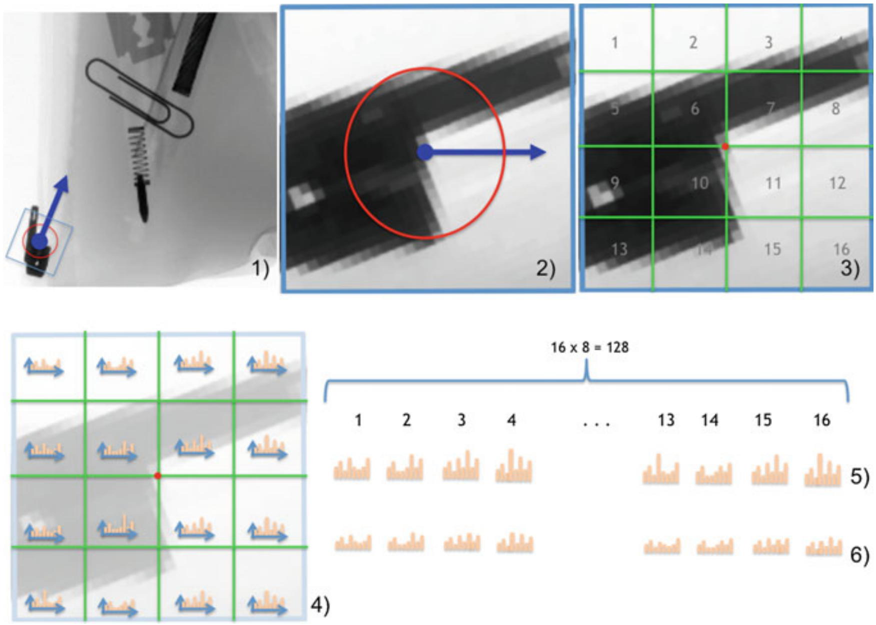


Fig. 5.17 Keypoint description (see explanation of six steps in text). This example corresponds to keypoint number 18 in Fig. 5.18

Matlab Example 5.7 In this example, we find matching points in two views. SIFT keypoints are estimated in each view, and those with the most similar descriptors are matched.

Listing 5.7 : Defects detections using area and CLP features

```
% SIFTmatching.m

I1 = single(Xloadimg('B',2,1));          % image 1
I2 = single(Xloadimg('B',2,2));          % image 2
figure(1); imshow(I1,[]); hold on
figure(2); imshow(I2,[]); hold on
[f1,d1] = vl_sift(I1);                   % SIFT descriptors for image 1
[f2,d2] = vl_sift(I2);                   % SIFT descriptors for image 2
[mt,sc] = vl_ubcmatch(d1,d2);            % matching points
[ii,jj] = sort(sc);                      % sort of scores
mt = mt(:,jj); sc = sc(:,jj);
n = 25;                                  % the best 25 matchings are selected

figure(1)                                % display results on image 1
h1 = vl_plotsiftdescriptor(d1(:,mt(1,1:n)),f1(:,mt(1,1:n))) ;
set(h1,'color','g') ;
for i=1:n
    plot(f1(1,mt(1,i)),f1(2,mt(1,i)),'ro')
    text(f1(1,mt(1,i))+5,f1(2,mt(1,i)),num2str(i),'fontsize',15)
end

figure(2)                                % display results on image 2
```

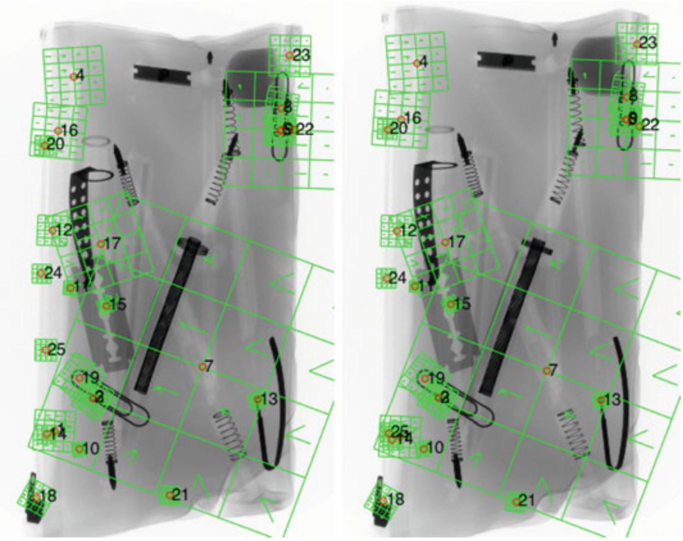


Fig. 5.18 Matching points of two different views of the same object. The object was rotated 10° around its horizontal axis from first to second image. The SIFT approach is able to find keypoints (*red* small circles) and descriptors (represented as *green* histograms). The descriptors that are similar can be matched. The figure shows the best 25 matching pair points (→ Example 5.7) (Color figure online)

```
h2 = vl_plotsiftdescriptor(d2(:,mt(2,1:n)),f2(:,mt(2,1:n))) ;
set(h2,'color','g') ;
for i=1:n
    plot(f2(1,mt(2,i)),f2(2,mt(2,i)),'ro')
    text(f2(1,mt(2,i))+5,f2(2,mt(2,i)),num2str(i),'fontsize',15)
end
```

The output of this code is shown in Fig. 5.18. In this example, the SIFT descriptors are detected using command `vl_sift` and matched with command `vl_ubcmatch` of VLfeat Toolbox [32]. □

The reader can find descriptors with similar properties in SURF: *Speeded Up Robust Feature* [33], BRIEF: *Binary robust independent elementary features* [34], BRISK: *Binary Robust-Invariant Scalable Keypoints* [35] among others.

5.5 Sparse Representations

In recent years, sparse representation has been widely used in signal processing [36], neuroscience [37], statistics [38], sensors [39] and computer vision [40, 41]. In many computer vision applications, under assumption that natural images can be represented using sparse decomposition [42] state-of-the-art results have been significantly improved. In these applications, the performance can be improved by learning nonparametric dictionaries for the sparse representation (instead of using fixed dictionaries).

In signal processing, it is very convenient to estimate a new representation of a signal in order to analyze it efficiently. The idea is that this representation captures a useful characterization of the signal for analytical tasks, e.g., feature extraction for pattern recognition, frequency spectrum for denoising, etc. An appropriate representation, due to its simplicity, is obtained by a linear transform. Thus, a signal $\mathbf{x} \in \mathbb{R}^n$ can be expressed as a linear combination of a set of elementary signals $\mathbf{D} = [\mathbf{d}_1\ \mathbf{d}_2\ \ldots\ \mathbf{d}_K] \in \mathbb{R}^{n\times K}$ as:

$$\mathbf{x} = \mathbf{D}\mathbf{z}, \tag{5.34}$$

where the vector $\mathbf{z} \in \mathbb{R}^K$ corresponds to the representation coefficients of signal $\mathbf{x}$. In this representation, matrix $\mathbf{D}$ and its columns $\mathbf{d}_k$ are commonly known as *dictionary* and *atoms* respectively.

5.5.1 Traditional Dictionaries

When every signal can be uniquely represented by a linear combination, the dictionary $\mathbf{D}$ corresponds to a *basis*. This is the case of Discrete Fourier Transform (DFT), for example, where the basis functions are sine and cosine waves with unity amplitude. In this case, the element j of atom $\mathbf{d}_k$ is defined as $d_{jk} = \exp(2\pi ijk/n)$ with $K = n$ and $i = \sqrt{-1}$ [24]. It is well known that for some applications, e.g., signal filtering, instead of processing the signal $\mathbf{x}$, it can be more convenient to process the signal in frequency domain $\mathbf{z}$, because it can be used to separate low and high frequencies effectively. Nevertheless, the Fourier basis is very inefficient when representing, for example a discontinuity, because its representation coefficients are over all frequencies and the analysis becomes difficult or even impossible. Other *predefined* basis, i.e., where the atoms are *fixed*, are Discrete Cosine Transformation (DCT) and Wavelets (e.g., Gabor) among others [24]. In many applications, since these dictionaries are fixed, they cannot represent more complex and high-dimensional signals satisfactorily [43].

In order to avoid the mentioned problem with fixed dictionaries, another way to represent a signal is using a *learned* dictionary, i.e., a dictionary that is estimated from representative signal examples. This is the case of Principal Component Analysis (PCA), or Karhunen–Loève Transform (KLT) [44], where the dictionary $\mathbf{D}$ is computed using the first K eigenvectors of the eigenvalue decomposition of the covariance matrix Σ, which is usually estimated from a set of zero-means signal examples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$. The basis here represents K orthogonal functions (with $K \leq n$) that transforms $\mathbf{X}$ into a set of linearly uncorrelated signals $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^N$ called the K *principal components*. This relationship is expressed as $\mathbf{X} = \mathbf{D}\mathbf{Z}$. In this case, KLT represents a signal more efficiently than DFT because the dictionary is not fixed and it is *learned* from signal examples [43].

The mentioned dictionaries are *orthogonal*, i.e., each atom $\mathbf{d}_i$ is orthogonal to atom $\mathbf{d}_j$ in $\mathbb{R}^n$ space $\forall i \neq j$. Therefore, a signal $\mathbf{x}$ is represented as a sum of orthogonal vectors $z_i \mathbf{d}_i$. In addition, most of these dictionaries are *orthonormal*, with $||\mathbf{d}_i|| = 1$ and $\mathbf{D}^{\mathsf{T}}\mathbf{D} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Hence, it is very simple to calculate $\mathbf{Z} = \mathbf{D}^{\mathsf{T}}\mathbf{X}$.

5.5.2 Sparse Dictionaries

Due to their mathematical simplicity, the orthogonal dictionaries dominated this kind of analysis for years. Nevertheless, there is no reason to accept as true that the number of atoms, required to characterize a set of signals, must be smaller than the dimension of the signal. Moreover, why should the atoms of the dictionary be orthogonal? The limited effectiveness of these dictionaries led to the development of newer dictionaries that can represent a wider range of signal phenomena, namely the *overcomplete* ones that have more atoms than the dimension of the signal ($K > n$) with no necessarily orthogonal atoms [45]. A seminal work in learning overcomplete dictionaries for image representation was presented by Olshausen and Field [42, 46]. They estimated—from small image patches of natural images—a *sparse* representation which was extremely similar to the mammalian simple-cell receptive fields (at that time, this phenomenon could only be described using Gabor filters). The key idea for representing natural signals is that although the number of possible atoms in the overcomplete dictionary is huge, the number of those atoms required to represent a signal is much smaller, i.e., the signals are sparse in the set of all possible atoms [45].

Sparse coding models a signal as a linear combination (5.34), or approximate, $\mathbf{x} \approx \mathbf{D}\mathbf{z}$, using a *sparse* linear combination of atoms from a learned dictionary, i.e., only a few atoms from $\mathbf{D}$ are allowed to be used in the linear combination (most coefficients of $\mathbf{z}$ are zero) and the atoms are not fixed (the dictionary is adapted to fit a given set of signal examples). In this case, the basis is not orthogonal.

Thus, from a representative set of signals $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, the idea is (i) to learn a dictionary $\mathbf{D} = \{\mathbf{d}_k\}_{k=1}^{K}$ and (ii) to estimate the corresponding sparse representations $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^{N}$ of the original signals $\mathbf{X}$.

In K-means algorithm –a very well–known algorithm used in clustering–, the sparsity is *extreme* because for the representation of $\mathbf{x}$ only one atom of $\mathbf{D}$ is allowed, and the corresponding coefficient of $\mathbf{z}$ is 1. In this case, the dictionary and coefficients are estimated by:

$$\mathbf{D}^*, \mathbf{Z}^* = \underset{\mathbf{D},\mathbf{Z}}{\text{argmin}}\ ||\mathbf{X} - \mathbf{D}\mathbf{Z}||_F^2 \quad \text{subject to } \forall i, \mathbf{z}_i = \mathbf{e}_k \text{ for some } k \tag{5.35}$$

where $\mathbf{e}_k$ is a vector from the trivial basis, with all zero entries except a one in kth position. In this equation, the Frobenius norm is used defined as $||\mathbf{A}||_F^2 = \sum_{ij} a_{ij}^2$. In clustering problems, the atom $\mathbf{d}_k$ is the centroid of samples $\mathbf{x}_i$ that fulfill $\mathbf{z}_i = \mathbf{e}_k$. Thus, a signal $\mathbf{x}$ belongs to cluster k if it is closer to centroid k than any other centroids (in this case, its representation is $\mathbf{z} = \mathbf{e}_k$ and the corresponding atom is $\mathbf{d}_k$).

Sparsity in general, can be expressed as:

$$\mathbf{D}^*, \mathbf{Z}^* = \underset{\mathbf{D},\mathbf{Z}}{\operatorname{argmin}} ||\mathbf{X} - \mathbf{DZ}||_F^2 \quad \text{subject to } ||\mathbf{x}_i||_0 \leq T \tag{5.36}$$

where $||\mathbf{x}_i||_0$ is the ℓ^0 norm, counting the nonzero entries of $\mathbf{x}_i$. The goal is to express a new signal $\mathbf{x}$ as a linear combination of a small number of signals take from the dictionary. This optimization problem can be expressed as:

$$\mathbf{z}^* = \underset{\mathbf{z}}{\operatorname{argmin}}(||\mathbf{x} - \mathbf{D}^* z||_2^2 + \lambda ||z||_1) \tag{5.37}$$

It can be demonstrated that the solution of the ℓ^0 minimization problem (5.36) is equivalent to the solution of the ℓ^1 minimization problem [47]:

$$\underset{\mathbf{D},\mathbf{Z}}{\operatorname{argmin}} ||\mathbf{X} - \mathbf{DZ}||_F^2 \quad \text{subject to } ||\mathbf{z}_i||_1 \leq T \tag{5.38}$$

Thus, on the one hand, the *dictionary learning problem* is as follows: given a set of training signals $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$, find the dictionary $\mathbf{D}$ (and a set of representation coefficients $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^N$) that represents at best each signal using the sparsity constraint (5.38), where no more than T atoms are allowed in each decomposition $\mathbf{z}_i$. On the other hand, the *sparse coding problem* can be stated as follows: given a signal $\mathbf{x}$ and a learned dictionary $\mathbf{D}$, find $\mathbf{z}$, the representation of signal $\mathbf{x}$, as:

$$\underset{\mathbf{z}}{\operatorname{argmin}} ||\mathbf{z}||_0 \quad \text{subject to } ||\mathbf{x} - \mathbf{Dz}||_2 < \varepsilon \tag{5.39}$$

where ε is the error tolerance.

5.5.3 Dictionary Learning

There are three categories of algorithms used to learn dictionaries [45]: (i) probabilistic methods, (ii) methods based on clustering, and (iii) methods with a particular construction. Probabilistic methods are based on a maximum likelihood approach, i.e., given the generative model (5.34), the objective is to maximize the likelihood that the representative samples have efficient, sparse representations in a redundant dictionary given by $\mathbf{D}$ [42, 48–50]. In clustering-based methods, the representative samples are grouped into patterns such that their distance to a given atom is minimal. Afterwards, the atoms are updated such that the overall distance in the group of patterns is minimal. This schema follows a K-means algorithm. In order to generalize the K-means algorithm, the 'K-SVD' algorithm was developed [51]. The method has two steps: (a) it uses orthogonal matching pursuit (OMP) algorithm for the sparse approximation,[2] (b) the columns of the dictionary are sequentially updated using singular value decomposition (SVD) decomposition to minimize the approximation error- It is reported that dictionaries learned with K-SVD show excellent performance in image denoising [53, 54] among other applications. Finally, dictionaries with specific structures use (instead of general forms of atoms) a set of *parametric functions* that can describe the atoms shortly, i.e., the generating functions and the parameters build the dictionary functions. Thus, the problem is reduced to learning the parameters for one or more generating functions (see for example [55, 56]). In Sect. 6.2.9 we will see how to use sparse representations for a classification task.

In 𝕏vis Toolbox, function Xsparsecl (see Appendix B) can be used to build a sparse dictionary. We will cover this topic in Sect. 6.2.9 when presenting the use of sparse representation in classification problems.

5.6 Feature Selection

Which features are relevant? or which features should be extracted? Such questions arise because there is a huge number of features that can be extracted and unfortunately, we do not know which or which of them are really necessary. First, we should not forget the reason why we extract features... so at least we could answer the question: why are they really necessary? As we explained in the introduction of this chapter (see Sect. 5.1), our task is to recognize or detect our *objects of interest*, and we need to differentiate them from the *background*. For example, in X-ray images of aluminum castings, we can have several *potential defects* that were detected using some segmentation approaches (see Sect. 4.5). As the segmentation is far from perfect, the potential defects consist of not only 'defects,' but 'regular structures' as well. Our object of interest in this example is the defects, whereas the background corresponds to the regular structure of the aluminum casting. From the X-ray images, we can extract features that describe the potential defects (e.g., area, width, height, location, contrast, statistical textures, etc.). In order to recognize the defects, we have to analyze the extracted features of the available potential defects and select those features that are able to properly separate the defects from the regular structures. In this example, we could expect a good separability of both classes by selecting the contrast (see Sect. 5.3.2) because it gives a measure of the difference in the gray value between the segmented region and its neighborhood.

[2] OMP is a greedy algorithm that iteratively selects locally optimal basis vectors [52].

5.6.1 Basics

In general, if we have two classes (ω_1 for 'object of interest' and ω_0 for 'background') and we want to analyze the performance of extracted feature x, e.g., contrast, we can investigate the frequency distribution for each class as illustrated in histograms of Fig. 5.19. In this case, for frequency distribution of class ω_k we only take into account the samples that belong to the kth class. In this *supervised* approach, the label d_i of ith sample must be available, for $i = 1 \ldots N$ for N samples. That means, someone, for example an expert, must annotate the label of each sample of the dataset. Thus, if the ith sample belongs to class ω_k, then $d_i = k$. For N samples we will have a vector $\mathbf{d}$ with N elements.

The available data should be representative enough, that means on the one hand that N_k, the number of samples of class ω_k, must be large enough, and on the other hand, for each class, the samples of the dataset must include the full range of variations that exist in the class itself. In our example, if x is the contrast of potential defects, we compute the frequency distribution of class ω_1 and the frequency distribution of class ω_0 by considering only the samples of 'defects' and 'regular structures' respectively. In addition, we can estimate the probability density functions from each frequency distribution known as $p(x|\omega_k)$, i.e., the probability of x given class ω_k. As we can see in Fig. 5.19, feature x is able to properly separate both classes because it takes low values for class ω_0 and high values for class ω_1, however, there is some degree of overlapping.

In feature selection, we have to decide just which features (extracted from our potential objects of interest) are relevant to the classification. By analyzing each extracted feature, three general scenarios are possible (see Fig. 5.20): a bad, a good, and a very good separability. In the first scenario, the confusion between both classes is so high that it is impossible to separate the classes satisfactorily, i.e., a classifier cannot distinguish either of the classes. In the second scenario, a good separation

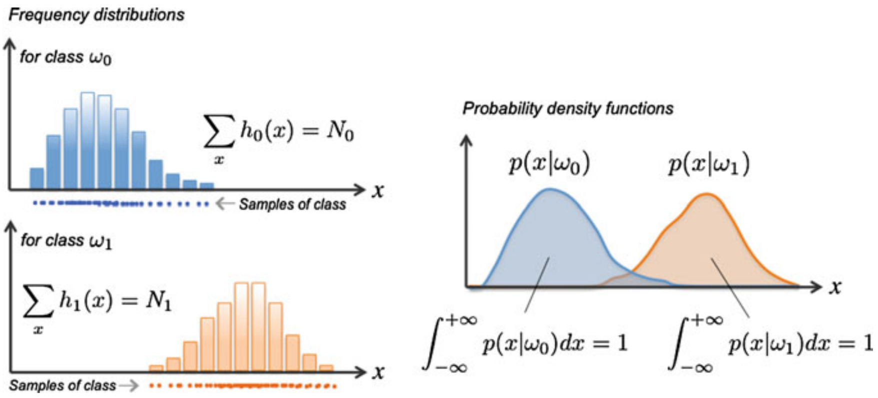


Fig. 5.19 A good class distribution for feature x and two classes ω_0 and ω_1

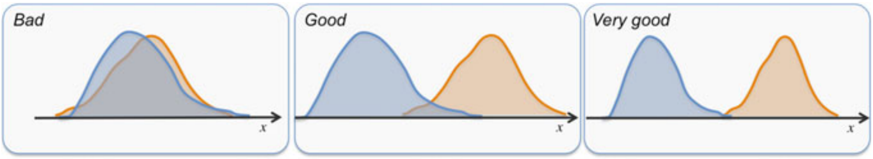


Fig. 5.20 Class distribution for there different features. It is clear that the best separability is achieved by the last features

is possible with some overlapping of the classes, i.e., a classifier will not recognize both classes perfectly, however, in many cases this scenario can be acceptable. In the third scenario, the separability is very good, and a classifier could identify both classes in approximately 100 % of the cases. If all extracted features are in the first scenario, there is no classifier that can separate both classes, i.e., new features are required. On the other hand, if we have a feature of the third scenario, the recognition can be easily performed by thresholding. In this case, no sophisticated classifiers are required. Unfortunately, the third scenario seldom occurs and we have to deal with some degree of overlapping.

In order to overcome the overlapping problem, more than one feature can be selected, however, the same three scenarios are also possible (see Fig. 5.21 for two features).

In this section, we will review some known techniques that can be used in feature selection. The reasons why feature selection is necessary are as follows:

1. It is possible that some extracted features are not *discriminative* enough, i.e., there is no information in these features for separating the classes. An example of this case is illustrated in the first scenario in Fig. 5.20. This may occur for example when we consider the mean gray value (5.16) of potential defects when detecting defects in welds. The (absolute) gray value of some defects can be very similar to the gray value of some regions of the background. In this example, we need rather a relative gray value such as a contrast (5.21).
2. Some extracted features with good separability could be redundant, i.e., they are somehow correlated. An example of this case is shown in the third scenario of Fig. 5.21 because x_1 are highly correlated with x_2. In this example, the separability by using (x_1, x_2) is very similar to the separability by using x_1 only. This may occur for example when we use two contrasts (5.21) to discriminate defects from background, maybe one contrast is enough and the second one does not increase the separability at all because it is redundant.
3. In order to simplify the testing stage, it is much better to extract a low number of features. In the training stage, we are allowed to investigate a huge number of features (in order to select some of them), however, in the testing stage it is recommended to use a reduced subset of these. Thus, the computational time of the testing stage will be significantly reduced.
4. In order to avoid the *curse of the dimensionality*, it is highly recommended to train a classifier with a low number of features. When we increase the number

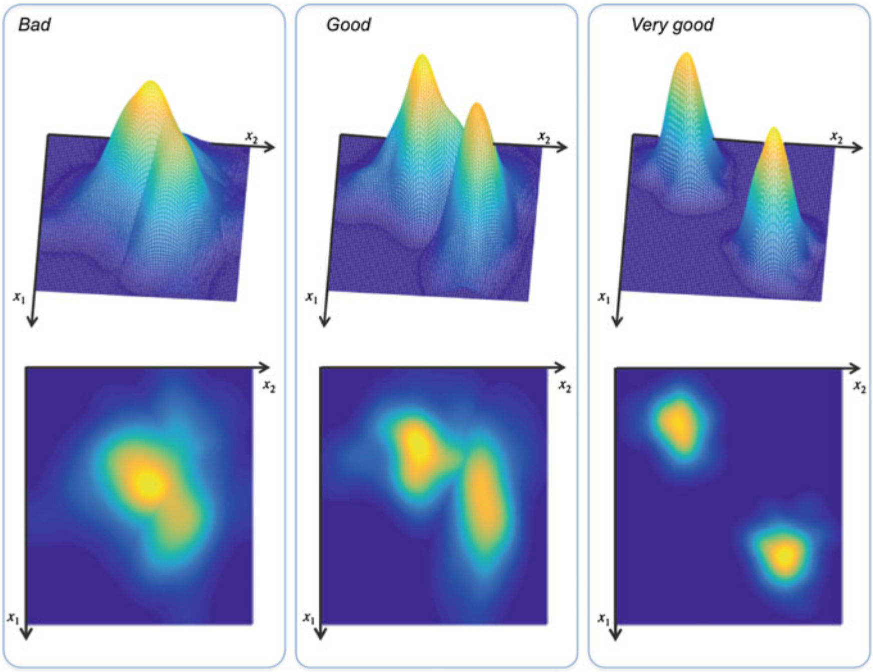


Fig. 5.21 Class distribution for three different pairs of features (x_1, x_2). As in Fig. 5.20, it is clear that the best separability is achieved by the last set of features. The figure shows two types of visualization of the feature space of two features: a 3D representation and a top view using a colormap. A third type of visualization for this data is available in Fig. 5.23

of selected features, the volume of our feature space increases exponentially. Thus, in order to be statistical significant we need to collect exponentially larger amounts of samples. This is not possible with a limited number of samples, for this reason the performance of the classifier tends to become reduced as the number of features increases [57].

5. Last but not least, in order to avoid *false correlations* some features should not have been extracted at all and must be filtered out in this step... just in case they were extracted. This is a very common mistake and it must be avoided before a classifier is trained. An example may be by trying to recognize a threat object (e.g., a knife) in baggage screening using features that are not rotation invariant. Imagine that we extract all elliptical features (see Sect. 5.2.2) of potential knives. The orientation α of the fitted ellipse is extracted as well (5.8). It is possible, that in our training dataset the orientation of the potential knives is always very vertical, as in the series `B0008` of 𝔾𝔻𝕏ray (see Fig. 2.10). That means, the extracted feature α could have a distribution like scenario two or three of Fig. 5.20. The *separability* of this feature could lead to misinterpretation because we could think that we found an extraordinary good feature that can separate

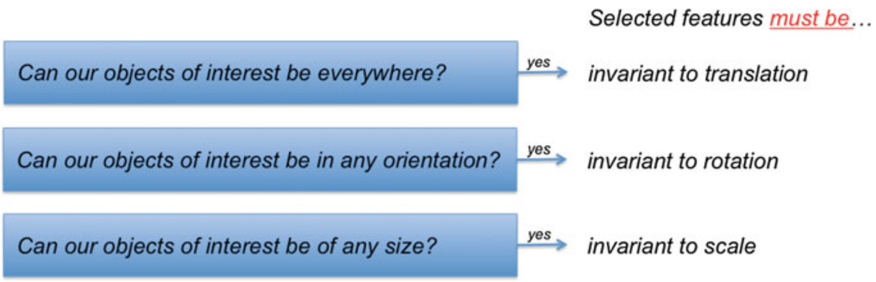

Selected features must be…
Can our objects of interest be everywhere?
yes
invariant to translation
Can our objects of interest be in any orientation?
yes
invariant to rotation
Can our objects of interest be of any size?
yes
invariant to scale

Fig. 5.22 In order to avoid false correlations we can follow these steps when extracting features. In these three cases, features that are extracted with 𝕏VIS Toolbox can be manually eliminated using commands Xnotranslation, Xnorotation and Xnoscale (see Appendix B) respectively

knives from background, however, we are saying that a knife must be always vertical if we want to recognize it! It is clear that the orientation should not have been extracted in order to avoid a false correlation. Another typical mistake occurs when considering the location (5.2) as feature in defect recognition. In our training data, it is possible that all defects are located in one part of the image, however, in real life they can be everywhere. Obviously, there is no algorithm that detects this error. When we design an automated system, we have to be very careful in order to select manually those features that could lead to false correlations. A guide to avoid this problem is suggested in Fig. 5.22.

Formally, the extracted features of a sample can be represented as a row vector $\mathbf{x}$ of m elements, where m is the number of extracted features. Thus, a sample can be viewed as a point $\mathbf{x} = [x_1 \ldots x_m]$ in the feature space of m dimensions (see Fig. 5.19 for one dimension and Fig. 5.23 for two dimensions). The feature vector of all samples can be stored in matrix $\mathbf{X}$ of size $N \times m$, where N is the number of samples, i.e., $N = \sum_k N_k$, and N_k is the number of samples of class ω_k. The jth column of $\mathbf{X}$, called $\mathbf{x}_j$, consists of the values that take feature x_i in all samples. In

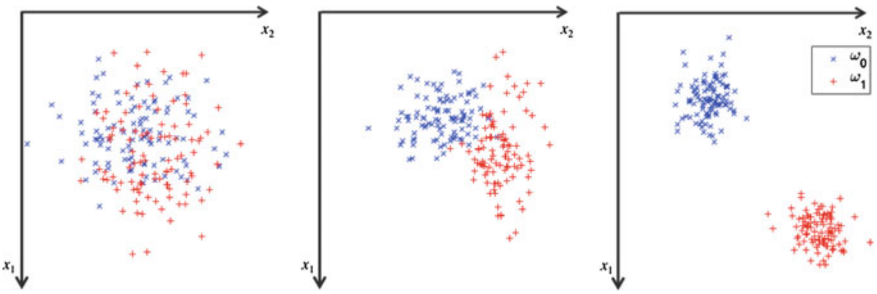


Fig. 5.23 In this visualization each sample is represented as a point in the feature space of two dimensions (x_1, x_2). The figure shows the visualization for the three examples of Fig. 5.21

addition, element x_{ij} means the feature x_j of ith sample. The features are usually normalized as:

$$\tilde{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{5.40}$$

for $i = 1, \ldots, N$ and $j = 1, \ldots, m$, where μ_j and σ_j are the mean and standard deviation respectively of the $\mathbf{x}_j$. The normalized features have zero mean and a standard deviation equal to one.[3]

A very good practice is to eliminate (i) those features that are very constant, i.e., $\sigma_j < \theta_1$, where θ_1 is some threshold, e.g., 10^{-8}, and (ii) those features that are very correlated, i.e., if two of any extracted features ($\mathbf{x}_i$ and $\mathbf{x}_j$) are highly correlated (if $|\text{cov}(\mathbf{x}_i, \mathbf{x}_j)|/(\sigma_i\sigma_j) > \theta_2$) one of them is eliminated. We can set θ_2 to 0.99 for example. The feature 'cleaning' is implemented in function Xfclean (see Appendix B) of $\mathbb{X}$VIS Toolbox.

The key idea of the feature selection is to select a subset of p features ($p \leq m$) that leads to the smallest classification error. The selected p features are arranged in a new row vector of p elements $\mathbf{z} = [z_1 \ldots z_p]$. The selected feature vector of all samples can be stored in matrix $\mathbf{Z}$ of size $N \times p$. This process is illustrated in Fig. 5.24 for $m = 10$ and $p = 3$. The p selected features are columns $s_1, s_2 \ldots s_p$ of $\mathbf{X}$, that means column j of $\mathbf{Z}$ is equal to column s_j of $\mathbf{X}$, $\mathbf{z}_j = \mathbf{x}_{s_j}$, for $j = 1 \ldots p$.

For a given set of selected features $\mathbf{s} = (s_1, s_2 \; \ldots \; s_p)$ we need some measurement of *separability* that can be used to assess the performance of the selection, i.e., for our three scenarios (see Figs. 5.20 and 5.21), this measurement should be low, high and very high respectively. We define the separability J as a function of $\mathbf{Z}$ (selected features) and $\mathbf{d}$ (labels of the samples). Since $\mathbf{Z}$ corresponds to the selected columns of $\mathbf{X}$ that are defined by $\mathbf{s}$, we can write the separability as $J(\mathbf{X}, \mathbf{s}, \mathbf{d})$.

The problem of feature selection can be stated as follows, given the extracted features for N samples ($\mathbf{X}$) and the labels of each sample ($\mathbf{d}$), find a set of features (indexed by $\mathbf{s} = (s_1, s_2 \; \ldots \; s_p)$) that maximizes the separability ($J(\mathbf{X}, \mathbf{s}, \mathbf{d})$). This is an optimization problem

$$\hat{\mathbf{s}} = \underset{\mathbf{s} \subseteq \mathbf{Q}}{\text{argmax}} \, J(\mathbf{X}, \mathbf{s}, \mathbf{d}), \qquad \text{s.t.} |\mathbf{s}| = p, \tag{5.41}$$

where $\mathbf{Q} = (1, 2, \ldots m)$ is the set of all possible indices that can take $\mathbf{s}$.

There are many approaches that can be used to measure the separability. A very common one is based on Fisher criterion that ensures: (i) a small intraclass variation and (ii) a large interclass variation in the space of the selected features.

For the first condition, the intraclass covariance (known also as between-class covariance matrix) is used:

$$\mathbf{C}_b = \sum_k p_k (\bar{\mathbf{z}}_k - \bar{\mathbf{z}})(\bar{\mathbf{z}}_k - \bar{\mathbf{z}})^{\mathsf{T}}, \tag{5.42}$$

[3] In $\mathbb{X}$VIS Toolbox, (5.40) is implemented in function Xfnorm (see Appendix B).

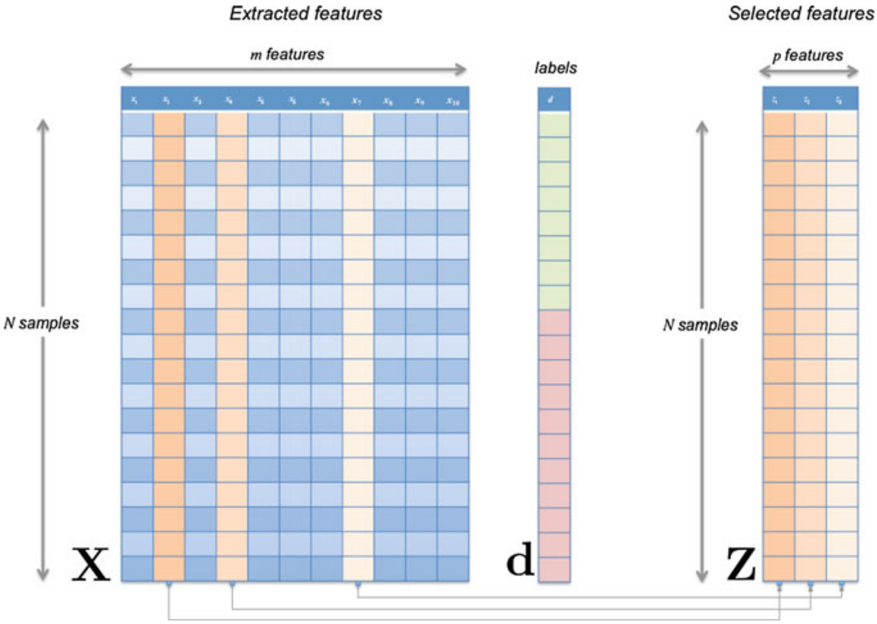


Fig. 5.24 Feature selection: there are m extracted features, from them p are selected. As we can see in the labels, the first samples belong to one class and the last one to another

where p_k denotes the a priori probability of the kth class, $\bar{\mathbf{z}}_k$ and $\bar{\mathbf{z}}$ are the mean value of the kth class and the mean value of the selected features.

For the second condition, the interclass covariance (known also as within-class covariance matrix) is used:

$$\mathbf{C}_w = \sum_{k=1}^{K} p_k \mathbf{C}_k, \tag{5.43}$$

where the covariance matrix of the kth class is given by:

$$\mathbf{C}_k = \frac{1}{N_k - 1} \sum_{j=1}^{N_k} (\mathbf{z}_{kj} - \bar{\mathbf{z}}_k)(\mathbf{z}_{kj} - \bar{\mathbf{z}}_k)^{\mathsf{T}}, \tag{5.44}$$

with $\mathbf{z}_{kj}$ the jth selected feature vector of the kth class, N_k is the number of samples in the kth class. Selection performance can be evaluated using the spur criterion for the selected features $\mathbf{z}$:

$$J = \text{spur}\left(\mathbf{C}_w^{-1}\mathbf{C}_b\right). \tag{5.45}$$

where 'spur' means the sum of the diagonal. The larger the objective function J, the higher the selection performance. For the examples of Fig. 5.23, this function takes the values 0.1, 2.1 and 27.8 respectively. The objective function defined in (5.45) can be used directly in (5.41).

Another approach that can be used to measure the separability is to compute the accuracy of a classifier with the selected features. In this approach, we divide $\mathbf{Z}$ in two subsets of samples: training and testing datasets. A classifier is designed using the training set, and afterwards is tested using the testing set. The separability J is defined as the accuracy evaluated on the testing set, i.e., the ratio of samples that were correctly classified to the total number of samples.[4]

The features can be selected using several state-of-art algorithms reported in the literature. In the following, some selection algorithms are presented.

5.6.2 Exhaustive Search

The selection of the features is performed by evaluating (5.41) for all possible combination of p features of $\mathbf{X}$. The combination that achieves the highest value for J is selected. This approach ensures that global maximum of J is attained, however, it requires $n = m!/(p!(m-p)!)$ evaluations of J. The number n can be prohibited for large m and p values. For instance, if we have $m = 100$ extracted features and we want to select $p = 10$ features, then 1.73×10^{13} evaluations of J are required using exhaustive search. This function is implemented in command Xfexsearch (see Appendix B) of $\mathbb{X}$VIS Toolbox.

5.6.3 Branch and Bound

In Branch and bound, the global maximum of J is ensured also [58]. Given that J is a monotonically increasing function, i.e., $J(\mathbf{z}_1) < J(\mathbf{z}_1, \mathbf{z}_2) < \cdots J(\mathbf{z}_1, \ldots \mathbf{z}_p)$, we can considerably reduce the number of evaluations of J. In branch and bound technique, we use a tree representation, where the root corresponds to the set of all features, and a node of the tree corresponds to a combination of features. The children's nodes are subsets of their parents. Nodes in the kth level represent combinations of $m - k$ features. We starts by evaluating J at the main node ($k = 0$) with all features. This will be our *bound*, the current maximum. The key idea of the algorithm is to evaluate those children nodes that have a separability J higher than the bound. If that is the case, the we update the bound. Consequently, nodes whose separability J is lower than the bound will not be evaluated. This method is implemented in command Xfbb (see Appendix B) of $\mathbb{X}$VIS Toolbox.

[4] Classifiers and accuracy estimation are covered in Chap. 6.

5.6.4 Sequential Forward Selection

This method selects the best single feature and then adds one feature at a time that, in combination with the selected features, maximizes the separability. The iteration is stopped once the selected subset reaches p features. This method requires $n = pm - p(p-1)/2$ evaluations. For instance, if we have $m = 100$ extracted features and we want to select $p = 10$ features, then 955 evaluations of J are required using SFS, this is a very low number in comparison with the number of evaluations required for exhaustive search. This method is implemented in command Xsfs (see Appendix B) of $\mathbb{X}$VIS Toolbox.

Matlab Example 5.8 In this example, we extract intensity features of small cropped X-ray images (100 × 100 pixels) of salmon filets. The cropped images are in series N0002 of $\mathbb{GDX}$ray. There are 100 cropped images with fishbones and 100 with no fishbones. The idea is to select those features that can be relevant for the separation between both classes 'fishbones' and 'background' (labels 1 and 0 respectively). Using the selected features, we could detect small regions with fishbones in an X-ray image of a salmon filet. In this series, the labels of the 200 cropped images are available. We initially extract several intensity features (more than 300). Afterwards, features that are not rotation invariant are eliminated because fishbones can be oriented in any direction. Additionally, high correlated or constant features are eliminated as well. We select 15 features using SFS and 3 from them using exhaustive search. The computational time of the feature selection step is short because we are dealing with a small number of features and samples.

Listing 5.8 : Feature selection with SFS

```
% FeatureSelectionSFS.m
close all
gdx_dir      = Xgdxdir('N',2);                  % directory of series N0002 of GDX
opf.b        = Xfxbuild({'basicint','gabor',... % features to be extracted
                         'lbpri','haralick'});  % rotation invariant for LBP
[X0,Xn0]     = Xfxtractor(gdx_dir,'png',opf);   % feature extraction
[X,Xn]       = Xnorotation(X0,Xn0);             % only rotation invariant features
d            = Xloaddata('N',2,'labels.txt');   % labels
sc           = Xfclean(X);                      % delete constant and correlated features
figure
Xc           = X(:,sc);                         % sc = indices of selected features
Xcn          = Xn(sc,:);
opsfs.show   = 1;                               % display results
opsfs.p      = 15;                              % 15 features will be selected
s1           = Xsfs(Xc,d,opsfs);                % using SFS
Y1           = Xc(:,s1);                        % s1 = indices of selected features
Y1n          = Xcn(s1,:);
opexs.show   = 1;                               % display results
opexs.p      = 3;                               % 3 (from 15) features will be selected
s2           = Xfexsearch(Y1,d,opexs);          % using exhaustive search
Y2           = Y1(:,s2);
Y2n          = Y1n(s2,:);
figure
Xplotfeatures(Y2,d,Y2n)                         % plot of feature space
grid on; view(-25,30)
```

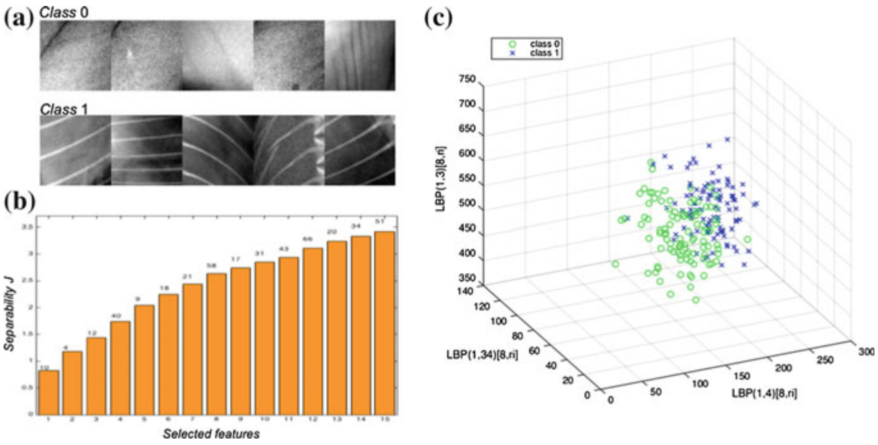


Fig. 5.25 Feature selection using SFS and exhaustive search: **a** Some cropped X-ray images of both classes ‘background’ and ‘fishbones’. In this example there are 200 cropped X-ray images, 100 from each class. There are $m = 241$ extracted features from $N = 200$ samples, i.e., the extracted features are stored in matrix $\mathbf{X}$ of size 200 × 241. **b** Sequential forward selection. There are $p_1 = 15$ selected features and they correspond to columns 10, 4, 12, 40, 9 ... of $\mathbf{X}$, i.e., the selected features are stored in matrix $\mathbf{Y}_1$ of size 200 × 15. **c** Using exhaustive search, $p_2 = 3$ features are selected from $\mathbf{Y}_1$. The result is stored in matrix $\mathbf{Y}_2$ of 200 × 3 elements. The figure shows the feature space in 3D. The selected features are certain LBP features. We can see that the separability is ‘good’ and correspond to our second scenario (→ Example 5.8)

The output of this code is shown in Fig. 5.25. In this example, the features were extracted using commands Xfxtractor (see Appendix B), and the features were selected using Xsfs and Xfexsearch (see Appendix B) of 𝕏vis Toolbox. In this experiment, only LBP feature were selected, i.e., in testing stage it is not necessary to extract Gabor, Haralick and other basic intensity features. This result is very meaningful because the computational time is considerably reduced: from 0.32 s/image to 0.018 s/image (in a computer with 8GB RAM and a processor Intel Core i5 of 2.8 GHz, OS X 10.10.2). □

5.6.5 Sequential Backward Selection

This method selects all features and then eliminates one feature at a time that maximizes the separability. The iteration is stopped once the selected subset reaches p features. This method requires $n = (m - p + 1)m - (m - p)(m - p + 1)/2$ evaluations. For instance, if we have $m = 100$ extracted features and we want to select $p = 10$ features, then 5005 evaluations of J are required using SBS.

5.6.6 Ranking by Class Separability Criteria

Features are ranked using an independent evaluation criterion to assess the significance of every feature for separating two labeled groups. The absolute value two–sample t–Student test with pooled variance estimate is used as an evaluation criterion [59]. This method is implemented in command Xfrank (see Appendix B) of $\mathbb{X}$VIS Toolbox.

5.6.7 Forward Orthogonal Search

In FOS, features are selected one at a time, by estimating the capability of each specified candidate feature subset to represent the overall features in the measurement feature space using a squared correlation function to measure the dependency between features [60]. This method is implemented in command Xfosmod (see Appendix B) of $\mathbb{X}$VIS Toolbox.

5.6.8 Least Square Estimation

In LSE, features are selected one at a time, evaluating the capacity of the select feature subsets to reproduce sample projections on principal axis using Principal Component Analysis (PCA) [61]. This method is implemented in command Xlsef (see Appendix B) of $\mathbb{X}$VIS Toolbox.

5.6.9 Combination with Principal Components

The first p principal components of the large set of features $\mathbf{X}$ (or a preselected subset of features using one of the mentioned approaches) are appended as new columns (features) of $\mathbf{X}$. Thus, we have a new set of features $\mathbf{X}_{\text{new}} = [\mathbf{X} \quad \text{pca}(\mathbf{X}, p)]$. Afterwards, a feature selection algorithm (like SFS or exhaustive search) is computed on $\mathbf{X}_{\text{new}}$. As result, the selected features can be some original features and some principal components [62]. An example is shown in Fig. 5.26. In this example, this method achieved the best separability with only three features, however, it is worth mentioning that using this method the computational time is increased significantly in the testing stage. The reason is not because we have to compute the PCA transformation, but because we have to extract all features required by PCA.

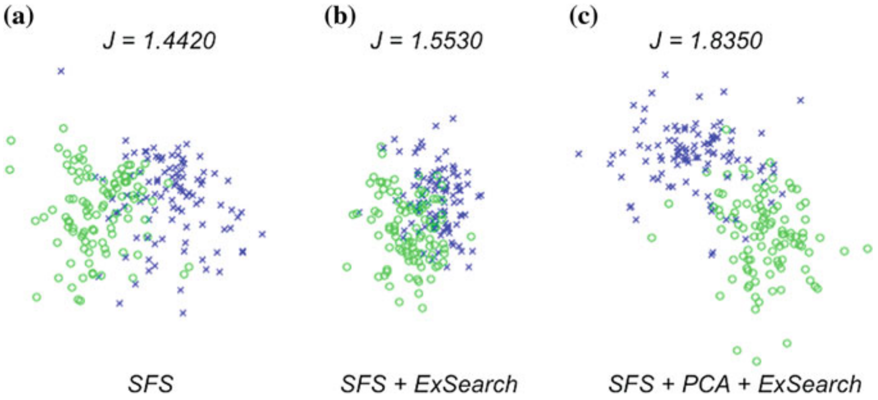


Fig. 5.26 Separability of three different feature selection methods for fishbone detection (see Example 5.8 for details). Each visualization is a 3D plot, the axes and the grid are not represented for the sake of simplicity. **a** The best three features after SFS. It corresponds to the first three columns of variable `Y1` that has 15 columns. **b** The best three features (from 15 features selected by SFS) after exhaustive search. It correspond to variable `Y2`. This plot is the same of Fig. 5.25c. **c** The three principal components of `Y1` is computed (`Ypca = Xpca(Y1,3);`). A new set of features including `Y1` and `Ypca` is defined (`Ynew = [Y1 Ypca];`). The plot shows the best three features of `Ynew` that are selected using exhaustive search. The last selection included one principal component. We observe how the separability J after Fisher criterion (5.45) is increased in each step

5.6.10 Feature Selection Based in Mutual Information

In mRMR, the features are selected based on two criteria: minimal redundancy in order to remove redundant variables; and maximal relevance in order to select the relevant features that are able to separate the classes [63]. This method is implemented in command XfmRMR (see Appendix B) of 𝕏vis Toolbox.

5.7 A Final Example

In this example, we show how to extract and select features for a three-class problem. We want to separate handguns, shuriken and razor blades (see some samples in Fig. 5.27). We extract geometric features that are invariant to rotation, translation and scale. The separation is easy because the shapes are very different. Probably, this particular example does not have any application in real life, but it shows how we can use 𝕏vis Toolbox easily to extract and select features for a classification task.

The features can be extracted using a simple Matlab code (as shown in Example 5.9) or using a graphic user interface (as shown in Fig. 5.28). With these commands it is really simple to design a program that is able to extract and select many features.

Matlab Example 5.9 This example shows a simple code that is used to extract and select features. The task is to separate handguns, shuriken and razor blades (Fig. 5.27). The reader can easily adapt this code to similar recognition problems.

Listing 5.9 : Feature extraction and selection

```
% SeparationExample.m
clt
dir_obj1 = Xgdxdir('B',49);                       % directory of guns (class 1)
dir_obj2 = Xgdxdir('B',50);                       % directory of shuriken (class 2)
dir_obj3 = Xgdxdir('B',51);                       % directory of razor blades (class 3)
opf.b    = Xfxbuild({'basicgeo','fitellipse','flusser','fourierdes',...
                     'gupta','hugeo'});           % features to be extracted
opf.segmentation = 'Xsegbimodal';                 % segmentation approach
[X1,X1n] = Xfxtractor(dir_obj1,'png',opf);        % feature extraction of class 1
[X2,X2n] = Xfxtractor(dir_obj2,'png',opf);        % feature extraction of class 2
[X3,X3n] = Xfxtractor(dir_obj3,'png',opf);        % feature extraction of class 3
N1 = size(X1,1); N2 = size(X2,1); N3 = size(X3,1);% number of samples per class
d = [ones(N1,1); 2*ones(N2,1); 3*ones(N3,1)];     % labels
X0 = [X1;X2;X3]; X0n = X1n;                       % features of all classes
[Xa,Xan] = Xnorotation(X0,X0n);                   % only rotation invariant
[Xb,Xbn] = Xnotranslation(Xa,Xan);                % only translation invariant
[Xc,Xcn] = Xnoscale(Xb,Xbn);                      % only scale invariant
s0 = Xfclean(Xc);                                 % cleaning
Xd = Xc(:,s0); Xdn = Xcn(s0,:);
op.p = 3; op.show = 1;                            % 3 features will be selected
f = Xfnorm(Xd,1);                                 % normalization
s = Xsfs(f,d,op);                                 % SFS with Fisher cirterion.
Xs = f(:,s); Xns = Xdn(s,:);                      % selected features
figure
Xplotfeatures(Xs,d,Xns);                          % feature space
```

The output of this code is shown in Fig. 5.29. In this example, we use several powerful functions of 𝕏VIS Toolbox:

- Xfxbuild (see Appendix B): Build structure for feature extraction with default values.
- Xfxtractor (see Appendix B): Feature extraction from a set of images.
- Xnorotation (see Appendix B): Delete all no rotation-invariant features.
- Xnotranslation (see Appendix B): Delete all no translation invariant features.
- Xnoscale (see Appendix B): Delete all no scale-invariant features.
- Xplotfeatures (see Appendix B): Plot feature space.

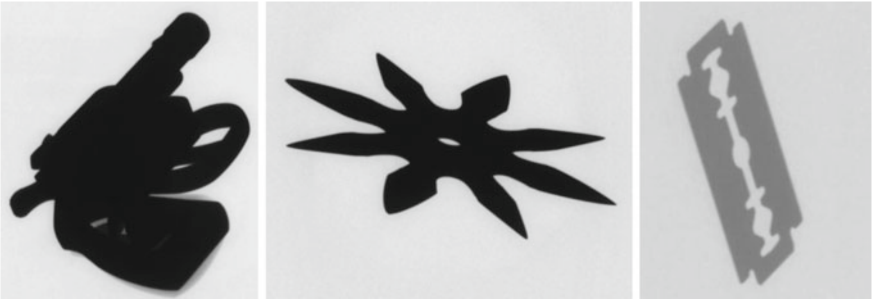
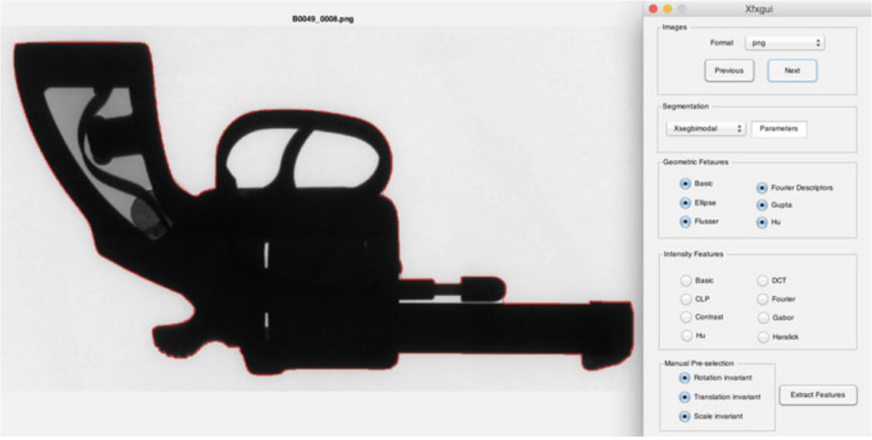


Fig. 5.27 Some objects used in example of Sect. 5.7: a handgun, a shuriken and a razor blade, from 𝔾𝔻𝕏ray series `B0049`, `B0050`, and `B0051` respectively



Format: png

Segmentation: Xsegbimodal

Geometric Fetaures: ● Basic ● Ellipse ● Flusser ● Fourier Descriptors ● Gupta ● Hu

Intensity Features: ○ Basic ○ CLP ○ Contrast ○ Hu ○ DCT ○ Fourier ○ Gabor ○ Haralick

Manual Pre-selection: ● Rotation invariant ● Translation invariant ● Scale invariant

Fig. 5.28 Graphic user interface Xfxgui (see Appendix B) of 𝕏VIS Toolbox for feature extraction. In this example, all geometric features of all images of series B0049 of 𝔾𝔻𝕏ray are extracted. In addition, the user can specify that the features to be extracted must be rotation, translation and scale invariant

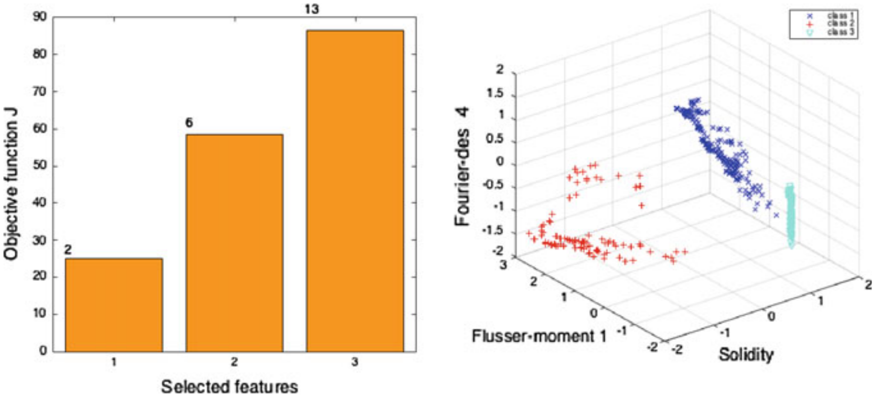


Fig. 5.29 Separation of three classes: *1* Handguns, *2* Shuriken, *3* Razor blades. *Left* Feature selection using SFS. *Right* Feature space (→ Example 5.9)

From Fig. 5.29 it is very simple to design a classification strategy (e.g., using thresholds) and Table 5.1. The basic geometric features are extracted by command Xbasicgeo (see Appendix B) of 𝕏VIS Toolbox. □

5.8 Summary

In this chapter we covered several topics that are used to represent an X-ray image (or a specific region of an X-ray image). This representation means that new features are extracted from the original image and that they can give us more information than the raw information expressed as a matrix of gray values.

In the first part of this chapter, we learned about geometric and intensity features. We reviewed basic geometric features (such as area and perimeter among others), elliptical features, Fourier descriptors, and invariant moments. Further, we addressed basic intensity features, several definitions of contrast, crossing line profiles (CLP), intensity moments, statistical textures, Gabor and filter banks (such as Fourier and Discrete Cosine Transform).

In the second part of this chapter, we gave an overview of certain descriptors that are widely used in computer vision and can be a powerful tool in X-ray testing. We covered local binary patterns (LBP), binarized statistical image features (BSIF), histogram of oriented gradients (HOG), and scale-invariant feature transform (SIFT).

In the third part of this chapter, we studied sparse representations. They have been widely used in computer vision. In X-ray testing, they can be used in problems of object recognition as we will see in the next chapter.

In the fourth part of this chapter, we presented different feature selection techniques that can be used to choose which features are relevant for a classification problem. Some of the techniques are sequential feature selection, branch and bound and feature selection based on mutual information.

Finally, we gave a simple code as an example that can be used to extract and select features for a classification problem.

References

1. Coeurjolly, D., Klette, R.: A comparative evaluation of length estimators of digital curves. IEEE Trans. Pattern Anal. Mach. Intell. **26**(2), 252–258 (2004)
2. MathWorks: Image Processing Toolbox for Use with MATLAB: User's Guide. The MathWorks Inc. (2014)
3. Fitzgibbon, A., Pilu, M., Fisher, R.: Direct least square fitting ellipses. IEEE Trans. Pattern Anal. Mach. Intell. **21**(5), 476–480 (1999)
4. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2003)
5. Zahn, C., Roskies, R.: Fourier descriptors for plane closed curves. IEEE Trans. Comput. **C-21**(3), 269–281 (1971)
6. Chellappa, R., Bagdazian, R.: Fourier coding of image boundaries. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-6**(1), 102–105 (1984)
7. Persoon, E., Fu, K.: Shape discrimination using Fourier descriptors. IEEE Trans. Syst. Man Cybern. **SMC-7**(3), 170–179 (1977)
8. Castleman, K.: Digital Image Processing. Prentice-Hall, Englewood Cliffs (1996)
9. Hu, M.K.: Visual pattern recognition by moment invariants. IRE Trans. Inf. Theory **IT**(8), 179–187 (1962)

10. Teh, C., Chin, R.: On digital approximation of moment invariants. Comput. Vis. Graph. Image Process. **33**(3), 318–326 (1986)
11. Gupta, L., Srinath, M.D.: Contour sequence moments for the classification of closed planar shapes. Pattern Recognit. **20**(3), 267–272 (1987)
12. Sonka, M., Hlavac, V., Boyle, R.: Image Processing, Analysis, and Machine Vision, 2nd edn. PWS Publishing, Pacific Grove (1998)
13. Flusser, J., Suk, T.: Pattern recognition by affine moment invariants. Pattern Recognit. **26**(1), 167–174 (1993)
14. Jähne, B.: Digitale Bildverarbeitung, 2nd edn. Springer, Berlin (1995)
15. Kamm, K.F.: Grundlagen der Röntgenabbildung. In: Ewen, K. (ed.) Moderne Bildgebung: Physik, Gerätetechnik, Bildbearbeitung und -kommunikation, Strahlenschutz, Qualitätskontrolle, pp. 45–62. Georg Thieme Verlag, Stuttgart (1998)
16. Mery, D., Filbert, D.: Classification of potential defects in automated inspection of aluminium castings using statistical pattern recognition. In: 8th European Conference on Non-Destructive Testing (ECNDT 2002), pp. 1–10. Barcelona (2002)
17. Klette, R.: Concise Computer Vision: An Introduction into Theory and Algorithms. Springer Science & Business Media, London (2014)
18. Mery, D.: Crossing line profile: a new approach to detecting defects in aluminium castings. In: Proceedings of the Scandinavian Conference on Image Analysis (SCIA 2003). Lecture Notes in Computer Science, vol. 2749, pp. 725–732 (2003)
19. Mery, D., Filbert, D.: Automated flaw detection in aluminum castings based on the tracking of potential defects in a radioscopic image sequence. IEEE Trans. Robot. Autom. **18**(6), 890–901 (2002)
20. Mery, D., Filbert, D., Jaeger, T.: Image processing for fault detection in aluminum castings. In: MacKenzie, D., Totten, G. (eds.) Analytical Characterization of Aluminum and Its Alloys. Marcel Dekker, New York (2003). In Press
21. Haralick, R., Shanmugam, K., Dinstein, I.: Textural features for image classification. IEEE Trans. Syst. Man Cybern. **SMC-3**(6), 610–621 (1973)
22. Kumar, A., Pang, G.: Defect detection in textured materials using Gabor filters. IEEE Trans. Ind. Appl. **38**(2), 425–440 (2002)
23. Randen, T., Husoy, J.: Filtering for texture classification: a comparative study. IEEE Trans. Pattern Anal. Mach. Intell. **21**(4), 291–310 (1999). doi:10.1109/34.761261
24. Gonzalez, R., Woods, R.: Digital Image Processing, 3rd edn. Prentice Hall, Pearson (2008)
25. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 1615–1630 (2005)
26. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 971–987 (2002)
27. Mu, Y., Yan, S., Liu, Y., Huang, T., Zhou, B.: Discriminative local binary patterns for human detection in personal album. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008), pp. 1–8 (2008)
28. Ojansivu, V., Heikkilä, J.: Blur insensitive texture classification using local phase quantization. In: Image and Signal Processing, pp. 236–243. Springer, Berlin (2008)
29. Kannala, J., Rahtu, E.: BSIF: Binarized statistical image features. In: 2012 21st International Conference on Pattern Recognition (ICPR), pp. 1363–1366. IEEE (2012)
30. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. Conf. Comput. Vis. Pattern Recognit. (CVPR2005) **1**, 886–893 (2005)
31. Lowe, D.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
32. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms. In: Proceedings of the International Conference on Multimedia, pp. 1469–1472. ACM (2010)
33. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: 9th European Conference on Computer Vision (ECCV2006). Graz Austria (2006)

34. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: binary robust independent elementary features. In: Computer Vision-ECCV 2010, pp. 778–792. Springer (2010)
35. Leutenegger, S., Chli, M., Siegwart, R.Y.: BRISK: binary robust invariant scalable keypoints. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 2548–2555. IEEE (2011)
36. Rubinstein, R., Zibulevsky, M., Elad, M.: Double sparsity: learning sparse dictionaries for sparse signal approximation. IEEE Trans. Signal Process. **58**(3), 1553–1564 (2010)
37. Olshausen, B., Field, D.: Sparse coding of sensory inputs. Curr. Opin. Neurobiol. **14**(4), 481–487 (2004)
38. Donoho, D., Elad, M.: Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. Proc. Natl. Acad. Sci. **100**(5), 2197–2202 (2003)
39. Yang, A., Gastpar, M., Bajcsy, R., Sastry, S.: Distributed sensor perception via sparse representation. Proc. IEEE **98**(6), 1077–1088 (2010)
40. Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T., Yan, S.: Sparse representation for computer vision and pattern recognition. Proc. IEEE **98**(6), 1031–1044 (2010)
41. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 1794–1801 (2009)
42. Olshausen, B., Field, D.: Sparse coding with an overcomplete basis set: a strategy employed by v1? Vis. Res. **37**(23), 3311–3325 (1997)
43. Rubinstein, R., Bruckstein, A., Elad, M.: Dictionaries for sparse representation modeling. Proc. IEEE **98**(6), 1045–1057 (2010)
44. Joliffe, I.: Principal Component Analysis. Springer, New York (1986)
45. Tosic, I., Frossard, P.: Dictionary learning. Signal Processing Magazine, IEEE **28**(2), 27–38 (2011)
46. Olshausen, B., Field, D.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature **381**(6583), 607–609 (1996)
47. Donoho, D.L.: For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution. Commun. Pure Appl. Math. **59**(6), 797–829 (2006)
48. Kreutz-Delgado, K., Murray, J., Rao, B., Engan, K., Lee, T., Sejnowski, T.: Dictionary learning algorithms for sparse representation. Neural Comput. **15**(2), 349–396 (2003)
49. Gorodnitsky, I., Rao, B.: Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm. IEEE Trans. Signal Process. **45**(3), 600–616 (1997)
50. Yaghoobi, M., Blumensath, T., Davies, M.: Dictionary learning for sparse approximations with the majorization method. IEEE Trans. Signal Process. **57**(6), 2178–2191 (2009)
51. Aharon, M., Elad, M., Bruckstein, A.: K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation. IEEE Trans. Signal Process. **54**(11), 4311–4322 (2006)
52. Tropp, J.: Greed is good: algorithmic results for sparse approximation. IEEE Trans. Inf. Theory **50**(10), 2231–2242 (2004)
53. Elad, M., Aharon, M.: Image denoising via sparse and redundant representations over learned dictionaries. IEEE Trans. Image Process. **15**(12), 3736–3745 (2006)
54. Mairal, J., Elad, M., Sapiro, G.: Sparse representation for color image restoration. IEEE Trans. Image Process. **17**(1), 53–69 (2008)
55. Mailhé, B., Lesage, S., Gribonval, R., Bimbot, F., Vandergheynst, P., et al.: Shift-invariant dictionary learning for sparse representations: extending K-SVD. Proc. Eur. Signal Process. Conf. **4** (2008)
56. Gribonval, R., Nielsen, M.: Sparse representations in unions of bases. IEEE Trans. Inf. Theory **49**(12), 3320–3325 (2003)
57. Hughes, G.: On the mean accuracy of statistical pattern recognizers. IEEE Trans. Inf. Theory **14**(1), 55–63 (1968)
58. Narendra, P.M., Fukunaga, K.: A branch and bound algorithm for feature subset selection. IEEE Trans. Comput. **C-26**(9), 917–922 (1977)
59. MathWorks: Matlab Toolbox of Bioinformatics: User's Guide. Mathworks Inc. (2007)

60. Wei, H.L., Billings, S.: Feature subset selection and ranking for data dimensionality reduction. IEEE Trans. Pattern Anal. Mach. Intell. **29**(1), 162–166 (2007). doi:10.1109/TPAMI.2007.250607

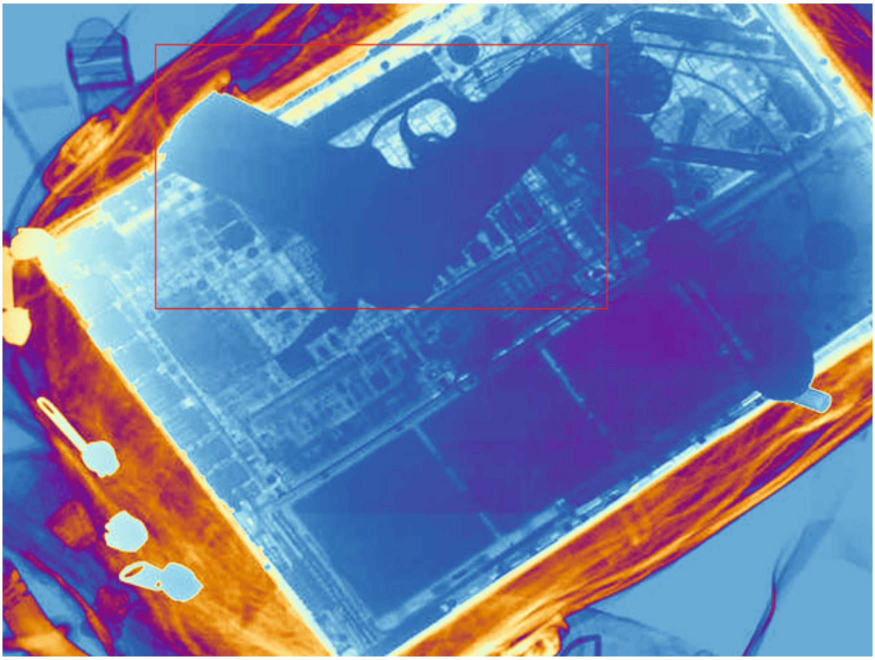
61. Mao, K.: Identifying critical variables of principal components for unsupervised feature selection. IEEE Trans. Syst. Man Cybern. Part B Cybern. **35**(2), 339–344 (2005). doi:10.1109/TSMCB.2004.843269

62. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley, New York (2001)

63. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Mach. Intell. **27**(8), 1226–1238 (2005)

Chapter 6
Classification in X-ray Testing

Abstract In this chapter we will cover known classifiers that can be used in X-ray testing. Several examples will be presented using Matlab. The reader can easily modify the proposed implementations in order to test different classification strategies. We will then present how to estimate the accuracy of a classifier using hold-out, cross-validation, and leave-one-out. Finally, we will present an example that involves all steps of a pattern recognition problem, i.e., feature extraction, feature selection, classifier's design, and evaluation. We will thus propose a general framework to design a computer vision system in order to select—automatically—from a large set of features and a bank of classifiers, those features and classifiers that can achieve the highest performance.



Cover image: *Ideal detection of a handgun superimposed onto a laptop (X-ray image* `B0019_0001` *colored with 'sinmap').*

D. Mery, *Computer Vision for X-Ray Testing*, DOI 10.1007/978-3-319-20747-6_6

6.1 Introduction

Considerable research efforts in computer vision applied to industrial applications have been developed in recent decades. Many of them have been concentrated on using or developing tailored methods based on visual features that are able to solve a specific task. Nevertheless, today's computer capabilities are giving us new ways to solve complex computer vision problems. In particular, a new paradigm on machine learning techniques has emerged posing the task of recognizing visual patterns as a search problem based on training data and a hypothesis space composed of visual features and suitable classifiers. Furthermore, now we are able to extract, process, and test in the same time more image features and classifiers than before. In our book, we propose a general framework that designs a computer vision system automatically, i.e., it finds—without human interaction—the features and the classifiers for a given application avoiding the classical trial and error framework commonly used by human designers. The key idea of the proposed framework is to design a computer vision system as shown in Fig. 6.1 in order to select—automatically—from a large set of features and a bank of classifiers, those features and classifiers that achieve the highest performance.

Whereas Chap. 5 covered feature extraction and selection, the focus of this chapter will be the classification. Once the proper features are selected, a classifier can be designed. Typically, the classifier assigns a feature vector **x** with n features $(x_1 \ldots x_n)$ to one class. In case of defects detection, for example, there are two

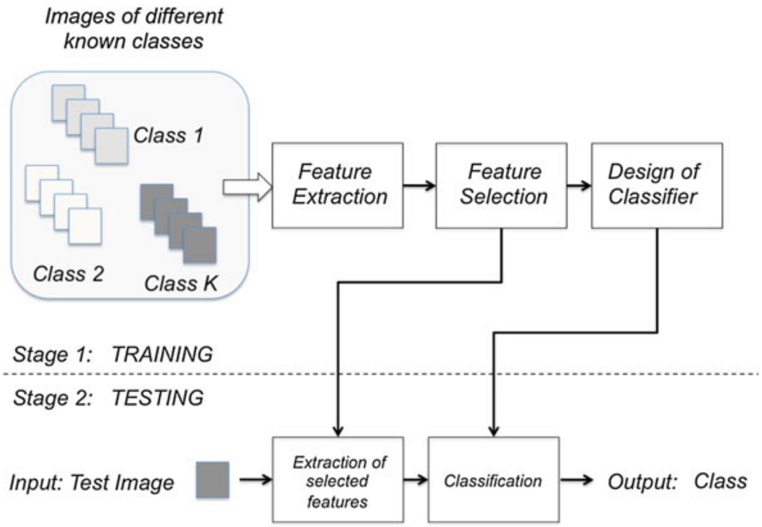


Fig. 6.1 Supervised pattern recognition schema. In the training stage, features are extracted and selected (see Chap. 5). In addition, a classifier is designed. In the testing stage, selected features are extracted and the test image is classified

classes: *flaws* or *no-flaws*. In case of baggage screening, there can be more classes: *knives*, *handguns*, *razor blades*, etc. In pattern recognition, classification can be performed using the concept of similarity: patterns that are *similar* are assigned to the same class [1]. Although this approach is very simple, a good metric defining the similarity must be established. Using representative samples, we can make a supervised classification finding a discriminant function $h(\mathbf{x})$ that provides us information on how similar a feature vector $\mathbf{x}$ is to a class representation.

In this chapter, we will cover many known classifiers (such as linear discriminant analysis, Bayes, support vector machines, neural networks among others). Several examples will be presented using Matlab. The reader can easily modify the proposed implementations in order to test different classification strategies. Afterwards, we present how to estimate the accuracy of a classifier using hold-out, cross-validation, and leave-one-out. The well-known confusion matrix and receiver-operation-characteristic curve will be outlined as well. We will explain by detailing the advantages and disadvantages of each one. Finally, we will present an example that involves all steps of a pattern recognition problem, i.e., feature extraction, feature selection, classifier's design, and evaluation.

6.2 Classifiers

In this section, the most relevant classifiers are explained with several examples.

6.2.1 Minimal Distance

A very simple classifier is based on the concept of 'minimal distance'. In this classifier, each class is represented by its center of mass that can be viewed as a template [2]. Thus, a mean value $\bar{\mathbf{x}}_k$ of each class is calculated on the training data:

$$\bar{\mathbf{x}}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathbf{x}_{jk} \tag{6.1}$$

where $\mathbf{x}_{jk}$ is the jth sample of class ω_k of the training data, and N_k is the number of samples of the kth class. A test sample $\mathbf{x}$ is assigned to class ω_k if the Euclidean distance $\| \mathbf{x} - \bar{\mathbf{x}}_k \|$ is minimal. Formerly,

$$h_{\text{dmin}}(\mathbf{x}) = \underset{k}{\text{argmin}} \{\| \mathbf{x} - \bar{\mathbf{x}}_k \|\}. \tag{6.2}$$

A useful formulation is defining the distance function $d_{\text{dmin}}(\mathbf{x}, k) = \| \mathbf{x} - \bar{\mathbf{x}}_k \|$. Thus, we can write (6.2) as

$$h_{\text{dmin}}(\mathbf{x}) = \underset{k}{\text{argmin}} \{d_{\text{dmin}}(\mathbf{x}, k)\}. \tag{6.3}$$

This formulation based on minimal distances will be used in the following sections. In $\mathbb{X}$vis Toolbox, this classifier is implemented in function Xdmin (see Appendix B).

Matlab Example 6.1 In this example, we show how to train and test a classifier based on Euclidean minimal distance. We use data that was simulated using a mixture of Gaussian distributions. The data consists of 800 samples for training and 400 samples for testing purposes. Each sample has two features x_1 and x_2 and it belongs to class ω_1 or ω_0. Figure 6.2 shows the feature spaces for training and testing.

Listing 6.1 : Classification using Euclidean minimal distance

```
% DminExample.m
close all
load datasim                      % simulated data (2 classes, 2 features)
                                  % Xtrain,dtrain features and labels for training
                                  % Xtest, dtest  features and labels for testing
subplot(1,2,1);
Xplotfeatures(Xtrain,dtrain)      % plot feature space for training data
subplot(1,2,2);
Xplotfeatures(Xtest,dtest)        % plot feature space for testing data
opc = [];                         % options of the classifier
par = Xdmin(Xtrain,dtrain,opc);   % Euclidean distance classifier - training
ds  = Xdmin(Xtest,par);           % Euclidean distance classifier - testing
                                  % ds = predicted labels for testing data
acc = Xaccuracy(ds,dtest)         % accuracy on test data
```

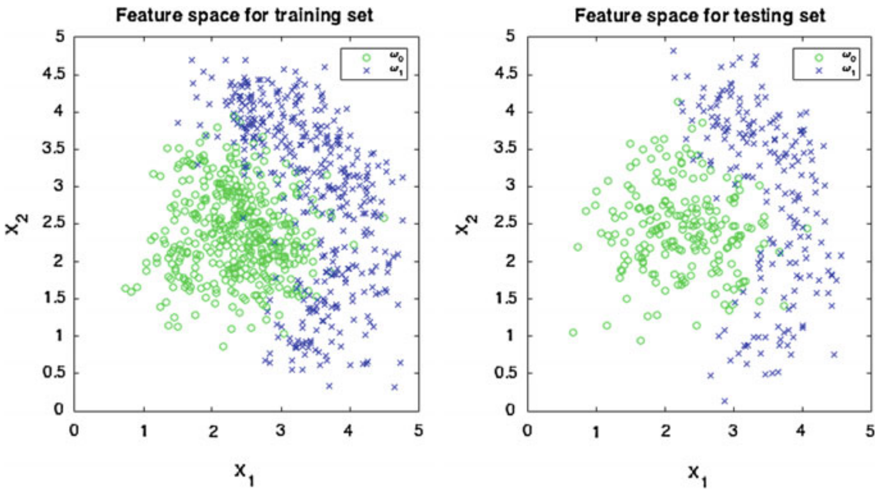


Fig. 6.2 Simulated data that is used in Sect. 6.2 (→ Example 6.1)

The output of this code is `acc`, i.e., the accuracy obtained by classifying the testing data. In this case, it is only 85.5 %. The low performance of this classifier is because the decision line is a straight line. The reader can imagine that the decision line can be computed in three steps: (i) Compute the centers of mass of each class distribution in the training set as $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_0$ according to (6.1). (ii) Compute ℓ_C the straight-line that contains both centers of mass. (iii) Compute the decision line ℓ as the line that is perpendicular to ℓ_C and equidistant to $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_0$. The decision line is shown in Fig. 6.9. Obviously, the straight line is not able to separate these curved distributions.

We use Xdmin (see Appendix B) for training and again Xdmin for testing. In training, the arguments are `Xtrain` (for the features of the training samples), `dtrain` the corresponding labels and `opc` the options of the classifier (in this case the classifier has no options). The output of the training is stored in `par` containing all parameters of the learned classifier, i.e., for this classifier the centers of mass only. In testing, the arguments are `Xtest` (for the features of the testing samples) and `par` (the parameters of the classifier). The output is `ds` the predicted labels of the testing data. In order to compute the performance, we compute the accuracy using command Xaccuracy (see Appendix B) of $\mathbb{X}$VIS Toolbox. This is defined as the ratio of number of samples correctly classified to the total number of samples.

It is important to know that we could compute training and testing in just one step using `ds = Xdmin(Xtrain,dtrain,Xtest,opc);`. □

6.2.2 Mahalanobis Distance

The Mahalanobis classifier employs the same concept as minimal distance (see Sect. 6.2.1), however, it uses a distance metric based on the 'Mahalanobis distance', in which, by means of the covariance matrix, the features to be evaluated are weighted according to their variances. A test sample $\mathbf{x}$ is assigned to class ω_k if the Mahalanobis distance of $\mathbf{x}$ to class ω_k, denoted as $d_{\text{maha}}(\mathbf{x}, k)$, is minimal. The Mahalanobis distance is defined as

$$d_{\text{maha}}(\mathbf{x}, k) = (\mathbf{x} - \bar{\mathbf{x}}_k)^{\mathsf{T}} \mathbf{C}_k^{-1}, (\mathbf{x} - \bar{\mathbf{x}}_k) \tag{6.4}$$

where $\mathbf{C}_k$ is the covariance matrix of the kth class. It can be estimated as

$$\mathbf{C}_k = \frac{1}{N_k - 1} \sum_{j=1}^{N_k} (\mathbf{x}_{kj} - \bar{\mathbf{x}}_k)(\mathbf{x}_{kj} - \bar{\mathbf{x}}_k)^{\mathsf{T}}, \tag{6.5}$$

where $\mathbf{x}_{jk}$ is the jth sample of class ω_k of the training data, and N_k is the number of samples of the kth class. Some examples are illustrated in Fig. 6.3. Formerly,

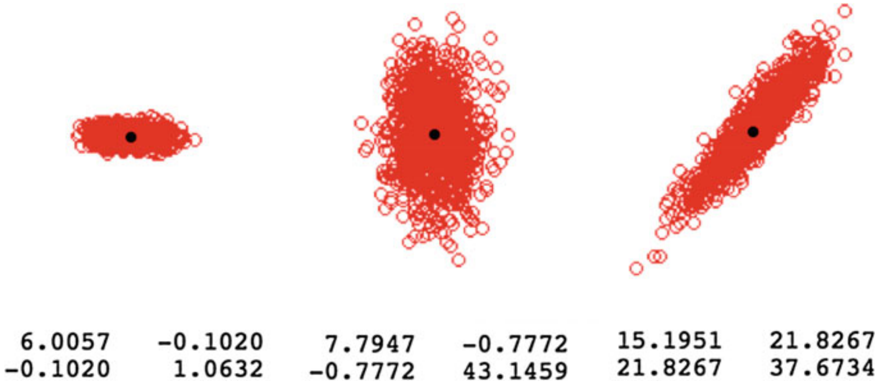


Fig. 6.3 Examples of three different Gaussian distributions $p(\mathbf{x}|\omega_k)$ in 2D. The *black point* represents the mean μ_k and the 2×2 matrices the covariances Σ_k

$$h_{\text{maha}}(\mathbf{x}) = \underset{k}{\text{argmin}}\,\{d_{\text{maha}}(\mathbf{x}, k)\}, \tag{6.6}$$

where distance d_{maha} is defined in (6.4). In $\mathbb{X}$vis Toolbox, Mahalanobis classifier is implemented in function Xmaha (see Appendix B). An example of this classifier is presented in Example 6.3.

6.2.3 Bayes

In Bayes classifier the idea is to assign the test sample $\mathbf{x}$ to the most *probable* class. For this purpose, we use the conditional probability $p(\omega_k|\mathbf{x})$, that gives the probability of class ω_k occurs given sample $\mathbf{x}$. Thus, if $p(\omega_k|\mathbf{x})$ is maximal the $\mathbf{x}$ is assigned to class ω_k:

$$h_{\text{Bayes}}(\mathbf{x}) = \underset{k}{\text{argmax}}\,\{p(\omega_k|\mathbf{x})\}, \tag{6.7}$$

Using Bayes theorem we can write the conditional probability as

$$p(\omega_k|\mathbf{x}) = p(\omega_k)\frac{p(\mathbf{x}|\omega_k)}{p(\mathbf{x})}, \tag{6.8}$$

where $p(\omega_k|\mathbf{x})$ is known as 'posterior', $p(\omega_k)$ as 'prior', $p(\mathbf{x}|\omega_k)$ as 'likelihood' and $p(\mathbf{x})$ as 'evidence'. Since $p(\mathbf{x})$ is the same by evaluating $p(\omega_k|\mathbf{x})$ for all k we can rewrite (6.7) as follows:

$$h_{\text{Bayes}}(\mathbf{x}) = \underset{k}{\text{argmax}}\,\{p(\mathbf{x}|\omega_k)p(\omega_k)\}. \tag{6.9}$$

In order to evaluate (6.9) properly, we need good estimations for $p(\mathbf{x}|\omega_k)$ and $p(\omega_k)$. There are several known approaches to estimate these, some of which will be covered in the following sections under the assumption of Gaussian distributions of the classes (see Sects. 6.2.4 and 6.2.5).

The prior $p(\omega_k)$ can be estimated by the number of available samples in the training dataset of each class. Thus, $p(\omega_k) = N_k/N$, where N_k is the number of samples that belong to class ω_k and $N = \sum_k N_k$ the total number of samples. Nevertheless, in many cases of X-ray testing the available samples are not balanced, e.g., in defect detection problems there are a reduced number of flaws in comparison with the large number of non-flaws [3]. If we use the estimation $p(\omega_k) = N_k/N$ then the most important class to be detected will have a very low prior, and it will be difficult to detect. In such cases, the prior must be considerably increased in order to be the more probable.

In order to estimate $p(\mathbf{x}|\omega_k)$, we can use an approach based on Kernel Density Estimation (KDE) [4]:

$$\hat{p}(\mathbf{x}|\omega_k) = \alpha_k \sum_{j=1}^{N_k} K\left(\frac{\mathbf{x} - \mathbf{x}_{jk}}{\Delta}\right) \tag{6.10}$$

where K is a kernel function such as a Gaussian, that has a mean zero and variance of one, Δ is the bandwidth and α_k is a normalization factor equal to $1/(N_k\Delta)$. Since $K(\mathbf{x}/\Delta)$ integrates to Δ, with this normalization factor we ensure that $\hat{p}(\mathbf{x}|\omega_k)$ integrates to one. Example of KDE can be found in Fig. 5.21 that were estimated using the training data of Fig. 5.23. In 𝕏vis Toolbox, command Xbayes (see Appendix B) is implemented using KDE.

Matlab Example 6.2 In this example, we show how to train and test a Bayes classifier using Kernel Density Estimation. We use the same simulated data addressed in Example 6.1 and illustrated in Fig. 6.2.

Listing 6.2 : Classification using Bayes

```
% BayesExample.m
close all; clear opc
load datasim  % simulated data (2 classes, 2 features)
              % Xtrain,dtrain features and labels for training
              % Xtest, dtest  features and labels for testing
Xplotfeatures(Xtrain,dtrain)         % plot feature space
op.p = [0.5 0.5];                    % prior probability
op.show = 1;                         % display results
ds = Xbayes(Xtrain,dtrain,Xtest,op); % Bayes classifier
acc = Xaccuracy(ds,dtest)            % accuracy on test data
```

The output of this code is Fig. 6.4 for the Kernel Density Estimation of each class, and `acc`, the accuracy obtained by classifying the testing data. In this case, we obtain 94.5 %. The reader can compare this result with the accuracy obtained by classifier of Example 6.1. It is clear that Bayes classifier can properly model the curved distributions. The decision lines are shown in Fig. 6.9. In this example we use command Xbayes (see Appendix B) of 𝕏vis Toolbox. □

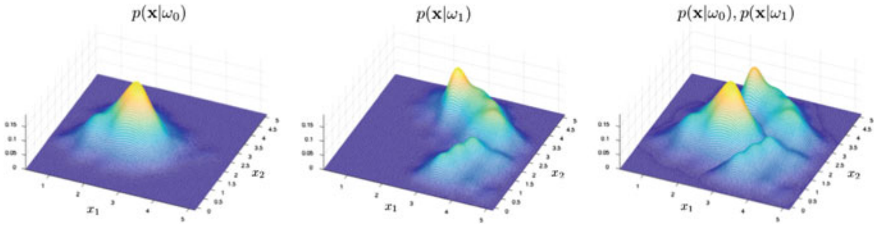


Fig. 6.4 Estimation of $p(\mathbf{x}|\omega_k)$ using Kernel Density Estimation (KDE) for distributions of the training set of Fig. 6.2 (→ Example 6.2)

6.2.4 Linear Discriminant Analysis

For Gaussian distributions with $\mathbf{x} \in \mathbb{R}^n$:

$$p(\mathbf{x}|\omega_k) = \frac{1}{(2\pi)^{n/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^{\mathsf{T}}\Sigma_k^{-1}(\mathbf{x}-\mu_k)\right\}, \tag{6.11}$$

where a good estimation for center of mass μ_k and covariance Σ_k of class ω_k can be taken from (6.1) and (6.5) respectively. Since the logarithm is a monotonically increasing function $\text{argmax}_k\,\{p\} = \text{argmax}_k\,\{\log(p)\}$. Thus, (6.9) can be written as

$$h(\mathbf{x}) = \underset{k}{\text{argmax}}\,\{\log\{p(\mathbf{x}|\omega_k)p(\omega_k)\}\}. \tag{6.12}$$

Using some manipulation,

$$\log\{p(\mathbf{x}|\omega_k)p(\omega_k)\} = \log\{p(\mathbf{x}|\omega_k)\} + \log\{p(\omega_k)\} \tag{6.13}$$

$$= \underbrace{-\frac{1}{2}(\mathbf{x}-\mu_k)^{\mathsf{T}}\Sigma_k^{-1}(\mathbf{x}-\mu_k)}_{\textcircled{1}} \underbrace{-\frac{1}{2}\log(|\Sigma_k|)}_{\textcircled{2}} \underbrace{-\frac{n}{2}\log(2\pi)}_{\textcircled{3}} \underbrace{+\log(p(\omega_k))}_{\textcircled{4}} \tag{6.14}$$

It is clear that we do not need to evaluate $\textcircled{3}$ because this term is constant and the location of the maximum does not change.

In Linear Discriminant Analysis (LDA) [5], we assume $\Sigma_k = \Sigma$ (constant) for all k, i.e., term $\textcircled{2}$ in (6.14) is constant as well, and it is not necessary to be evaluated. Consequently,

$$\log\{p(\mathbf{x}|\omega_k)p(\omega_k)\} = \underbrace{-\frac{1}{2}(\mathbf{x}-\mu_k)^{\mathsf{T}}\Sigma^{-1}(\mathbf{x}-\mu_k) + \log(p(\omega_k))}_{-d_{\text{LDA}}(\mathbf{x},k)} + C \tag{6.15}$$

where constant C corresponds to terms ② + ③. Covariance matrix Σ can be computed from training data. A good estimation is the average of the individual covariance matrices $\Sigma = \frac{1}{K}\sum_k \mathbf{C}_k$. Formerly, the LDA classifier is defined as follows:

$$h_{\mathrm{LDA}}(\mathbf{x}) = \underset{k}{\operatorname{argmin}}\,\{d_{\mathrm{LDA}}(\mathbf{x}, k)\}, \tag{6.16}$$

where $d_{\mathrm{LDA}}(\mathbf{x}, k)$ is defined in (6.15). In $\mathbb{X}$vis Toolbox, the LDA classifier is implemented in function Xlda (see Appendix B). An example of this classifier is presented in Example 6.3.

A variant of Mahalanobis classifier is obtained by assuming that not only Σ_k is constant, but also $p(\omega_k)$ is constant. Thus, $\Sigma_k = \Sigma$ and $p(\omega_k) = p_c$ for all k. This means that in (6.14) terms ④ is constant as well:

$$\log\{p(\mathbf{x}|\omega_k)p(\omega_k)\} = \underbrace{-\frac{1}{2}(\mathbf{x}-\mu_k)^{\mathsf{T}}\Sigma^{-1}(\mathbf{x}-\mu_k)}_{-d_{\mathrm{maha}}(\mathbf{x},k)} + C \tag{6.17}$$

where constant C corresponds to terms ②+③+④. The classification is performed by (6.6) where $d_{\mathrm{maha}}(\mathbf{x}, k)$ is defined in (6.17). The reader can observe that if we assume that $\Sigma = \mathbf{I}$ we obtain the Minimal Distance classifier (6.3).

6.2.5 Quadratic Discriminant Analysis

In Quadratic Discriminant Analysis (QDA) [5], we assume that Σ_k and $p(\omega_k)$ are not constant for all k, i.e., in (6.14) only term ③ is constant:

$$\log\{p(\mathbf{x}|\omega_k)p(\omega_k)\} = \underbrace{-\frac{1}{2}(\mathbf{x}-\mu_k)^{\mathsf{T}}\Sigma^{-1}(\mathbf{x}-\mu_k) - \frac{1}{2}\log(|\Sigma_k|) + \log(p(\omega_k))}_{-d_{\mathrm{QDA}}(\mathbf{x},k)} + C, \tag{6.18}$$

where constant C corresponds to terms ③. Formerly,

$$h_{\mathrm{QDA}}(\mathbf{x}) = \underset{k}{\operatorname{argmin}}\,\left\{d_{\mathrm{QDA}}(\mathbf{x}, k)\right\}, \tag{6.19}$$

where $d_{\mathrm{QDA}}(\mathbf{x}, k)$ is defined in (6.18). In $\mathbb{X}$vis Toolbox, QDA classifier is implemented in function Xqda (see Appendix B).

Matlab Example 6.3 In this example, we show how to train and test three different classifiers: Mahalanobis (see Sect. 6.2.2), LDA (see Sect. 6.2.4) and QDA (see Sect. 6.2.5). We use the same simulated data addressed in Example 6.1 and illustrated in Fig. 6.2.

Listing 6.3 : Classification using Mahalanobis, LDA and QDA

```
% SimpleClassifiersExample.m
close all; clear opc;
load datasim  % simulated data (2 classes, 2 features)
              % Xtrain,dtrain features and labels for training
              % Xtest, dtest  features and labels for testing
pc = [0.5 0.5];                              % priors for each class
opc(1).name = 'maha';opc(1).options.p = pc;  % Mahalanobis distance
opc(2).name = 'lda'; opc(2).options.p = pc;  % LDA
opc(3).name = 'qda'; opc(3).options.p = pc;  % QDA
ds = Xclassify(Xtrain,dtrain,Xtest,opc);     % Training and Testing
acc = Xaccuracy(ds,dtest);                   % Accuracy on test data
for i=1:3                                    % Output
    fprintf('%8s: %5.2f%%\n',opc(i).name,100*acc(i));
end
```

The output of this code is `acc`, the accuracy obtained by classifying the testing data. In this case, we obtain 89.00, 87.00 and 92.75 % for Mahalanobis, LDA and QDA respectively. It is clear that Mahalanobis and QDA achieve better performance than LDA because they can model the curved distributions. The decision lines are shown in Fig. 6.9. In this example we use command Xclassify (see Appendix B) of 𝕏vis Toolbox which is able to train and test several classifiers simultaneously. The reader can try to use commands Xmaha, Xlda and Xqda (see Appendix B) independently as explained for Xdmin (see Appendix B) in Example 6.1. □

6.2.6 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a nonparametric approach, in which the K most similar training samples to a given test feature vector $\mathbf{x}$ are determined [5]. The assigned class is the most frequent class from those K samples [6]. In other words, we find—in the training set—the K nearest neighbors of $\mathbf{x}$ and we evaluate the majority vote of their classes:

$$h_{\text{knn}}(\mathbf{x}) = \text{mode}(y(\mathbf{x}^1), \ldots y(\mathbf{x}^K)), \tag{6.20}$$

where $\{\mathbf{x}^i\}_{i=1}^K$ are the K nearest neighbors of $\mathbf{x}$, and $y(\mathbf{x}^i)$ the labeled class of $(\mathbf{x}^i)$.

KNN can be implemented (avoiding the exhaustive search of all samples of the training set) by a search using a $k-$d tree structure [7] to search the nearest neighbors. In 𝕏vis Toolbox, KNN classifier is implemented in function Xknn (see Appendix B).

6.2.7 Neural Networks

Artificial neuronal networks are mathematical tools derived from what is known about the mechanisms and physical structure of biological learning, based on the function of a neuron. They are parallel structures for the distributed processing of information [8]. A neural network consists of artificial neurons connected in a network that is able to classify a test feature vector $\mathbf{x}$ evaluating a linear or nonlinear weighed sum of functions. The weights, the functions, and the connections are estimated in a training phase by minimizing the classification error [8, 9].

The basic processing unit is the neuron, made up of multiple inputs and only one output. This output is determined by an activation function that operates on input values, and a transfer function that operates on the activation value. In other words, if we consider the input vector $\mathbf{x} = [x_1 \dots x_n]^\mathsf{T}$, the weight vector $\mathbf{w} = [w_1 \dots w_n]^\mathsf{T}$, the activation value a and the output value of the neuron y, the values of a and y can be described by:

$$a = \mathbf{w}^\mathsf{T}\mathbf{x} + b \qquad y = f(a) \tag{6.21}$$

where b is the bias and $f(a)$ is the so-called transfer function and is generally a linear function or a sigmoid such as

$$f(a) = \frac{1}{1 + e^{-a}} \quad \text{or} \quad f(a) = \frac{\tanh(a) + 1}{2}. \tag{6.22}$$

The structure of a neuronal network can have one or more neurons and depending on the type of problem and the training, these networks receive different names. They have the capacity to associate and classify patterns, compress data, perform process control, and approximate nonlinear functions [10].

The most often used type of neural network in classification is the Multi Layer Perceptron (MLP) which consists of sequential layers of neurons. The structure of an MLP is shown in Fig. 6.5 where each neuron has Eq. (6.21) associated to it.

Backpropagation is the learning algorithm normally used to train this type of network. Its goal is to minimize the error function constructed from the difference between the desired and the modeled output.

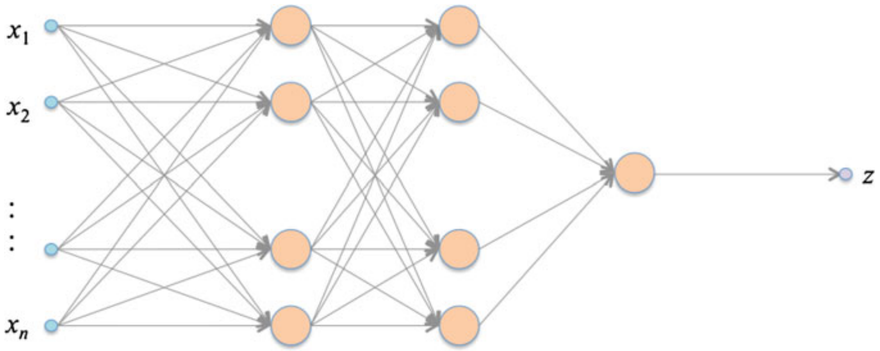


Fig. 6.5 Multi Layer Perceptron (MLP) with two hidden layers

The initially developed backpropagation algorithm used a steepest descent first-order method as the learning rule. Nonetheless, other more powerful second-order methods are in common use today, such as the conjugate gradient, which consists of finding the gradient directions that satisfy:

$$\mathbf{d}^{(t+1)^{\mathsf{T}}}\mathbf{H}\mathbf{d}^{(t)} = 0 \tag{6.23}$$

where **d** is the slope direction and **H** is the Hessian matrix evaluated at point $\mathbf{w}^{t+1}$. Vector **w** is the network weight vector and is updated by means of:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \lambda^{(t)}\mathbf{d}^{(t)} \tag{6.24}$$

Parameter $\lambda^{(t)}$ is selected for minimizing:

$$E(\lambda) = E(\mathbf{w}^{(t)} + \lambda\mathbf{d}^{(t)}) \tag{6.25}$$

where E is the error function [8], computes as the mean squared error between target value and the value computed by the perceptron. Depending on the number of hidden layers we will have structures such as the diagram presented in Fig. 6.5. The output, called z is a real number between 0 and 1. The classification, for the two-class problem of our test sample **x** will be ω_1 (or ω_0) if z is greater (or less) than 0.5.

In $\mathbb{X}$VIS Toolbox there are two classifiers based on neural networks: Xmlp (see Appendix B) for Multi Layer Perceptron and Xglm (see Appendix B) that uses a Generalized Linear Model [11]. In addition, Xpnn (see Appendix B) for probabilistic neural networks is also available [12].

6.2.8 Support Vector Machines

The original Support Vector Machines (SVM) find a decision line that separates two classes (ω_1 and ω_0) as illustrated in Fig. 6.6a. In this example, we can see that there are many possible decision lines like ℓ_1, ℓ_2 and ℓ_3 among others (see Fig. 6.6b). A relevant question arises: which decision line ℓ can separate both classes at 'best'? In SVM strategy, we define the 'margins' b_1 and b_0 as the minimal distance from the decision line to a sample of class ω_1 and ω_0 respectively. After SVM criterion, the 'best' separation line ℓ_{SVM} is one that (i) it is in the middle, i.e., $b_1 = b_0 = b$, and (ii) its margin is maximal, i.e., $b = b_{\max}$. Thus, decision line ℓ_{SVM} is equidistant to the margin lines and the margin is maximal.

In $\mathbb{R}^2$ we have a decision line, however, in general, in $\mathbb{R}^n$, we have a hyperplane that is defined as

$$\ell_{\text{SVM}} : \; g(\mathbf{x}) = \mathbf{a}^{\mathsf{T}}\mathbf{x} + a_0 = 0 \tag{6.26}$$

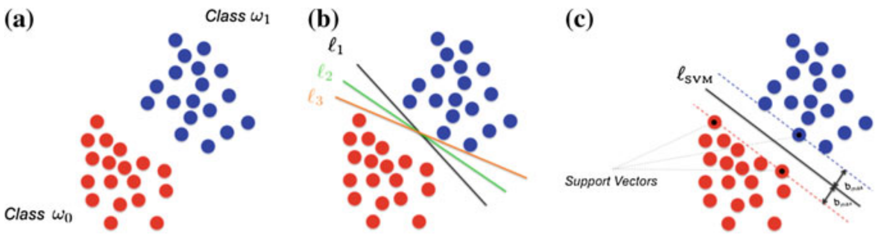


Fig. 6.6 Key idea of support vector machine: **a** Given a two-class problem, find a decision line ℓ. **b** There are many possible decision lines that can separate both classes. **c** In SVM, we search decision line ℓ_{SVM} so that the margin b is maximized. The support vectors are defined as those samples that belong to the margin lines

where $\mathbf{x} = [x_1 \dots x_n]^{\mathsf{T}}$ is our feature vector and $\mathbf{a} = [a_1 \dots a_n]^{\mathsf{T}}$ and a_0 are the linear parameters to be estimated. The solution for $\{a_j\}_{i=0}^{n}$ can be found following an optimization approach [13]. In the solution, $\{a_j\}_{i=0}^{n}$ depends only on the *support vectors*, i.e., the samples of both classes that belong to the margin lines as shown in Fig. 6.6c. The solution of this optimization problem consists of parameter values λ_i corresponding to ith support vector:

$$\mathbf{a} = \sum_{i=1}^{m} \lambda_i z_i \mathbf{x}_i, \tag{6.27}$$

for m support vectors, where $z_i = \pm 1$ if $\mathbf{x}_i$ belongs to ω_1 and ω_0 respectively. In addition, a_0 can be calculated from any support vector as $a_0 = z_i - \mathbf{a}^{\mathsf{T}}\mathbf{x}_i$ [5]. In SVM, the classification of a test sample $\mathbf{x}$ can be formulated as follows:

$$h_{\text{SVM}}(\mathbf{x}) = \begin{cases} 1 \text{ if} & \mathbf{a}^{\mathsf{T}}\mathbf{x} + a_0 > 0 \\ 0 \text{ otherwise} & \end{cases}. \tag{6.28}$$

In practice, however, there is some overlapping between the classes as shown in Fig. 6.7a. If we have a decision line that separates the feature space, we will have misclassified samples. In SVM strategy, we consider only the misclassified samples as illustrated in Fig. 6.7b. They will be the *support vectors*. The ith support vector has a distance e_i to the decision line that corresponds to an error (see Fig. 6.7c). After SVM criterion, the 'best' decision line ℓ_{SVM} is one that minimizes the total error $e = \sum_i e_i$. Again, the solution for $\{a_i\}_{i=0}^{n}$ depends only on the support vectors, and they can be estimated using an optimization approach [13]. The classification is performed according to (6.28).

The previous approach estimates a straight-line decision boundary in feature space. In many cases, however, it is convenient to find a curve that separates the classes as illustrated in Fig. 6.8a. In order to use SVM linear classification, the feature space can be transformed into a new enlarged feature space (Fig. 6.8b) where the classification boundary can be linear. Thus, as shown in Fig. 6.8c, a simple linear classification (6.28) can be designed in the transformed feature space in order to separate both classes [13].

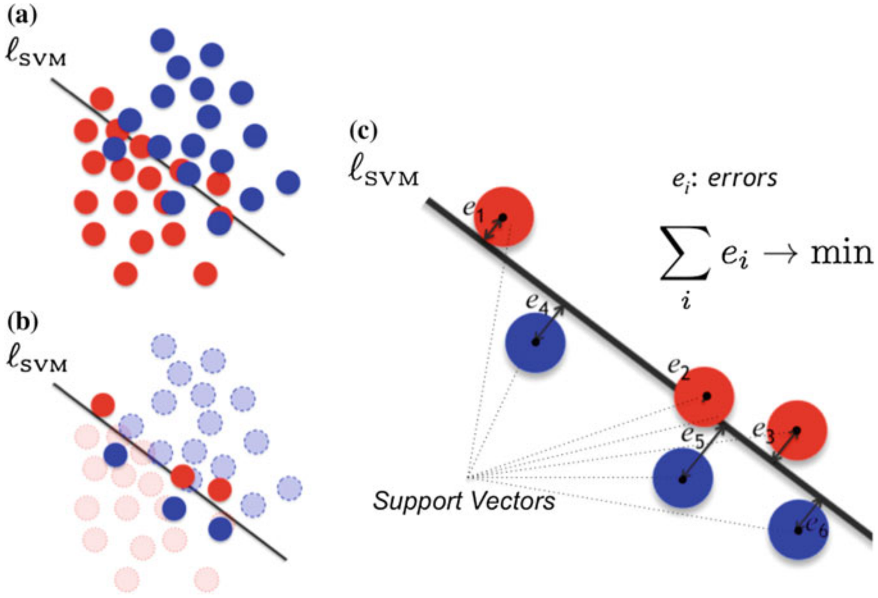


Fig. 6.7 Key idea of support vector machine with overlapping: **a** Given a two-class problem with overlapping, find a decision line ℓ_{SVM}. **b** By choosing a decision line ℓ_{SVM} there will be misclassified samples. **c** The misclassified samples are the support vectors. Each of them has an error e_i defined as the perpendicular distance to the decision line ℓ_{SVM}. In SVM, we search decision line ℓ_{SVM} so that the total error $\sum e_i$ is minimized

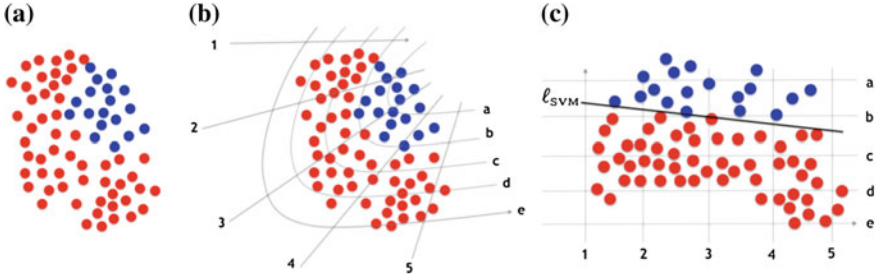


Fig. 6.8 Nonlinear decision line. **a** Feature space with two classes that can be separated using a curve. **b** The feature space can be described in a new coordinate system. **c** Transformed coordinate system in which a linear decision line can be used

The original feature space is transformed using a function $f(\mathbf{x})$. Thus, according to (6.26) and (6.27) we obtain:

$$\begin{aligned} g(f(\mathbf{x})) &= \mathbf{a}^{\mathsf{T}} f(\mathbf{x}) + a_0 \\ &= \textstyle\sum_i \lambda_i z_i \langle f(\mathbf{x}_i), f(\mathbf{x})\rangle + a_0 \end{aligned} \tag{6.29}$$

Table 6.1 Kernel functions used by SVM

Name	$K(\mathbf{x}_i, \mathbf{x})$
Linear	$\langle \mathbf{x}_i, \mathbf{x} \rangle$
qth degree polynomial	$(1 + \langle \mathbf{x}_i, \mathbf{x} \rangle)^q$
Radial basis (RBF)	$\exp(-\gamma \|\|\mathbf{x}_i - \mathbf{x}\|\|^2)$
Sigmoid	$\tanh(\alpha_1 \langle \mathbf{x}_i, \mathbf{x} \rangle + \alpha_2)$

where $\langle f(\mathbf{x}_i), f(\mathbf{x}) \rangle$ is the inner product $[f(\mathbf{x}_i)]^\mathsf{T} f(\mathbf{x})$. In (6.29), we can observe that for the classification, only the kernel function $\langle f(\mathbf{x}_i), f(\mathbf{x}) \rangle = K(\mathbf{x}_i, \mathbf{x})$ that computes inner products in the transformed space is required. Consequently, using (6.29) we can write (6.28) in general as

$$h_{\text{SVM}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum_i \lambda_i z_i K(\mathbf{x}_i, \mathbf{x}) + a_0 > 0 \\ 0 & \text{otherwise} \end{cases} . \tag{6.30}$$

Table 6.1 shows typical kernel functions that are used by SVM classifiers. They should be a symmetric positive (semi-) definite function [5]. In 𝕏vis Toolbox, SVM classifier is implemented in function Xsvm (see Appendix B).

Matlab Example 6.4 In this example, we show how to train many classifiers together. We use the same simulated data addressed in Example 6.1 and illustrated in Fig. 6.2.

Listing 6.4 : Classification using several classifiers

```matlab
% ManyClassifiersExample.m
close all; clear opc
load datasim  % simulated data (2 classes, 2 features)
              % Xtrain,dtrain features and labels for training
              % Xtest, dtest  features and labels for testing
k = 0;
k=k+1;opc(k).name = 'dmin';  opc(k).options.p = [];          % Euclidean distance
k=k+1;opc(k).name = 'maha';  opc(k).options.p = [];          % Mahalanobis distance
k=k+1;opc(k).name = 'bayes'; opc(k).options.p = [];          % Bayes
k=k+1;opc(k).name = 'lda';   opc(k).options.p = [];          % LDA
k=k+1;opc(k).name = 'qda';   opc(k).options.p = [];          % QDA
k=k+1;opc(k).name = 'knn';   opc(k).options.k = 5;           % KNN with 5 neighbors
k=k+1;opc(k).name = 'knn';   opc(k).options.k = 15;          % KNN with 5 neighbors
k=k+1;opc(k).name = 'glm';   opc(k).options.method = 2; opc(k).options.iter = 12;   % GLM
k=k+1;opc(k).name = 'mlp';   opc(k).options.method = 2; opc(k).options.alpha = 0.2; % MLP
      opc(k).options.iter=12;opc(k).options.nhidden = 6;opc(k).options.ncycles = 60;
k=k+1;opc(k).name = 'svm';   opc(k).options.kernel = '-t 0';  % linear-SVM
k=k+1;opc(k).name = 'svm';   opc(k).options.kernel = '-t 1';  % polynomial-SVM
k=k+1;opc(k).name = 'svm';   opc(k).options.kernel = '-t 2';  % rbf-SVM
par = Xclassify(Xtrain,dtrain,opc);                            % Training
ds  = Xclassify(Xtest,par);                                    % Testing
Xdecisionline(Xtrain,dtrain,Xn,par);                           % Draw decision lines
acc = Xaccuracy(ds,dtest);                                     % Accuracy on test data
for i=1:k                                                      % Output
    fprintf('%8s: %5.2f%%\n',par(i).options.string,100*acc(i));
end
```

The output of this code is acc, the accuracy obtained by classifying the testing data (see Table 6.2). In this example, there are many classifiers that obtain more

Table 6.2 Accuracy of classifiers of Example 6.4

Classifier	Accuracy (%)
dmin	85.50
maha	89.00
bayes	94.50
lda	87.00
qda	92.75
knn, 5	93.75
knn, 15	93.75
glm	86.75
mlp	94.50
svm-lin 0	86.75
svm-poly 1	94.50
svm-rbf 2	94.50

than 94.00 % because they can model the curved distributions. On the other hand, we observe that the linear classifiers cannot achieve more than 87.0 %. The decision lines are shown in Fig. 6.9. In this example, we use command Xclassify (see Appendix B) of 𝕏vis Toolbox which is able to train and test several classifiers simultaneously, and command Xdecisionline (see Appendix B) to plot the decision lines of each classifier as shown in Fig. 6.9. □

6.2.9 Classification Using Sparse Representations

In this kind of classifier, the strategy is to use sparse representations of the original data to perform the classification. Thus, the features are first transformed into a sparse representation (see Sect. 5.5) and afterwards, the sparse representation is used by the classifier.

According to Eq. (5.38) it is possible to learn the dictionary $\mathbf{D}$ and estimate the most important constitutive components $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^N$ of the representative signals $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$. In a supervised problem—with labeled data $(\mathbf{x}_i, d_i)$, where d_i is the class of sample $\mathbf{x}_i$—, naturally the classification problem can be stated as follows [14]: given training data $(\mathbf{x}_i, d_i)$, design a classifier h—with parameters θ—which maps the transformed samples $\mathbf{z}_i$ to its classification label d_i, thus, $h(\mathbf{z}_i, \theta)$ should be d_i. In order to classify a new sample data $\mathbf{x}$, it is transformed into $\mathbf{z}$ using dictionary $\mathbf{D}$ and then it is classified as $d = h(\mathbf{z}, \theta)$. Nevertheless, since $\mathbf{Z}$ is estimated to represent the original data efficiently, there is no reason to accept as true that this new representation can ensure an optimal separation of the classes. Another classification

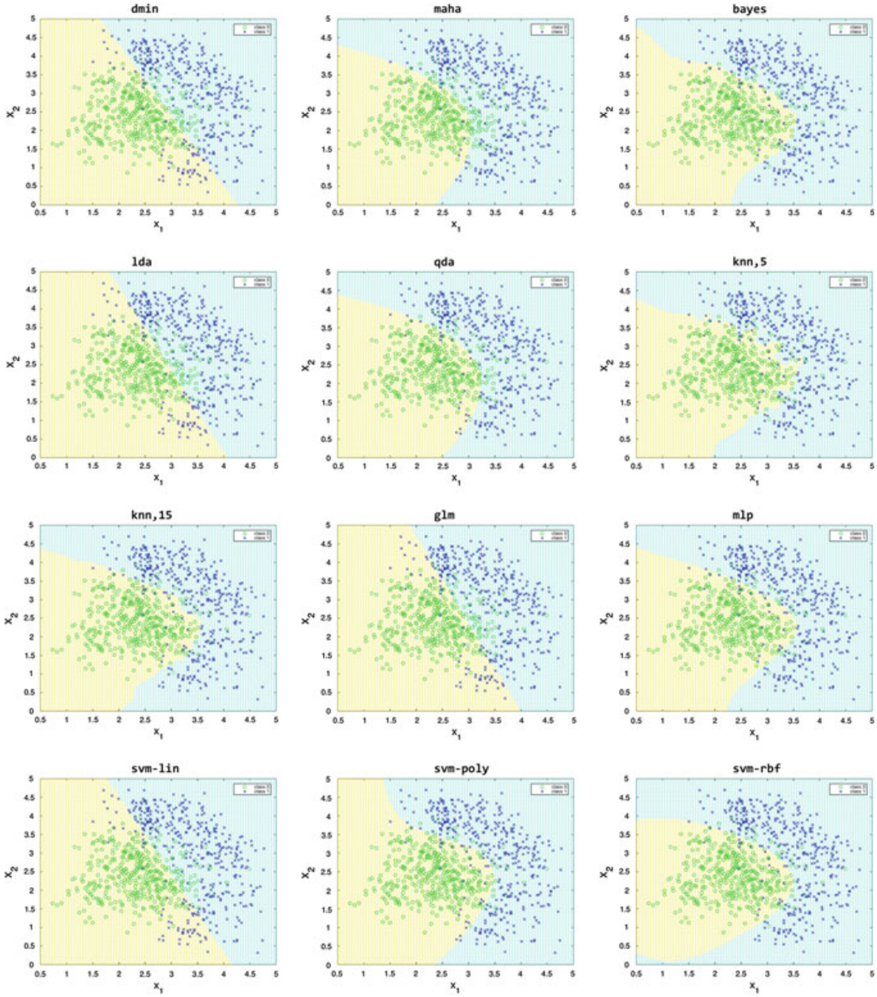


Fig. 6.9 Examples of classification (→ Example 6.4)

strategy uses one dictionary $\mathbf{D}_k$ per class [15], that is learned using the set $\mathbf{X}_k$,[1] that contains only the samples of class ω_k of the training data: $\mathbf{X}_k = \{\mathbf{x}_i | d_i = k\}$. With this strategy, using (5.39) a test sample $\mathbf{x}$ is codified by $\mathbf{z} = \mathbf{z}_k$ with dictionary $\mathbf{D} = \mathbf{D}_k$ for all classes $k = 1 \ldots K$, and a reconstruction error is computed as $e_k = ||\mathbf{x} - \mathbf{D}_k\mathbf{z}_k||$. Finally, sample $\mathbf{x}$ is assigned to the class with the smallest reconstruction error:

[1] There are some approaches that define the dictionary as the original samples (see Sparse Representation Classification (SRC) [16]) where $\mathbf{D}_k = \mathbf{X}_k$.

$$h_{\text{SPAr}}(\mathbf{x}) = \underset{k}{\operatorname{argmin}}\, ||\mathbf{x} - \mathbf{D}_k\mathbf{z}_k||, \tag{6.31}$$

This method is implemented in function Xsparsecl (see Appendix B) in 𝕏vis Toolbox.

This test strategy, however, does not scale well for a large number of classes. For these reasons, new strategies have been developed in order to learn at the same time *reconstructive* and *discriminative* dictionaries (for robustness to noise and for efficient classification respectively) [17]. This can be achieved by adding a new discrimination term in the objective function that includes the representation that is also the most different from the one of signals in other data classes:

$$\underset{\mathbf{D},\mathbf{Z},\theta}{\operatorname{argmin}}[||\mathbf{X} - \mathbf{DZ}||_2^2 + \gamma J(\mathbf{D}, \mathbf{Z}, \mathbf{d}, \theta)] \quad \text{subject to } ||\mathbf{z}||_0 \leq T. \tag{6.32}$$

The discrimination term $J(\mathbf{D}, \mathbf{Z}, \mathbf{c}, \theta)$ depends on the dictionary, the coefficient vectors, the labels of the samples $\mathbf{d}$, and the parameters θ of the model used for classification. Parameter γ weights the tradeoff between approximation and classification performance. This strategy with a common dictionary has the advantage of sharing some atoms of the dictionary when representing samples of different classes. Equation (6.32) can be solved efficiently by fixed-point continuation methods when the classifier is based on logistic regression methods [18].

Another approach that can be used to classify samples in X-ray testing is based on sparse representations of random patches. This approach, called Adaptive Sparse Representation of Random Patches (ASR+), has been successfully used in other recognition problems [19, 20]. The method consists of two stages (see Fig. 6.10): In the training stage, random patches are extracted from representative images of each class (e.g., in baggage screening we can have handguns, razor blades, etc.) in order to construct representative dictionaries. A stop list is used to remove very common words from the dictionaries [21]. In the testing stage, random test patches of the query image are extracted, and for each non-stopped test patch a dictionary is built concatenating the 'best' representative dictionary of each class. Using this adapted dictionary, each non-stopped test patch is classified following the Sparse Representation Classification (SRC) methodology [16] by minimizing the reconstruction error. Finally, the query image is classified by patch voting. Thus, this approach is able to learn a model for each recognition task dealing with a larger degree of variability in contrast, pose, expression, occlusion, object size, and distance from the X-ray detector.

This method was tested in the recognition of five classes in baggage screening: handguns, shuriken, razor blades, clips, and background (see some samples in Fig. 6.11). In our experiments, there are 100 images per class. All images were resized to 128 × 128 pixels. The evaluation is performed using leave-one-out (see Sect. 6.3.3). The obtained accuracy was $\eta = 97.17\,\%$.

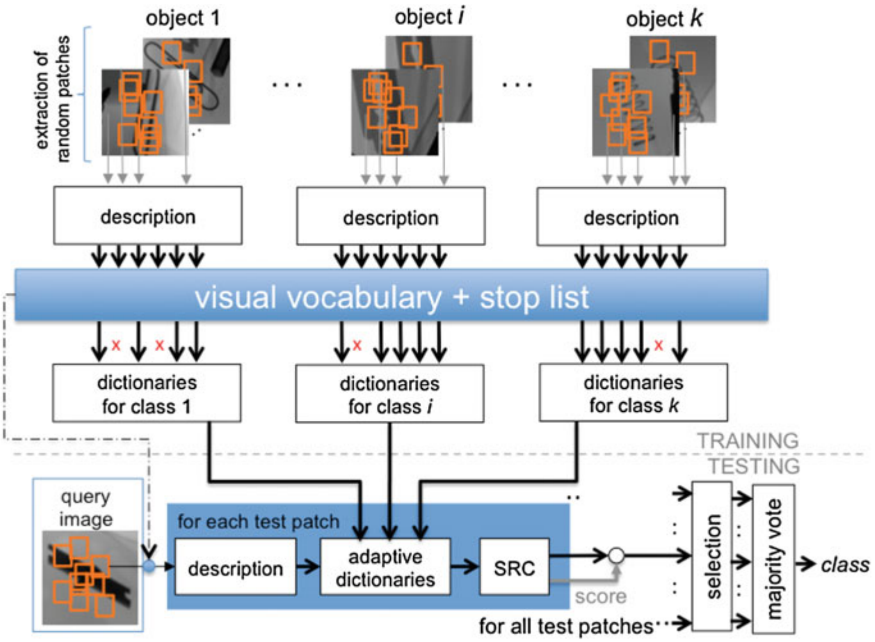


Fig. 6.10 Overview of the proposed method . The figure illustrates the recognition of three different objects. The shown classes are three: clips, razor blades, and springs. There are two stages: Learning and Testing. The stop list is used to filter out patches that are not discriminating for these classes. The stopped patches are not considered in the dictionaries of each class and in the testing stage

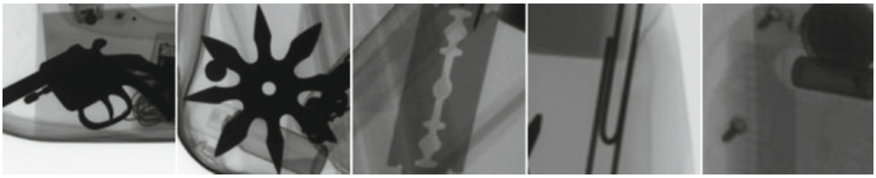


Fig. 6.11 Images used in our experiments. The five classes are: handguns, shuriken, razor blades, clips, and background

6.3 Performance Evaluation

In this section we will see how to evaluate the performance of a classifier and how to build the datasets 'training data' and 'testing data'. In general, there is a set $\mathbb{D}$ that contains all available data, that is, the features of representative samples and their corresponding labels. Sometimes, from set $\mathbb{D}$ a subset $\mathbb{X} \subset \mathbb{D}$ is chosen, however, in most cases $\mathbb{X} = \mathbb{D}$. We call subset $\mathbb{X}$ the 'used data' because it is used to evaluate the performance of a classifier as illustrated in Fig. 6.12. Set $\mathbb{X}$ consists of (i) a matrix

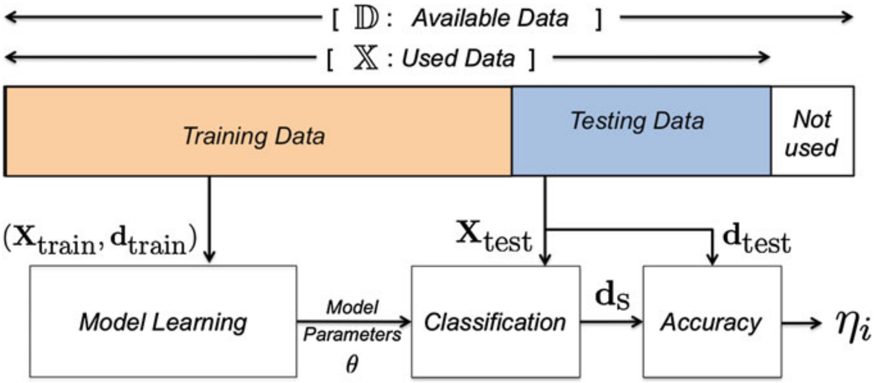

[$\mathbb{D}$: Available Data]
[$\mathbb{X}$: Used Data]
Training Data
Testing Data
Not used
$(\mathbf{X}_{\text{train}}, \mathbf{d}_{\text{train}})$
$\mathbf{X}_{\text{test}}$
$\mathbf{d}_{\text{test}}$
Model Learning
Model Parameters θ
Classification
$\mathbf{d}_S$
Accuracy
η_i

Fig. 6.12 Estimation of the accuracy of a classifier. Figures 6.13, 6.14 and 6.15 show different strategies

$\mathbf{X}$ of size $N \times p$, for N samples and p features; and (ii) a vector $\mathbf{d}$ of N elements with the labels (one label per sample).

In order to estimate the accuracy of a classifier, we can follow this general strategy:

1. From $\mathbb{X}$, select training data $(\mathbf{X}_{\text{train}}, \mathbf{d}_{\text{train}})$ and testing data $(\mathbf{X}_{\text{test}}, \mathbf{d}_{\text{test}})$:

$$(\mathbf{X}_{\text{train}}, \mathbf{d}_{\text{train}}, \mathbf{X}_{\text{test}}, \mathbf{d}_{\text{test}}) = \text{DataSelection}(\mathbb{X}) \tag{6.33}$$

   Typically, a given percentage S of $\mathbb{X}$ is used for training and the rest (100-S) for testing. This means that, we have $N_{\text{train}} = N \times S/100$ samples for training and $N_{\text{test}} = N - N_{\text{train}}$ for training. There are many ways to perform the data selection:

   - Random (yes/no): we can choose randomly N_{train} of $\mathbb{X}$ or for example the first N_{train} samples of $\mathbb{X}$.
   - Stratified (yes/no): in stratified case, we select the same S percentage of each class (so the relative number of samples for each class is the same in original data set and selected data set), whereas in unstratified cases we select S percentage of $\mathbb{X}$ (so the relative number of samples for each class is not necessarily the same in original data set and selected data set).
   - Replacement (with/without): Data selection without replacement means that once a sample has been selected, it may not be selected again. In data selection with replacement a sample of $\mathbb{X}$ is allowed to be replicated. It must be ensured that samples in the training data are not in the testing data and vice versa.

2. Using training data $(\mathbf{X}_{\text{train}}, \mathbf{d}_{\text{train}})$ train a classifier:

$$\theta = \text{ClassifierTrain}(\mathbf{X}_{\text{train}}, \mathbf{d}_{\text{train}}) \tag{6.34}$$

where θ is a vector that contains all parameters of the classifier that was trained. For instance, in a simple classifier like Euclidean minimal distance (see Sect. 6.2.1) we store in θ only the centers of mass of each class in the training set.

3. Using the features of the testing data $\mathbf{X}_{\text{test}}$, the classifier and its parameters θ, we predict the labels of each testing sample and store them in vector $\mathbf{d}_s$ of N_{test} elements:
$$\mathbf{d}_s = \text{Classify}(\mathbf{X}_{\text{test}}, \theta). \tag{6.35}$$
It is worth mentioning that in this step it is not allowed to use the labels of the testing data $\mathbf{d}_{\text{test}}$.
4. Now, we can compute the accuracy of the testing data defined as
$$\eta_i = \frac{\#\text{test samples correctly predicted}}{N_{\text{test}}}. \tag{6.36}$$
5. In (6.36), we use index i because the procedure from steps 1 to 4 can be repeated n times, for $i = 1 \ldots n$. Thus, we can compute the final estimation of the accuracy as
$$\eta = \frac{1}{n}\sum_{i=1}^{n} \eta_i. \tag{6.37}$$

In the following section, we will explain typical strategies used in the literature.

6.3.1 Hold-Out

In hold-out, we take a percentage S of $\mathbb{X}$ for training and the rest for testing as shown in Fig. 6.13. In our general methodology, this strategy corresponds to $n = 1$ in (6.37). This is the simplest way of how to evaluate the accuracy. It is recommended just in case the computational time is so enormous that the cost of training a classifier several times is prohibitive. Hold-out can be a good starting point to test if the features and classifier that we are designing are suitable for the recognition task. Nevertheless, the standard deviation of the accuracy estimation can be very high as we will see in next example.

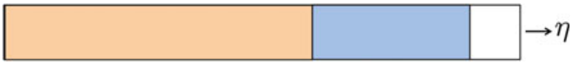


Fig. 6.13 Estimation of the accuracy of a classifier using hold-out. The figure follows the color representation of Fig. 6.12 for training and testing data

Matlab Example 6.5 In this example, we show how to evaluate a classifier using hold-out strategy. We use the same simulated data addressed in Example 6.1 and illustrated in Fig. 6.2.

Listing 6.5 : Hold-out

```
% HoldOutExample.m
load datasim                          % simulated data (2 classes, 2 features)
X = Xtrain; d = dtrain;               % data selection
c.name = 'knn'; c.options.k = 5;      % knn with 5 neighbors
op.c = c; op.strat = 1;op.s = 0.75;   % stratify with S=75% for training
acc = Xholdout(X,d,op)                % holdout
```

The output of this code is the value of the estimated accuracy. This number should be around 92 %. This method is implemented in function Xholdout (see Appendix B) in 𝕏vis Toolbox. If we repeat this experiment 1000 times, the mean of the accuracy is 92.25 %, the standard deviation is 1.6 %, the maximal value is 97.00 % and the minimal value is 86.50 %, i.e., the estimation is not very accurate because there is a variation of 10.5 % between maximal and minimal value! □

6.3.2 Cross-Validation

Cross-validation is widely used in machine learning problems [22]. In cross-validation, the data are divided into v folds. A portion $s = (v - 1)/v$ of the whole data is used to train and the rest $(1/v)$ for test. This experiment is repeated v times rotating train and test data to evaluate the stability of the classifier as shown in Fig. 6.14. Then, when training is performed, the samples that were initially removed can be used to test the performance of the classifier on these test data. Thus, one can evaluate the generalization capabilities of the classifier by testing how well the method will classify samples that have not already been examined. The estimated performance, η, is calculated as the mean of the v percentages of the true classifications are tabulated in each case, i.e., $n = v$ (6.37). In our experiments, we use

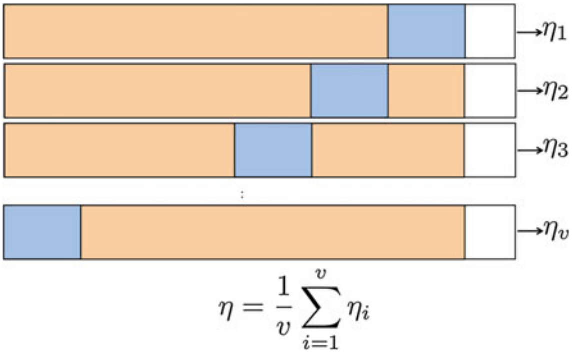


Fig. 6.14 Estimation of the accuracy of a classifier using cross-validation with v folds. The figure follows the color representation of Fig. 6.12 for training and testing data

$v = 10$ folds.[2] Confidence intervals, where the classification performance η expects to fall, are obtained from the test sets. These are determined by the cross-validation technique, according to a t–Student test [10]. Thus, the performance and also the confidence can be assessed.

Matlab Example 6.6 In this example, we show how to evaluate several classifiers using cross-validation strategy with 10 folds. We use the same simulated data addressed in Example 6.1 and illustrated in Fig. 6.2. The decision lines of the classifiers used in this example are illustrated in Fig. 6.9.

Listing 6.6 : Cross-validation

```
% CrossValidationExample.m
load datasim  % simulated data (2 classes, 2 features)
              % Xtrain,dtrain features and labels for training
              % Xtest, dtest  features and labels for testing
X = [Xtrain; Xtest]; d = [dtrain; dtest];                    % all data
k = 0;
k=k+1;opc(k).name = ‘dmin’;  opc(k).options.p = [];          % Euclidean distance
k=k+1;opc(k).name = ‘maha’;  opc(k).options.p = [];          % Mahalanobis distance
k=k+1;opc(k).name = ‘bayes’; opc(k).options.p = [];          % Bayes
k=k+1;opc(k).name = ‘lda’;   opc(k).options.p = [];          % LDA
k=k+1;opc(k).name = ‘qda’;   opc(k).options.p = [];          % QDA
k=k+1;opc(k).name = ‘knn’;   opc(k).options.k = 5;           % KNN with 5 neighbors
k=k+1;opc(k).name = ‘knn’;   opc(k).options.k = 15;          % KNN with 5 neighbors
k=k+1;opc(k).name = ‘glm’;   opc(k).options.method = 2; opc(k).options.iter = 12;   % GLM
k=k+1;opc(k).name = ‘mlp’;   opc(k).options.method = 2; opc(k).options.alpha = 0.2; % MLP
      opc(k).options.iter=12;opc(k).options.nhidden = 6;opc(k).options.ncycles = 60;
k=k+1;opc(k).name = ‘svm’;   opc(k).options.kernel = ‘–t 0’;  % linear–SVM
k=k+1;opc(k).name = ‘svm’;   opc(k).options.kernel = ‘–t 1’;  % polynomial–SVM
k=k+1;opc(k).name = ‘svm’;   opc(k).options.kernel = ‘–t 2’;  % rbf–SVM
op.strat=1; op.c = opc; op.v = 10; op.show = 1; op.p = 0.95;  % 10–fold cross–validation
[acc,ci] = Xcrossval(X,d,op);
```

The output of this code is the estimated accuracy of each classifier. The results are shown in Table 6.3, where the accuracy (η), the lower (η_{lower}), and upper (η_{upper}) endpoints of the 95 % confidence interval are presented for each method. The reader can compare these results with the accuracies presented in Table 6.2. This method is implemented in function Xcrossval (see Appendix B) in $\mathbb{X}$VIS Toolbox. In order to compare Hold-Out with Cross-Validation we can repeat the cross-validation 1000 times for classifier KNN with 5 neighbors (as computed in Example 6.5). The results are: mean of the accuracy is 91.99 %, the standard deviation is 0.39 %, the maximal value is 93.25 % and the minimal value is 90.62 %, i.e., the estimation is more accurate because there is a variation of 2.6 % between maximal and minimal. In hold-out the variation is 10.5 %. □

[2] The number of folds v can be another number, for instance 5-fold or 20-fold cross-validation estimate offers very similar performances. In our experiments, we use 10-fold cross-validation because it has become the standard method in practical terms [23].

Table 6.3 Accuracy using cross-validation given in [%]

Name	η_{lower}	η	η_{upper}
dmin	83.02	85.33	87.64
maha	86.87	88.92	90.97
bayes	91.61	93.25	94.89
lda	83.29	85.58	87.88
qda	90.14	91.92	93.70
knn, 5	91.06	92.75	94.44
knn, 15	91.70	93.33	94.96
glm	83.47	85.75	88.03
mlp	91.70	93.33	94.96
svm-lin	83.11	85.42	87.72
svm-poly	91.24	92.92	94.59
svm-rbf	91.52	93.17	94.81

6.3.3 Leave-One-Out

In leave-one-out strategy, we perform the cross-validation technique with N folds (the number of samples of $\mathbb{X}$). This means, we leave one sample out for testing and we train with the rest ($N - 1$ samples). The operation is repeated for each sample as illustrated in Fig. 6.15). The estimated accuracy is the average over the N estimations.

This method is implemented in function Xleaveoneout (see Appendix B) in $\mathbb{X}$VIS Toolbox. In order to illustrate the estimation accuracy using leave-one-out, we can change the last line of the code of Example 6.6 by:

```
[acc,ci] = Xleaveoneout(X,d,op);
```

The results are summarized in Table 6.4, where the accuracy (η), the lower (η_{lower}) and upper (η_{upper}) endpoints of the 95 % confidence interval are presented for each method. The reader can compare these results with the accuracies presented in

Fig. 6.15 Estimation of the accuracy of a classifier using leave-one-out. The figure follows the color representation of Fig. 6.12 for training and testing data

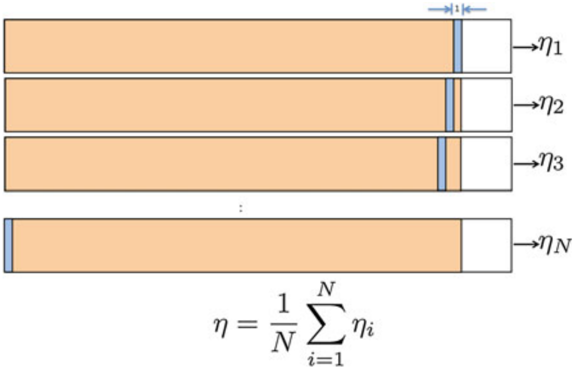


Table 6.4 Accuracy using leave-one-out in [%]

Name	η_{lower}	η	η_{upper}
dmin	83.60	85.58	87.57
maha	87.05	88.83	90.62
bayes	91.92	93.33	94.74
lda	83.51	85.50	87.49
qda	90.19	91.75	93.31
knn, 5	90.56	92.08	93.61
knn, 15	91.83	93.25	94.67
glm	83.60	85.58	87.57
mlp	92.01	93.42	94.82
svm-lin	83.16	85.17	87.18
svm-poly	91.56	93.00	94.44
svm-rbf	91.92	93.33	94.74

Tables 6.2 and 6.3. In order to compare Hold-Out and Cross-Validation we can run Leave-one-out once for classifier KNN with 5 neighbors (as computed in Examples 6.5 and 6.6). It is not necessary to repeat it, because Leave-one-out always obtains the same result. The results are: mean of the accuracy is 91.99 %. It is clear that the standard deviation is 0 %, and the maximal and minimal values are equal to the mean.

6.3.4 Confusion Matrix

The confusion matrix, $\mathbf{T}$, is a $K \times K$ matrix, where K is the number of classes of our data. The element $T(i, j)$ of the confusion matrix is defined as the number of samples that belong to class ω_i and were classified as ω_j. A perfect classification means that $T(i, i)$ is N_i and $T(i, j) = 0$ for $i \neq j$, where N_i is the number of samples of class ω_i.

Matlab Example 6.7 In this example, we show how to compute the confusion matrix for two classifiers LDA and SVM-RBF. We use the same simulated data addressed in Example 6.1 and illustrated in Fig. 6.2. The decision lines of the classifiers used in this example are illustrated in Fig. 6.9.

Listing 6.7 : Confusion Matrix

```
% ConfusionMatrixExample.m
close all;
load datasim  % simulated data (2 classes, 2 features)
              % Xtrain,dtrain features and labels for training
              % Xtest, dtest  features and labels for testing
op1.p = [0.5 0.5];                         % priors for LDA
```

```
ds1    = Xlda(Xtrain,dtrain,Xtest,op1);     % LDA classifier
T1     = Xconfusion(dtest,ds1)              % Confusion Matrix
figure(1);
Xshowconfusion(T1,1);title('lda')           % Display confusion matrix
op2.kernel = '-t 2';                        % SVM parameter for RBF kernel
ds2    = Xsvm(Xtrain,dtrain,Xtest,op2);     % SVM classifier
T2     = Xconfusion(dtest,ds2)              % Confusion Matrix
figure(2);
Xshowconfusion(T2,1);title('svm-rbf')       % Display confusion matrix
```

The output of this code is two confusion matrices that are illustrated in Fig. 6.16. This method is implemented in function Xconfusion (see Appendix B) in 𝕏vis Toolbox. In addition, command Xshowconfusion (see Appendix B) can be used to visualize a confusion matrix. □

Typically, in X-ray testing, there are two classes: ω_1 known as the target or object of interest, and ω_0 known as the no-target or background. In this two-class recognition problem (known as 'detection'), we are interested in detecting the target correctly. It is very helpful to build a 2 × 2 confusion matrix as shown in Table 6.5. We distinguish:

- True Positive (*TP*): number of targets correctly classified.
- True Positive (*TN*): number of non-targets correctly classified.
- False Positive (*FP*): number of non-targets classified as targets. The false positives are known as 'false alarms' and 'Type I error'.
- False Positive (*FN*): number of targets classified as no-targets. The false negatives are known as 'Type II error'.

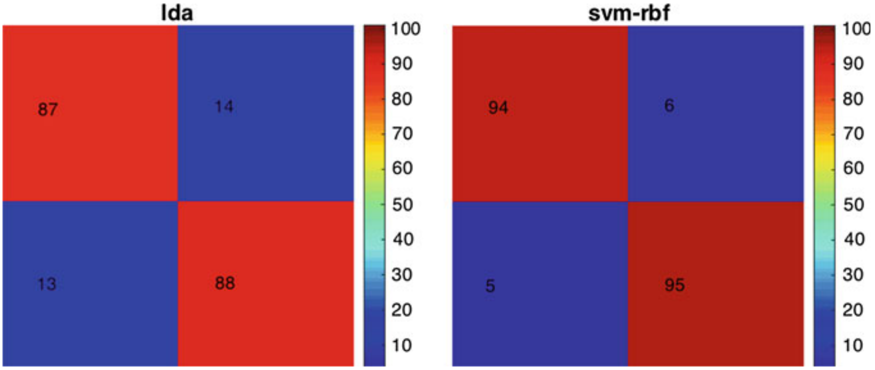


Fig. 6.16 Visualization of confusion matrix of LDA and SVM-RBF (→ Example 6.7)

Table 6.5 Confusion matrix for two classes

Predicted → Actual ↓	ω_1	ω_0
ω_1	*TP*	*FN*
ω_0	*FP*	*TN*

From these statistics, we can obtain the following definitions:
Positive instances:

$$P = TP + FN \tag{6.38}$$

Negative instances:

$$N = TN + FP \tag{6.39}$$

Detections:

$$D = TP + FP \tag{6.40}$$

True positive rate, known as Sensitivity or Recall:

$$TPR = S_n = RE = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{6.41}$$

Precision or Positive Predictive Value:

$$PR = \frac{TP}{D} = \frac{TP}{TP + FP} \tag{6.42}$$

True negative rate, known as Specificity:

$$TNR = Sp = \frac{TN}{N} = \frac{TN}{TN + FP} \tag{6.43}$$

False positive rate, known as 1-Specificity:

$$FPR = 1 - Sp = \frac{FP}{N} = \frac{FP}{TN + FP} \tag{6.44}$$

False negative rate, known as Miss Rate:

$$FNR = MR = \frac{FN}{P} = \frac{FN}{TP + FN} \tag{6.45}$$

Accuracy:

$$ACC = \frac{TP + TN}{P + N} \tag{6.46}$$

Ideally, a perfect detection means all existing targets are correctly detected without any false alarms, i.e., $TP = P$ and $FP = 0$. It is equivalent to: (i) $TPR = 1$ and $FPR = 0$, or (ii) $Pr = 1$ and $Re = 1$, or (ii) $FN = FP = 0$.

6.3.5 ROC Curve

It is clear that the performance of a detector depends on some parameters, e.g., the value of a threshold θ when segmenting a gray value image. This means that we can analyze the performance of the detector by variating its parameter θ. We can analyze the values *TPR* and *FPR* as defined in (6.41) and (6.44) respectively. In this case, we obtain $TPR(\theta)$ and $FPR(\theta)$ because the values of these variables depend on parameter θ.

The receiver operation characteristic (ROC) curve is a plot of $TPR(\theta)$ versus $FPR(\theta)$. Thus, we choose different values $\{\theta_i\}_{i=1}^{n}$ and for each value θ_i we plot the corresponding point (x_i, y_i), where $x_i = FPR(\theta_i)$ and $y_i = TPR(\theta_i)$ as illustrated in Fig. 6.17.

Matlab Example 6.8 In this example, we evaluate an approach that is used to detect defects in aluminum castings. The evaluation is performed on series `C0021` of 𝔾𝔻𝕏ray which contains a wheel with several defects. The wheel is captured in 37 positions. The defect detection is achieved by thresholding an image that is computed from the difference between original and a median filtered image as explained in Sect. 4.3.2. Since the series contains the annotations for the ground truth, it is very simple to evaluate the performance of the detector. We only need to generate a binary image with the detected potential defects. This detection is performed by program `MedianDetection.m`.[3] For example, the detection achieved by this program in a single image of the series is shown in Fig. 6.18. The presented code of this example, evaluates each image of the series and computes the true and false positives in order to plot the ROC curve.

Listing 6.8 : Performance using ROC curve

```
% ROCPerformance.m
clt
n = 4;
p(n).areamin = 0;
show = 0;
for i=1:n
    p(i).gaussianmask = 5;
    p(i).medianmask   = 23;
    p(i).threshold    = 8+4*i;
    p(i).areamin      = 20;
    p(i).dilationmask = 17;
end
pascalth = 0.5;
[TP,FP,P,N] = Xdetectionstats('C',21,'ground_truth.txt',...
              1:37,'MedianDetection',p,pascalth,show);
TPR = TP./P;
FPR = FP./N;
Xplotroc(FPR,TPR)
```

The output of this code is the ROC curve illustrated in Fig. 6.19. This code uses the powerful command Xdetectionstats (see Appendix B) of 𝕏vis Toolbox that evaluates each potential detection (of each image of the series) in terms of PASCAL

[3] Available in directory `cla` of the examples of 𝕏VIS Toolbox.

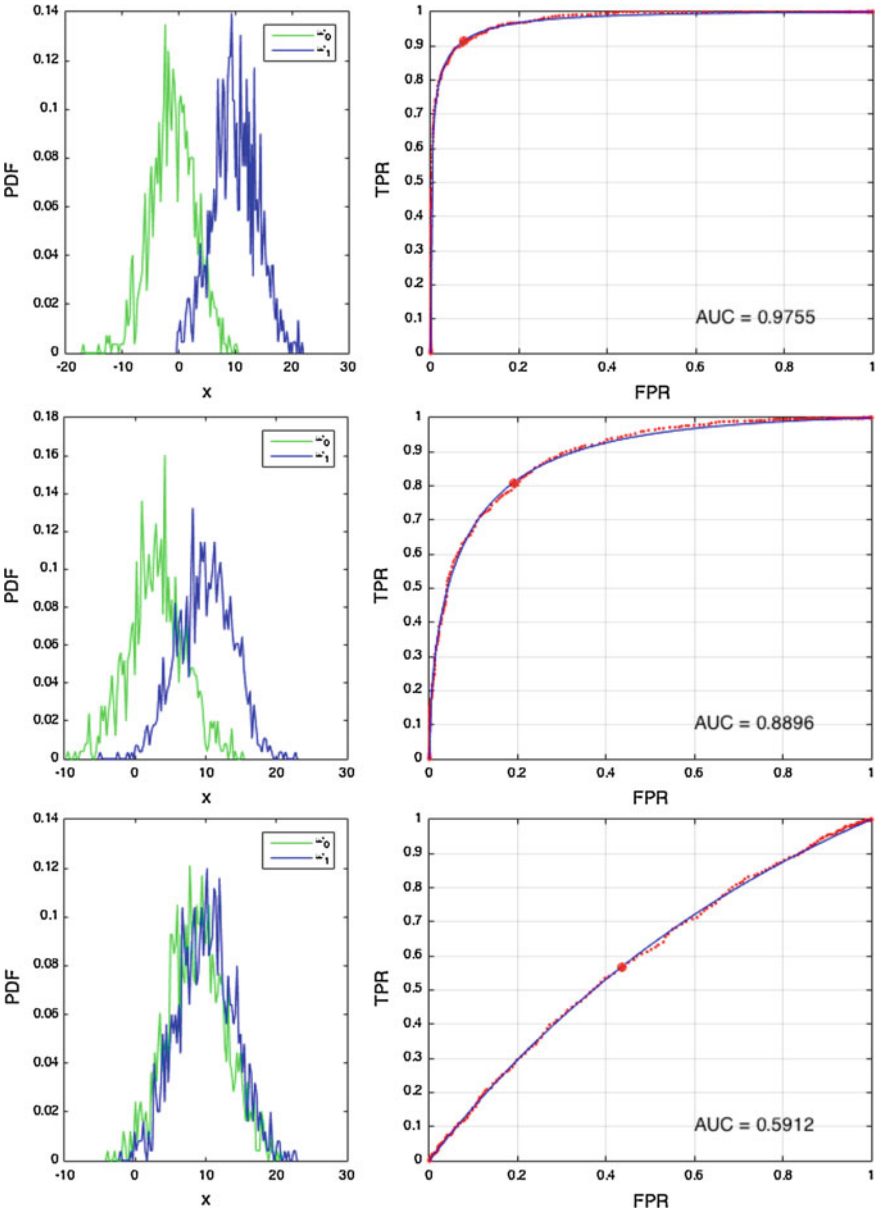


Fig. 6.17 ROC curves (*right*) for different class distributions (*left*). The area under the curve (AUC) gives a good measure of the performance of the detection. The obtained points (x_i, y_i) are used to fit the ROC curve to $y = (1 - a^{\gamma x^b})/(1 - a^{\gamma})$. In each ROC curve, the 'best operation point' is shown as *. This point is defined as the closest point to ideal operation point (0, 1)

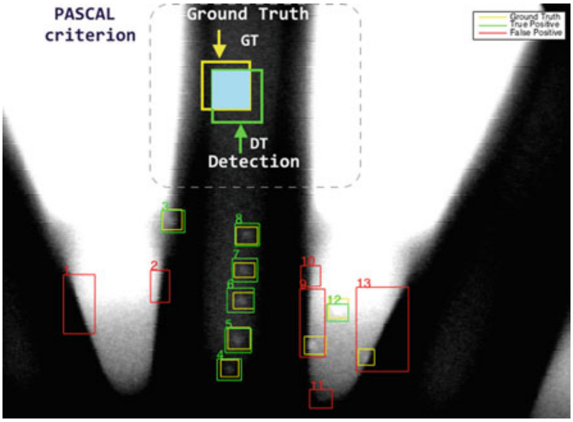


Fig. 6.18 Detection on a single image. A detection is considered as true positive is the normalized area of overlap (6.47) is greater than 50 %. In this example, the true positives are shown in *green*, the false positives in *red* and the ground truth in *yellow* (→ Example 6.8)

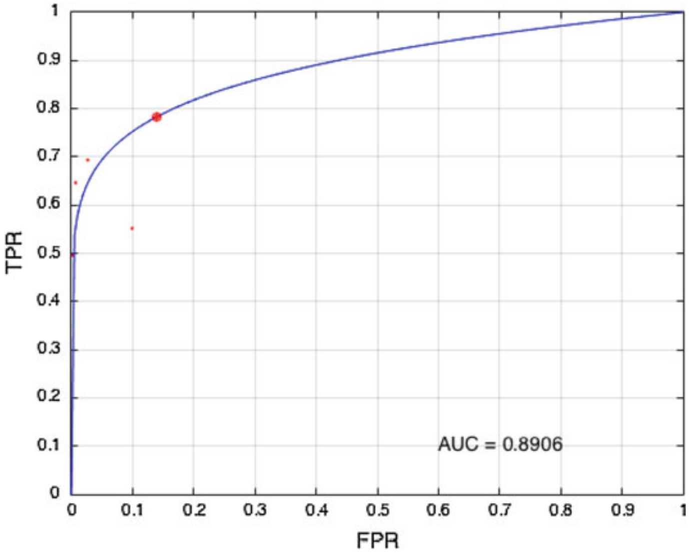


Fig. 6.19 ROC curve computed by analyzing detector approach in series `C0021` of $\mathbb{GDX}$ray (→ Example 6.8)

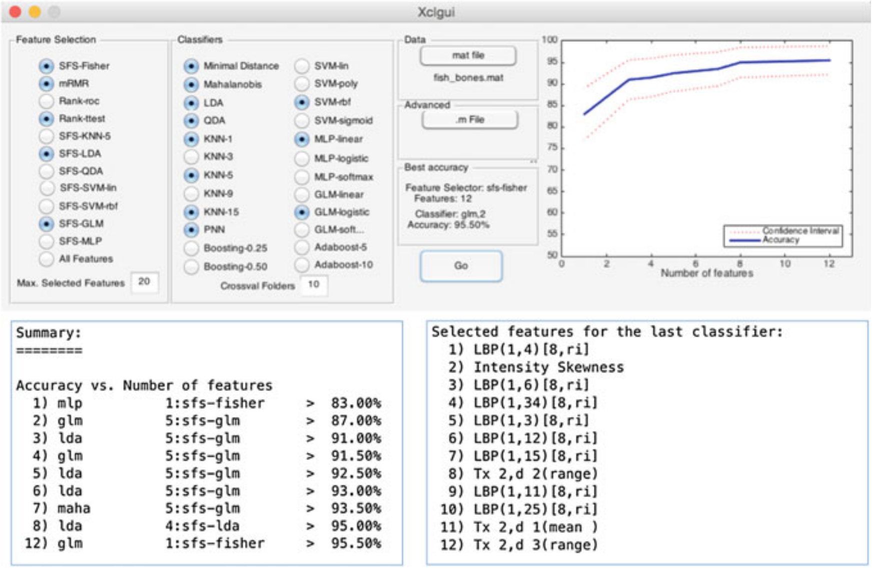
criterion [24], where a detection is considered correct if the normalized area of overlap A between the detected bounding box DT and the ground truth bounding box GT exceeds 0.5, where A is defined as follows:

$$A = \frac{\text{area}(\mathsf{GT} \cap \mathsf{DT})}{\text{area}(\mathsf{GT} \cup \mathsf{DT})}, \tag{6.47}$$

with GT ∩ DT the intersection of the detected and ground truth bounding boxes and GT ∪ DT their union, as illustrated in Fig. 6.18. □

6.4 A Final Example

Our final example is implemented in 𝕏VIS Toolbox. In this toolbox, there are two main graphic user interfaces (GUI): (i) For feature extraction, we have Xfxgui (see Appendix B) that was used in our final example of Chap. 5 (see Sect. 5.7 and Fig. 5.28). With Xfxgui (see Appendix B), the user can choose the feature groups that will be extracted. (ii) For feature and classifier selection, we have Xclgui (see Appendix B) (Fig. 6.20), the user can choose the feature selection algorithms to be used, the maximal number of features to be selected, the classifiers that will be evaluated, and the number of folds of the cross-validation technique. Using only these two graphic user interfaces, it is possible to easily design the computer vision system automatically according to the general computer vision framework explained in these two chapters.



Feature Selection
● SFS-Fisher
● mRMR
○ Rank-roc
● Rank-ttest
○ SFS-KNN-5
● SFS-LDA
○ SFS-QDA
○ SFS-SVM-lin
○ SFS-SVM-rbf
● SFS-GLM
○ SFS-MLP
○ All Features
Max. Selected Features 20

Classifiers
● Minimal Distance
● Mahalanobis
● LDA
● QDA
● KNN-1
○ KNN-3
● KNN-5
○ KNN-9
● KNN-15
● PNN
○ Boosting-0.25
○ Boosting-0.50
○ SVM-lin
○ SVM-poly
● SVM-rbf
○ SVM-sigmoid
● MLP-linear
○ MLP-logistic
○ MLP-softmax
○ GLM-linear
● GLM-logistic
○ GLM-soft...
○ Adaboost-5
○ Adaboost-10
Crossval Folders 10

Data
fish_bones.mat

Advanced

Best accuracy
Feature Selector: sfs-fisher
Features: 12
Classifier: glm,2
Accuracy: 95.50%

```
Summary:
========

Accuracy vs. Number of features
  1) mlp        1:sfs-fisher    >  83.00%
  2) glm        5:sfs-glm       >  87.00%
  3) lda        5:sfs-glm       >  91.00%
  4) glm        5:sfs-glm       >  91.50%
  5) lda        5:sfs-glm       >  92.50%
  6) lda        5:sfs-glm       >  93.00%
  7) maha       5:sfs-glm       >  93.50%
  8) lda        4:sfs-lda       >  95.00%
 12) glm        1:sfs-fisher    >  95.50%
```

```
Selected features for the last classifier:
  1) LBP(1,4)[8,ri]
  2) Intensity Skewness
  3) LBP(1,6)[8,ri]
  4) LBP(1,34)[8,ri]
  5) LBP(1,3)[8,ri]
  6) LBP(1,12)[8,ri]
  7) LBP(1,15)[8,ri]
  8) Tx 2,d 2(range)
  9) LBP(1,11)[8,ri]
 10) LBP(1,25)[8,ri]
 11) Tx 2,d 1(mean )
 12) Tx 2,d 3(range)
```

Fig. 6.20 Graphic user interface Xclgui (see Appendix B) for feature and classifier selection with 𝕏VIS Toolbox: the user can select the feature selection algorithms, the classifiers, the features file (computed by previous step and stored in `fish_bones.mat`) and the number of folds used by cross-validation technique. In this example, a performance of 95.5 % was achieved using a GLM-classifier with 12 features. In .mat file the user must store the features in variable `X`, the labels in variable `d` and the names of the features in variable `Xn`

In order to find the *best* classifier, we evaluate the performance of the r classifiers on the s subsets of selected features using an exhaustive search as shown in Algorithm 1. For instance, we could have: $s = 2$ feature selection algorithms (SFS with Fisher criterion and mRMR), and $r = 3$ classifiers (LDA, KNN with 3 neighbors, and SVM with RBF). The accuracy η is defined as the proportion of true results. As we can see, the accuracy and the confidence intervals of the classification are evaluated using cross-validation (see Sect. 6.3.2). According to Algorithm 1, the highest achieved accuracy (searching in all r classifiers and all s feature selection algorithms) is computed as $\hat{\eta}$. This algorithm is implemented in Xclgui (see Appendix B) as a graphic user interface (Fig. 6.20) and as command Xclsearch (see Appendix B).

In Algorithm 1, there are s feature selection approaches. Thus, there are s subsets of selected features $\mathbf{X}_i$ with p_i features each (for $i = 1 \ldots s$). In addition, there are r different classifiers: h_k, for $k = 1 \ldots r$. In our exhaustive search, each subset of features i and each classifier k will be tested using the first 1, 2, … p_i selected features of $\mathbf{X}_i$. The output of this exhaustive search is: the selected features, the selected classifier, the best accuracy, and its corresponding confidence intervals.

Matlab Example 6.9 In this example we can see the whole process: (i) feature extraction, (ii) feature selection and (ii) and classifier selection. 𝕏vis Toolbox provides a suite of helpful commands that can be used in this process. The idea is to design a classifier that can be used to detect fish bones in X-ray images of salmon filets (see Example 5.8). For this end, as we have 200 small X-ray images (100 × 100 pixels) of salmon filets, 100 with fish bones and 100 without fish bones. The images are available in series `N0002` of 𝔾𝔻𝕏ray. In this program, we show how to automatically design a computer vision system for this application.

Listing 6.9 : Feature extraction, feature selection and classification selection

```
% ClassificationSelection.m
close all
imagesdir   = Xgdxdir('N',2);                    % directory of series N0002 of GDX
opf.b       = Xfxbuild({'basicint','gabor',...   % features to be extracted
                        'lbpri','haralick'});
[X0,Xn0]    = Xfxtractor(imagesdir,'png',opf);   % feature extraction
[X,Xn]      = Xnorotation(X0,Xn0);               % only rotation invariant features
d           = Xloaddata('N',2,'labels.txt');     % labels
fs          = {'sfs-fisher','mRMR','rank-ttest'}; % Feature selectors to be tested
cl          = {'dmin','maha','lda','qda',...      % Classifiers to be tested
               'glm1','glm2','pnn','svm2'};
options.Xn  = Xn;                                % Feature names
options.p   = 20;                                % Maximal number of selected features
figure
[bcs,selec,acc] = Xclsearch(X,d,cl,fs,options);  % Feature and classifier selection
```

Features are extracted from directory `imagesdir`, in this case it is the directory of series `N0002` of 𝔾𝔻𝕏ray. The user can choose the features to be extracted using Xfxbuild (see Appendix B), in this example, basic intensity features, Gabor, LBP and Haralick are extracted. The features of all images of this directory are extracted using Xfxtractor (see Appendix B). The extracted features must be rotation invariant because the fish bones can be oriented in any direction. The labels are available in file `labels.txt` of this directory. The user can decide which feature selection techniques and classifiers will be tested in a simple way by defining variables `fs` and `cl` respectively. After these definitions, we run command Xclsearch (see Appendix B) that follows Algorithm 1. As shown in this code, it is really easy to define the features and classifiers to be tested. The output of this code is similar to the output presented in Fig. 6.20. The user can edit this file in order to add or delete other features, features selection algorithms, classifiers, and number of features to be selected. □

Algorithm 1 Classifier Selection

```
Input: Subsets $\{\mathbf{X}_i, p_i\}_{i=1}^{s}$ of selected features, and labels $\mathbf{d}$ of the samples.
$\hat{\eta} = 0$//Initialization of the highest accuracy
for $i = 1$ to $s$ do
  for $j = 1$ to $p_i$ do
    $\mathbf{X} = \text{LeftColumns}(\mathbf{X}_i, j)$ //First $j$ selected features of $\mathbf{X}_i$
    for $k = 1$ to $r$ do
      $\eta_k = \text{CrossValidation}(h_k, \mathbf{X}, \mathbf{d})$ //Accuracy of classifier $h_k$ on data $\mathbf{X}$
      if $\eta_k > \hat{\eta}$ then
        $\hat{\eta} = \eta_k$ //Highest performance
        $\hat{\mathbf{X}} = \mathbf{X}$ //Selected features
        $\hat{i} = i$ //Best feature selection approach
        $\hat{j} = j$ //Number of selected features
        $\hat{k} = k$ //Best classifier
      end if
    end for
  end for
end for
Output: $\hat{\eta}, \hat{\mathbf{X}}, \hat{i}, \hat{j}, \hat{k}$
```

6.5 Summary

In this chapter, we covered the following classifiers:

- Minimal distance (using Euclidean and Mahalanobis distance)
- Bayes
- Linear and quadratic discriminant analysis
- K-nearest neighbors
- Neural networks
- Support vector machines
- Classifiers using sparse representations

In addition, several simple examples were presented using simulated data. The reader can easily modify the proposed implementations in order to test different classification strategies or real data.

Afterwards, we presented how to estimate the accuracy of a classifier using hold-out, cross-validation, and leave-one-out. We covered the well-known confusion matrix and receiver-operation-characteristic curve will be outlined as well.

Finally, we presented an example that involves all steps of a pattern recognition problem, i.e., feature extraction, feature selection, classifier's design, and evaluation. All steps can be designed automatically using a simple code program of approximately 10 lines.

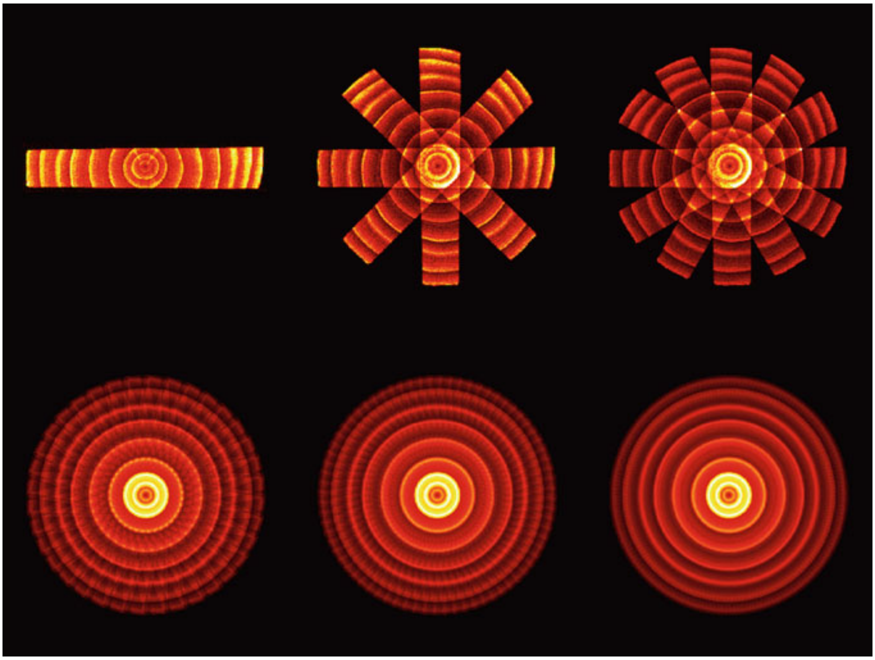
References

1. Jain, A., Duin, R., Mao, J.: Statistical pattern recognition: a review. IEEE Trans. Pattern Anal. Mach. Intell. **22**(1), 4–37 (2000)
2. Fukunaga, K.: Introduction to Statistical Pattern Recognition, 2nd edn. Academic Press Inc., San Diego (1990)
3. Carvajal, K., Chacón, M., Mery, D., Acuna, G.: Neural network method for failure detection with skewed class distribution. Insight **46**(7), 399–402 (2004)
4. Silverman, B.W.: Density Estimation for Statistics and Data Analysis, vol. 26. CRC Press, Boca Raton (2003)
5. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. Springer, New York (2009)
6. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley, New York (2001)
7. Bentley, J.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)
8. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (2005)
9. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
10. Mitchell, T.: Machine Learning. McGraw-Hill, Boston (1997)
11. Nabney, I.: NETLAB: Algorithms for Pattern Recognition. Springer Science & Business Media, New York (2002)
12. MathWorks: Neural Network Toolbox for Use with MATLAB: User's Guide. The MathWorks Inc., Massachusetts (2015)
13. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
14. Bar, L., Sapiro, G.: Hierarchical dictionary learning for invariant classification. In: 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 3578–3581 (2010)
15. Mairal, J., Bach, F., Ponce, J., Sapiro, G., Zisserman, A.: Discriminative learned dictionaries for local image analysis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2008)
16. Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S., Ma, Y.: Robust face recognition via sparse representation. IEEE Trans. Pattern Anal. Mach. Intell. **31**(2), 210–227 (2009)
17. Tosic, I., Frossard, P.: Dictionary learning. IEEE Signal Process. Mag. **28**(2), 27–38 (2011)
18. Mairal, J., Bach, F., Ponce, J., Sapiro, G., Zisserman, A.: Supervised dictionary learning. Technical report 6652, INRIA (2008)
19. Mery, D., Bowyer, K.: Recognition of facial attributes using adaptive sparse representations of random patches. In: 1st International Workshop on SoftBiometrics, in conjunction with European Conference on Computer Vision (ECCV 2014) (2014)
20. Mery, D., Bowyer, K.: Face recognition via adaptive sparse representations of random patches. In: IEEE Workshop on Information Forensics and Security (WIFS 2014) (2014)
21. Sivic, J., Zisserman, A.: Video Google: a text retrieval approach to object matching in videos. In: International Conference on Computer Vision (ICCV 2003), pp. 1470–1477 (2003)

22. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: International Joint Conference on Artificial Intelligence, vol. 14, pp. 1137–1145. Citeseer (1995)
23. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
24. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J., Zisserman, A.: The Pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)

Chapter 7
Simulation in X-ray Testing

Abstract In order to evaluate the performance of computer vision techniques, computer simulation can be a useful tool. In this chapter we review some basic concepts of the simulation of X-ray images, and present simple geometric and imaging models that can be used in the simulation. We explain the basic simulation principles and we address some techniques of simulated defects (that can be used to assess the performance of a computer vision method for automated defect recognition). The chapter also has some Matlab examples that the reader can run and follow, along with examples of simulated defects in castings and welds.



Cover image: *X-ray image of a wood located in 1, 4, 6, 36, 72 and 180 positions (image* `N0010_0051` *colored with 'hot' colormap).*

D. Mery, *Computer Vision for X-Ray Testing*, DOI 10.1007/978-3-319-20747-6_7

7.1 Introduction

In order to evaluate the performance of computer vision techniques, e.g., an automated defect recognition system, computer simulation can be a useful tool [1, 2].

The simulated X-ray images, however, should be as similar as possible to real X-ray images. For this purpose, the simulation should model the physics of the X-ray formation (generation, interaction and detection) and handle complex 3D objects efficiently [3]. State-of-the-art of computer modeling of X-ray testing methods are able to simulate different X-ray spectrum and X-ray source size, varied photon-matter interactions, and several X-ray detector responses. Special attention has been given to general purpose Monte Carlo methods that are able to calculate higher order scattering events [4–6]. A computer simulator for X-ray testing should include the following modules[7]:

- Source model: generates the spectra of X-ray tubes and isotopic sources.
- Ray-tracing engine: determines ray paths in complex geometries of test objects.
- Material data base: contains cross-section data.
- Straight line attenuation model: determines the contribution of direct radiation, a scatter model, and a post-processor, combining both contributions.
- Detector model: converts radiation to an optical density and a digital X-ray image.

In this chapter we review some basic concepts of simulation of X-ray images that can be used to understand other complex and more realistic approaches such as [4, 5]. In Sect. 7.2, we give the simple (geometric and imaging) models that can be used in the simulation. In Sect. 7.3, we explain the basic simulation principles providing some Matlab examples that the reader can run and follow. In Sect. 7.4, we address some techniques of simulated defects. The simulated X-ray images can be used to assess the performance of a computer vision method for automated defect recognition. Examples of simulated defects in castings and welds are also given.

7.2 Modelling

In this section we will explain the geometric model and the imaging model that we will use in the simulation.

7.2.1 Geometric Model

The model is based on a theoretical approach of Chap. 3 and follows the diagram of Fig. 7.1. The reader will be referred to the corresponding Sects. 3.2.4 and 3.3 to see the details.

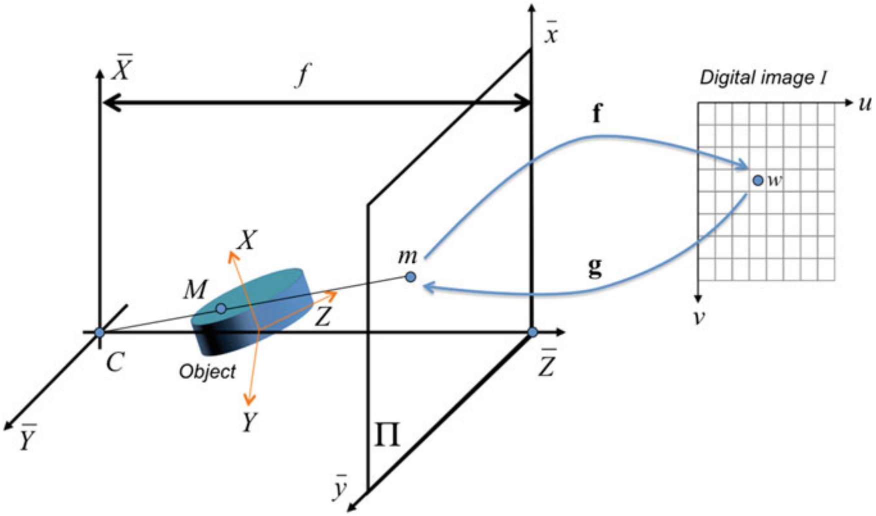


Fig. 7.1 Simplified geometric model taken from Fig. 3.6

As explained in Fig. 3.6, a 3D point M can be represented in world coordinate system $(\bar{X}, \bar{Y}, \bar{Z})$ as $\bar{\mathbf{M}} = [\bar{X}\ \bar{Y}\ \bar{Z}\ 1]^{\mathsf{T}}$ or in object coordinate system (X, Y, Z) as $\mathbf{M} = [X\ Y\ Z\ 1]^{\mathsf{T}}$ in homogenous coordinates. There is an Euclidean 3D → 3D transformation defined by (3.10):

$$\bar{\mathbf{M}} = \mathbf{HM} \tag{7.1}$$

where $\mathbf{H}$ is a 4×4 matrix that includes the rotation $\mathbf{R}$ and translation $\mathbf{t}$ between both coordinate systems (3.11):

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{7.2}$$

Point M is projected into projection plane Π as point m using a perspective transformation. Applying intercept theorem (3.14), the coordinates of m in this 2D system are $(\bar{x}, \bar{y})$, with

$$\bar{x} = f\bar{X}/\bar{Z} \qquad \text{and} \qquad \bar{y} = f\bar{Y}/\bar{Z}. \tag{7.3}$$

This equation can be rewritten as (3.17): $\lambda\mathbf{m} = \mathbf{PM}$, where λ is a scale factor $\lambda \neq 0$. Again m is given in homogeneous coordinates $\bar{\mathbf{m}} = [\bar{x}\ \bar{y}\ 1]^{\mathsf{T}}$. Perspective matrix $\mathbf{P}$ depends on the focal length f. Thus, a point M given in (X, Y, Z) is projected as point m in $(\bar{x}, \bar{y})$ as in (3.18):

$$\lambda \underbrace{\begin{bmatrix} \bar{x} \\ \bar{y} \\ 1 \end{bmatrix}}_{\bar{\mathbf{m}}} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_{\mathbf{M}}, \tag{7.4}$$

where $\mathbf{0} = [0\ 0\ 0]$. In image coordinate system (u, v), point m is viewed as point w that can be represented in homogenous coordinates as $\mathbf{w} = [u\ v\ 1]^{\mathsf{T}}$. The transformation $\bar{\mathbf{m}} \rightarrow \mathbf{w}$ is defined by function $\mathbf{f}$, and the back transformation $\mathbf{w} \rightarrow \bar{\mathbf{m}}$ by function $\mathbf{g}$. In linear case, where no distortion takes place, transformation $\mathbf{f}$ can be defined by (3.24):

$$\mathbf{f}: \quad \underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{bmatrix} \bar{x} \\ \bar{y} \\ 1 \end{bmatrix}}_{\bar{\mathbf{m}}}. \tag{7.5}$$

where scale factors (k_u, k_v) and a translation of the origin (u_0, v_0) are considered. In this model, we assume the skew factor s can be neglected. In this simplified linear model,

$$\mathbf{w} = \mathbf{f}(\bar{\mathbf{m}}) = \mathbf{K}\bar{\mathbf{m}} \ \text{ and } \ \bar{\mathbf{m}} = \mathbf{g}(\mathbf{w}) = \mathbf{K}^{-1}\mathbf{w}. \tag{7.6}$$

If the transformation $\bar{\mathbf{m}} \rightarrow \mathbf{w}$ is non linear, a non linear model for $\mathbf{f}$ and for $\mathbf{g}$ must be used (see examples in Sects. 3.3.2 and 3.3.3). In this section, we will assume a linear model only. Thus, a point $\mathbf{M}$ in object coordinate system is viewed as point $\mathbf{w}$ in image coordinate system as:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{7.7}$$

or using matrix notation:

$$\lambda \mathbf{w} = \mathbf{KPHM}. \tag{7.8}$$

If we have a pixel w in image coordinate system given by $\mathbf{w} = [u\ v\ 1]^{\mathsf{T}}$, and we want to estimate the X-ray beam $\langle C, m \rangle$ that defines w, we have to find the coordinates $\bar{\mathbf{m}} = [\bar{x}\ \bar{y}\ 1]^{\mathsf{T}}$ in projection coordinate systems using back transformation $\mathbf{g}$, i.e., $\bar{\mathbf{m}} = \mathbf{g}(\mathbf{w}) = \mathbf{K}^{-1}\mathbf{w}$ in linear case. Thus, the X-ray beam is defined by points $(\bar{X}, \bar{Y}, \bar{Z})$ that fulfill:

$$\bar{X} = \bar{x}\bar{Z}/f \qquad \text{and} \qquad \bar{Y} = \bar{y}\bar{Z}/f. \tag{7.9}$$

Equations (7.7) and (7.9) will be used by the simulation in the following sections.

7.2.2 X-ray Imaging

As we have already learned in Sect. 1.5, the intensity of X-ray penetrating radiation is modified by its passage through material and by discontinuities in the material. An example of this phenomenon is illustrated in Figs. 1.6 and 1.13.

Two properties of the X-rays are used to model the X-ray imaging process: (i) X-rays are differentially absorbed and (ii) X-rays travel in straight lines. The absorption can be macroscopically modeled using the exponential attenuation law for X-rays (1.2):

$$\varphi = \varphi_0 e^{-\mu x}, \tag{7.10}$$

where φ_0 is the incident intensity of radiation, φ the transmitted intensity, x thickness of the specimen and μ is a constant known as the *linear absorption coefficient* of the material under test with dimension cm^{-1}. Coefficient μ depends on the material and the X-ray energy. As an example, Fig. 1.6 illustrates the linear absorption coefficient for aluminum plotted against X-ray energy. Typically, X-ray testing of aluminum castings uses energy values between 50 and 150 keV [8]. Coefficient μ can be modeled as a fourth degree polynom [9]:

$$\mu \approx \sum_{i=0}^{4} \theta_i E^i \qquad \text{for } 50\,\text{keV} \leq E \leq 150\,\text{keV}, \tag{7.11}$$

with

$$\theta = (6.00, -0.210, -0.00304, -1.97 \times 10^{-5}, 4.72 \times 10^{-8}).$$

A flaw such as a cavity can be simulated as a material with no absorption. In Fig. 1.6 this simulation is shown schematically. An X-ray beam penetrates an object which has a cavity with thickness d. In this case, from (7.10) the transmitted radiation φ is given by:

$$\varphi = \varphi_0 e^{-\mu(x-d)}, \tag{7.12}$$

where we assume that the absorption coefficient of the cavity is zero. If the flaw is an incrusted material, its absorption coefficient μ_d must be considered.

In the example of Fig. 7.2, we have three materials with different linear absorption coefficients μ_1, μ_2 and μ_3. The thickness in direction of the X-ray beam is x_1, x_2 and x_3 for each material. It is worth noting that the thickness depends on the projection beam $\langle C, m \rangle$, i.e., for different locations of m, different thicknesses will be obtained. A simplified model can be used for different thicknesses and materials (1.4):

$$\varphi = \varphi_0 \exp\left(-\sum_i \mu_i x_i\right). \tag{7.13}$$

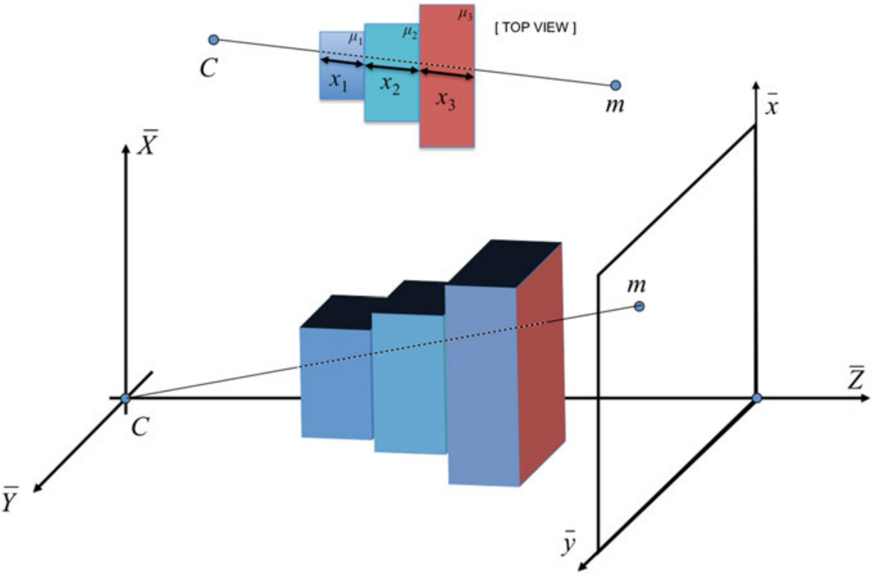


Fig. 7.2 Example of an X-ray beam that passes through three materials. The total path length through material i in direction of the beam is x_i, and the linear absorption coefficient of each material es μ_i for $i = 1, 2, 3$

Nevertheless, it is worth mentioning that μ and φ_0 depends on the energy E. In addition, if we want to compute the gray value of a pixel (u, v) of a simulated X-ray image, as a pixel is rather a square than a point, we must take into account the solid angle that corresponds to the pixel observed from the source point.

$$\varphi(R) = \varphi_0(E)\Delta\Omega \exp\left(-\sum_i \mu_i(E)x_i\right) \tag{7.14}$$

where $\varphi_0(E)$ is the incident radiation intensity of energy E, $\Delta\Omega$ is the solid angle that corresponds to region R of the image (e.g., pixel (u, v)) observed from the source point, $\mu_i(E)$ designates the attenuation coefficient associated with the material i at energy E, and x_i the total path length through material i. The X-ray source can be modeled as a raster of point sources, rays from every source point are traced to all pixels of the simulated image. The final simulated image will be an addition of each single simulation, one for each energy and each source point [1].

Finally, a linear transformation from incident energy to gray value is considered:

$$I = A\varphi + B, \tag{7.15}$$

where A and B are the linear parameters of I.

7.3 Basic General Simulation

In this section, we will see a basic approach to simulate an X-ray image of a 3D object based on *voxels*, i.e., the volume of the object is discretized in very small volume elements. In case, the volume is defined as a *polygon mesh*, the mesh can be *voxelized* [10].

A simple way to simulate an X-ray image of a 3D object is by modeling the object as a set of *voxels* as illustrated in Fig. 7.3. Thus, each voxel has a 3D location (X, Y, Z) and can have a linear absorption coefficient $V = \mu$. The value '0' for a voxel means that the voxel does not belong to the object (see cyan voxels in Fig. 7.3). In case the whole object is of the same material, e.g., an aluminum wheel, the value of a voxel can be a binary value: '0' for a voxel that does not belong to the object and '1' otherwise (see red voxels in Fig. 7.3).

In this approach, we assume that there are P voxels that belong to the object. The kth voxel, for $k = 1 \dots P$, is defined by its linear absorption coefficient $V_k = \mu_k > 0$ and its location in object coordinate system as $\mathbf{M}_k = [X_k\ Y_k\ Z_k\ 1]^\mathsf{T}$ in homogeneous coordinates. Using (7.8), we can obtain $\mathbf{w}_k = [u_k\ v_k\ 1]^\mathsf{T}$, the coordinates in image coordinate systems of each projected voxel:

$$\lambda_k \mathbf{w}_k = \mathbf{KPHM}_k. \tag{7.16}$$

A great advantage of using homogeneous coordinates is that the projection of all points $\mathbf{M}_k$ can be done with only one multiplication: $\mathbf{W} = \mathbf{KPHM}$, where $\mathbf{M}$ is a $4 \times P$ matrix $\mathbf{M} = [\mathbf{M}_1\ \mathbf{M}_2 \cdots \mathbf{M}_P]$ and $\mathbf{W}$ is a $3 \times P$ matrix $\mathbf{W} = [\lambda_1 \mathbf{w}_1\ \lambda_2 \mathbf{w}_2 \cdots \lambda_P \mathbf{w}_P]$.

According to (7.13), an X-ray beam passes through different materials with different levels of thickness. In our model, each voxel can be considered as an element with a linear absorption coefficient μ_i and a thickness x_i (in direction of the X-ray beam). It is simple to accumulate in a region R of the image the contribution of all voxels that are in the corresponding X-ray beam as illustrated in Fig. 7.4. In this

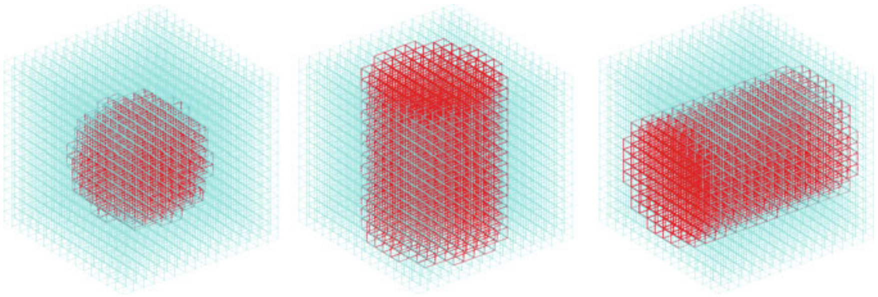


Fig. 7.3 Object modeling using voxels. In these examples (a *sphere* and *two cylinders*) there are 15^3 voxels. The radius of each object is 5. The *red* voxels belong to the object

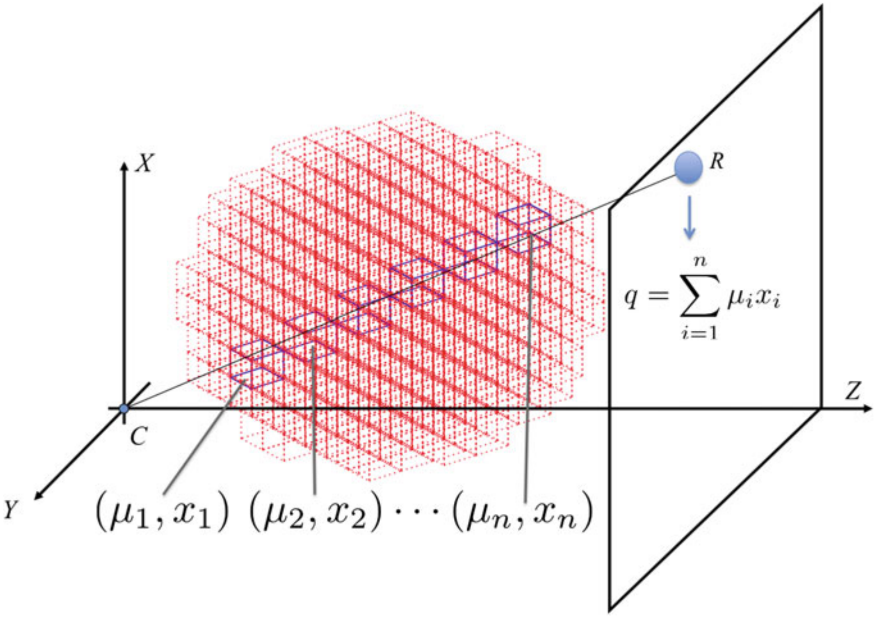

X
R
$q = \sum_{i=1}^{n} \mu_i x_i$
Z
C
Y
(μ_1, x_1) $(\mu_2, x_2) \cdots (\mu_n, x_n)$

Fig. 7.4 The contribution of all voxels aligned to the X-ray beam in a region R can be modeled as $q(R) = \sum_i \mu_i x_i$. The example shows the voxels of a *sphere* that are in the X-ray beam

example, we show the voxels that belong to a spherical object in red, and those voxels that contribute to region R in blue. For region R of the image we can compute $q(R) = \sum_i \mu_i x_i$. Finally, the gray value of this region is modeled using $q(R)$ and Eqs. (7.13) and (7.15) as:

$$I(R) = A\varphi_0 e^{-q(R)} + B. \tag{7.17}$$

There are two different ways to obtain the simulated X-ray image **I**:

- **From pixels to voxels:** In order to simulate **I** of $N \times M$ pixels, we can estimate the intensity of each pixel (u, v), for $u = 1 \ldots N$ and $v = 1 \ldots M$ as follows:

1. Each pixel (u, v) defines a point $(\bar{x}, \bar{y})$ in the projection coordinate system as explained in Sect. 7.2.1.
2. Point $(\bar{x}, \bar{y})$ defines a specific X-ray beam according to (7.9). If there is an intersection of the X-ray beam with the 3D object, follow the next steps.
3. The beam passes through n voxels of the object, that means there are corresponding linear absorption coefficients of each voxel (μ_i) and thickness (x_i) for $i = 1 \ldots n$. The absorption linear coefficient μ_i can be obtained from the corresponding voxel value. The thickness x_i can be estimated as the line segment length of the intersection of the corresponding X-ray beam that passes through the voxel with the cube defined by the voxel.
4. The contribution $q(u, v) = \sum_i \mu_i x_i$ is computed.

5. Using (7.17) the gray value for each pixel can be estimated. In this approach, the region R corresponds to the area defined by pixel (u, v).

- **From voxels to pixels:** In order to simulate image $\mathbf{I}$, we first define an image $\mathbf{Q}$ as a matrix with the same size of $\mathbf{I}$, i.e., $N \times M$ pixels. All pixels of $\mathbf{Q}$ are initialized to zero. Afterwards, we can deal with the projection of each voxel k, for $k = 1 \dots P$ as follows:

1. The kth voxel located at $\mathbf{M}_k$ is projected using (7.16), and coordinates (u_k, v_k) in the image coordinate system are obtained.
2. The contribution of kth voxel to our image is $q_k = \mu_k x_k$. The absorption linear coefficient μ_k can be obtained from the voxel value V_k. The thickness x_k can be estimated as the line segment length of the intersection of the corresponding X-ray beam that passes through the center of the voxel with the cube defined by the voxel.
3. The value q_k is added in those pixels (u, v) of image $\mathbf{Q}$ that are neighbors to (u_k, v_k).

Finally, using (7.17) the gray value for each pixel can be estimated. In this approach, the region R corresponds to the area defined by pixel (u, v).

In order to show a simple simulation of an X-ray image of a 3D object, we give some details of the second approach in the following example.

Matlab Example 7.1 In this example, we simulate the X-ray image of a homogeneous material using voxels. The implementation corresponds to the method 'from voxels to pixels' outlined in this section. The 3D matrix V contains 1800 × 1800 × 1800 binary voxels created by command Xobjvoxels (see Appendix B) of $\mathbb{X}$VIS Toolbox.

Listing 7.1 : Simulation of an X-ray image of 3D object

```
% WheelSimulation.m
close all
Nv = 1800;                                  % Number of voxels in each direction
V  = Xobjvoxels(1,Nv,1);                    % V has Nv^3 voxels od a wheel with a flaw

% Transformation (x,y)->(u,v)
u0 = 235; v0 = 305; ax = 1.1; ay = 1.1;     % translation and scaling
K  = [ax 0 u0; 0 ay v0; 0 0 1];             % transformation matrix

% Transformation (Xb,Yb,Zb)->(x,y)
f  = 1500;                                  % focal length
P  = [f 0 0 0; 0 f 0 0; 0 0 1 0];

% Transformation (X,Y,Z)->(Xb,Yb,Zb)
R  = Xmatrixr3(0.5,0.1,0.6);                % rotation
t  = [-120 -120 1000]';                     % translation
H  = [R t; 0 0 0 1];                        % transformation matrix

PP = K*P*H;                                 % complete projection matrix

Q = Xsimvoxels(512,512,V,7,PP);             % simulation

imshow(exp(-0.0001*Q),[])                   % display simulation
```

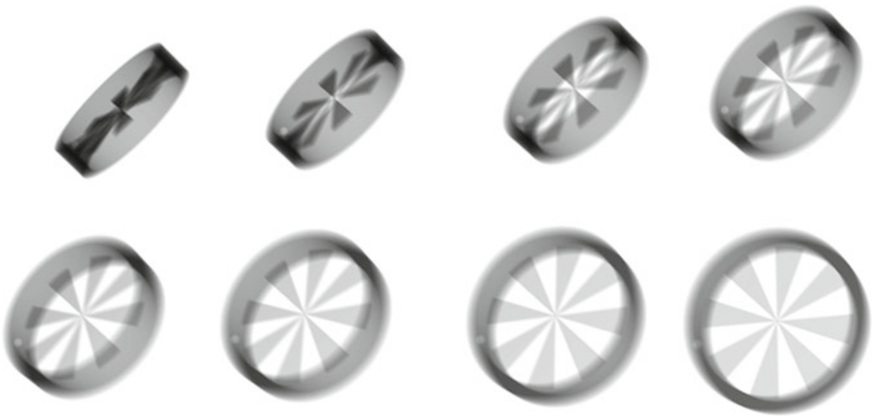


Fig. 7.5 Simulation of a wheel in eight different positions (→ Example 7.1)

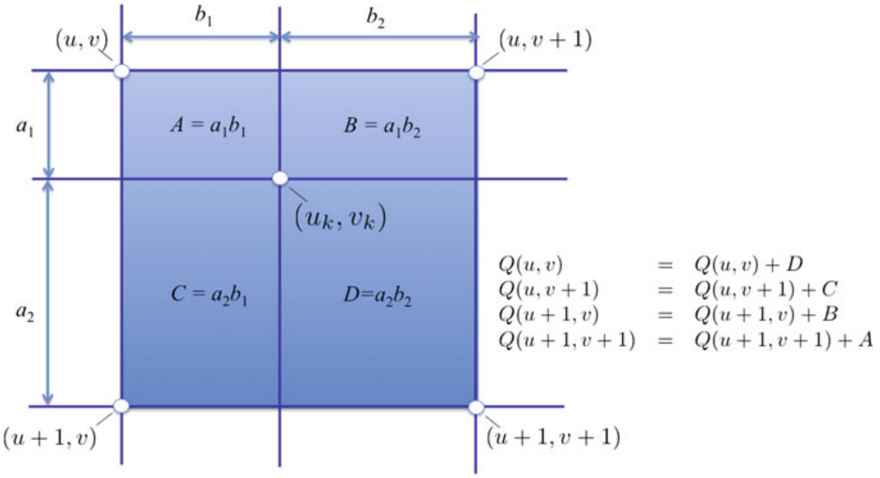


Fig. 7.6 Since pixel (u_k, v_k) does not exist, it is impossible to add the contribution $\mu_k x_k$ to this pixel. For this reason, the contribution is distributed in its four neighbor pixels according to their opposite areas A, B, C, D (note that $A+B+C+D=1$). In our simplified model, $x_k = 1$ and μ_k is constant, that means that the contribution of each voxel is constant (→ Example 7.1)

The output of this code is illustrated in Fig. 7.5, where eight different positions are shown. The eight positions were obtained varying the rotation angles of matrix R. In this example, the X-ray image was simulated using command Xsimvoxels (see Appendix B) of 𝕏vis Toolbox. In this implementation we assume that the thickness of a voxel (x_k) is always 1. This is not true, however, for homogenous material, when μ_k is constant, $x_k = 1$ is a good estimation of the average value. The weighted distribution explained in step 3 is implemented as shown in Fig. 7.6. Also the reader can simulate an X-ray image using files in the STL format (a standard for a polygon mesh).[1] In this case, we have to specify the name of the STL file (`'sample.stl';`), and change the command for the simulation in our example by:

```
Q = Xsimvoxels(512,512,'sample.stl',7,PP,400);
```

The last parameter (`400`) indicates that the conversion from STL to voxels will be into a cube of 400 × 400 400 voxels.[2] □

7.4 Flaw Simulation

Generally, the automatic defect recognition consists of a binary classification, where a decision is performed about whether or not an initially identified hypothetical defect in an image is in fact a defect. Unfortunately, in real automatic flaw detection problems there are a reduced number of flaws in comparison with the large number of non-flaws. This skewed class distribution seriously limits the application of classification techniques [11]. Usually, the performance of an inspection method can be assessed on a few images, and an evaluation of a broader and more representative data base is necessary. A good way of assessing the performance of a method for inspecting castings is to examine simulated data. This evaluation allows one the possibility of tuning the parameters of the inspection method and of testing how well the method works in critical cases.

Among the NDT community there are two groups of methods to obtain this simulated data: invasive and non-invasive methods. Table 7.1 summarizes their most important properties.

Invasive Methods

In the invasive methods, discontinuities are produced in the test object artificially. There are two published invasive methods: (i) drilling holes on the object surface [12] (see Fig. 7.7), and (ii) designing a test piece with small spherical cavities [13] (see Fig. 7.8). Usually, the first technique drills small holes (e.g., Ø $= 1.0 \sim 4.0$ mm) in positions of the casting which are known to be difficult to detect. In the second technique, a sphere is produced for example by gluing together two aluminum pieces containing half-spherical concavities. The principal advantage of these methods is that the discontinuity image is real. However, the disadvantages are: (i) it is impossible to introduce concavities in the middle of the object without destroying it, and (ii) concavities like cracks are practically impossible to reproduce.

Non-Invasive Methods

In the non-invasive methods, X-ray images are generated or modified without altering the test object. There are three widespread approaches that produce this

[1] http://en.wikipedia.org/wiki/STL_(file_format).

[2] In this example we used the Matlab code for mesh voxelization available on http://www.mathworks.com/matlabcentral/fileexchange/27390-mesh-voxelisation.

Table 7.1 Methods for simulation of defects

	Method	Description	Advantages	Disadvantages
Invasive	Drilling holes	It drills holes on the surface of the test object	• Real X-ray image with real defects	• Cracks cannot be produced • X-ray imaging system is required
	Spherical cavities	It produces defects inside of the test object by putting together two parts with cavities	• Real X-ray image with real defects	• It destroys the test object • Cracks cannot be produced • X-ray imaging system is required
Non invasive	Mask superimposition	It modifies the original gray value of the image by multiplying it with a factor	• Real X-ray image with simulated defects • Easy implementation	• Simulated defects differ significantly from the real ones • X-ray imaging system is required
	Full-CAD	It simulates the X-ray imaging process by projecting a CAD model including a defect	• No real X-ray imaging system is required • Defects and object can be modelled in 3D	• No real X-ray image of the test object • Sophisticated computer package • Time consuming
	Flaw-CAD	It modifies the original gray value of the image by superimposing the projection of a CAD model of a flaw	• Real X-ray image with simulated defects • No time consuming • Defects can be modelled in 3D	• X-ray imaging system is required

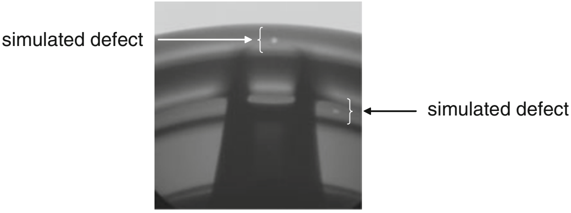


Fig. 7.7 Two defects generated using drilling holes

Fig. 7.8 Two generated defects using spherical cavities [13]

simulated data [14]: (i) mask superimposition, (ii) CAD models for casting and flaw and (iii) CAD models for flaws only. In this section, they will be described in further detail.

7.4.1 Mask Superimposition

The first technique attempts to simulate flaws by superimposing masks with different gray values onto real X-ray images [8, 15, 16]. This approach is quite simple, as it neither requires a complex 3D model of the object under test nor of the flaw. It also provides a real X-ray image with real disturbances, albeit with simulated flaws.

In this technique, the original gray value I_o of a pixel (u, v) of an X-ray image is altered by:

$$I_n(u,v) = I_o(u,v)\,(1+M(u-u_0, v-v_0))\,, \tag{7.18}$$

with $I_n(u, v)$ the new gray value and M the mask that is centered on pixel (u_0, v_0), where $M(i, j)$ is defined in the interval $-\frac{n}{2} \le i \le \frac{n}{2}$ and $-\frac{m}{2} \le j \le \frac{m}{2}$. Three typical masks are shown in 7.9.

Matlab Example 7.2 In this example, we simulate three different flaws in an aluminum casting using Gaussian, square and circle masks.

Listing 7.2 : Simulation of a defects using superimposed masks

```
% MaskFlawSimulation.m
close all
I   = imread('wheel.png');                              % Input image
h1 = fspecial('gaussian',27,3); h1 = h1/max(h1(:))*0.7; % Gaussian mask (a)
i1 = 160; j1 = 110;                                     % Location of the mask
J1 = Xsimmask(I,h1,i1,j1,0);                            % Simulation
h2 = ones(17,17)*0.4;                                   % Square mask (b)
i2 = 200; j2 = 120;                                     % Location of the mask
J2 = Xsimmask(J1,h2,i2,j2,0);                           % Simulation
h3 = fspecial('disk',7);h3 = h3/max(h3(:))*0.4;         % Circle mask (c)
i3 = 240; j3 = 130;                                     % Location of the mask
J3 = Xsimmask(J2,h3,i3,j3,0);                           % Simulation
imshow(J3)                                              % Output image
text(j1+15,i1,'a','fontsize',16,'color','w');
text(j2+15,i2,'b','fontsize',16,'color','w');
text(j3+15,i3,'c','fontsize',16,'color','w');
```

The output of this code is illustrated in Fig. 7.9, where three different defects are shown in. In this example, the X-ray image was simulated using command Xsimmask (see Appendix B) of 𝕏VIS Toolbox. □

7.4.2 CAD Models for Object and Defect

The second approach simulates the entire X-ray imaging process [17, 18]. In this approach, characteristics of the X-ray source, the geometry, and material properties

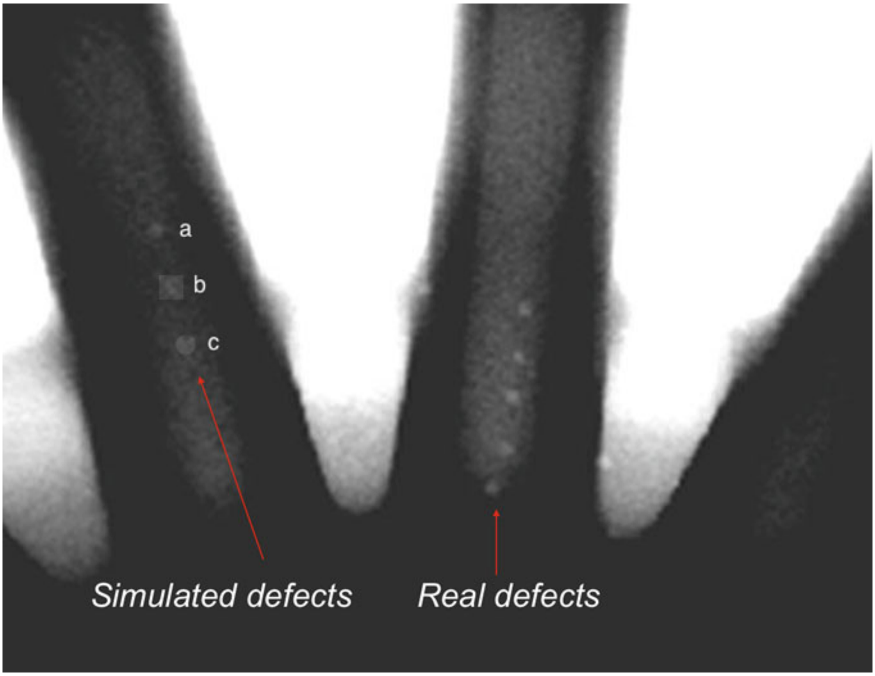


Fig. 7.9 Flaw simulation using **a** *Gaussian mask*, **b** *square*, and **c** *circle*. As we can see, the Gaussian mask achieves the best simulation (→ Example 7.2)

of objects and their defects, as well as the imaging process itself are modeled and simulated independently. Complex objects and defect shapes can be simulated using CAD models.

The principle of the simulation is shown in Fig. 7.10. The X-ray may intersect different parts of the object. The intersection points between the modeled object with the corresponding X-ray beam that is projected into pixel (u, v) are calculated for each pixel (u, v) of the simulated image as explained in Sect. 7.3.

Some complicated 3D flaw shapes are reported in [17]. The defect model is coupled with a CAD interface yielding 3D triangulated objects. Other kinds of flaws like cracks can also be obtained using this simulation technique.

Although this approach offers excellent flexibility for setting the objects and flaws to be tested, it has three disadvantages for the evaluation of the inspection methods' performance: (i) the X-ray image of the object under test is simulated (it would be better if we could count on real images with simulated flaws); (ii) the simulation approach is only available when using a sophisticated computer package; (iii) the computing time is expensive.

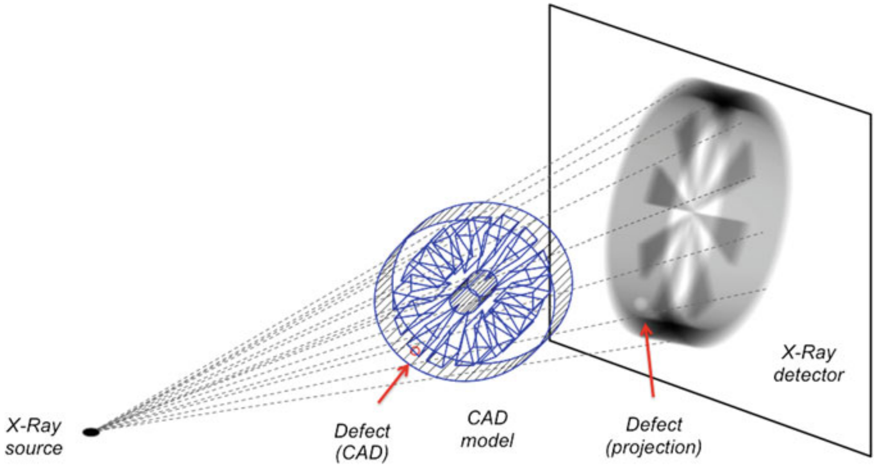


Fig. 7.10 X-ray image simulation using CAD models

7.4.3 CAD Models for Defects Only

This approach simulates only the flaws and not the whole X-ray image of the object under test [9]. This method can be viewed as an improvement of the first mentioned technique (Sect. 7.4.1) and the 3D modeling for the flaws of the second one (Sect. 7.4.2). In this approach, a 3D modeled flaw is projected and superimposed onto real X-ray images of a homogeneous object according to the exponential attenuation law for X-rays (7.10).

As explained in Sect. 7.2.2, the gray value I of a digital X-ray image can be expressed as a linear function of the transmitted radiation φ:

$$I(x) = A\varphi(x) + B, \tag{7.19}$$

where

$$\varphi(x) = \varphi_0 e^{-\mu x}, \tag{7.20}$$

and A and B are the linear parameters of I, and x the thickness of the object under test.

Now, we investigate what happens if the penetrated object has a cavity, the thickness of which is d as shown in Fig. 1.13 and its absorption coefficient $\mu' \approx 0$. In this case, from (7.20) the transmitted radiation is given by:

$$\varphi(x-d) = \varphi_0 e^{-\mu(x-d)} = \varphi(x)e^{\mu d}. \tag{7.21}$$

The gray value registered is calculated then from (7.21) and (7.19) as:

$$I(x-d) = A\varphi(x)e^{\mu d} + B. \tag{7.22}$$

Substituting the value of $A\varphi(x)$ from (7.19) we see that (7.22) may be written as:

$$I(x-d) = I(x)e^{\mu d} + B(1-e^{\mu d}). \tag{7.23}$$

Parameter B can be estimated as follows: The maximal gray value ($\mathrm{I}_{\mathrm{max}}$) in an X-ray image is obtained when the thickness is zero. Additionally, the minimal gray value ($\mathrm{I}_{\mathrm{min}}$) is obtained when the thickness is $\mathrm{x}_{\mathrm{max}}$. Substituting these values in (7.19), it yields:

$$\begin{cases} \mathrm{I}_{\mathrm{max}} = A\varphi_0 + B \\ \mathrm{I}_{\mathrm{min}} = A\varphi_0 e^{-\mu x_{max}} + B \end{cases} . \tag{7.24}$$

From these equations, one may compute the value for B:

$$B = \mathrm{I}_{\mathrm{max}} - \Delta I/(1-e^{-\mu x_{max}}) \,, \tag{7.25}$$

where $\Delta I = \mathrm{I}_{\mathrm{max}} - \mathrm{I}_{\mathrm{min}}$. Usually, $\mathrm{I}_{\mathrm{max}}$ and $\mathrm{I}_{\mathrm{min}}$ are 255 and 0 respectively. For these values, B can be written as:

$$B = 255/(1-e^{\mu \mathrm{x}_{\mathrm{max}}}) \,. \tag{7.26}$$

This means that the gray value of the image of the cavity is:

$$I(x-d) = I(x)e^{\mu d} + 255\frac{1-e^{\mu d}}{1-e^{\mu \mathrm{x}_{\mathrm{max}}}}. \tag{7.27}$$

Using Eq. (7.27), we can alter the original gray value of the X-ray image $I(x)$ to simulate a new image of a flaw $I(x-d)$. A 3D flaw can be modeled, projected and superimposed onto a real radioscopic image. The new gray value of a pixel, where the 3D-flaw is projected, depends only on four parameters: (a) Original gray value $I(x)$; (b) the linear absorption coefficient of the examined material μ; (c) the length of the intersection of the 3D-flaw with the modeled X-ray beam d, that is projected into the pixel; and (d) the maximal thickness observable in the radioscopic image x_{max}.

Now, we will explain in further details how a 3D defect, namely an ellipsoid, is projected onto an X-ray image [9]. Using this tool a simulation of an ellipsoidal flaw of any size and orientation can be made anywhere in the casting. This model can be used for flaws like blowholes and other round defects. Two examples are shown in Fig. 7.11. The simulated flaws appear to be real due to the irregularity of the gray values.

This technique presents two advantages: simulation is better than with the first technique; and with respect to the second, this technique is faster given the reduced computational complexity. However, the model used in this method has four simplifications that were not presumed in the second simulation technique: (i) the X-ray

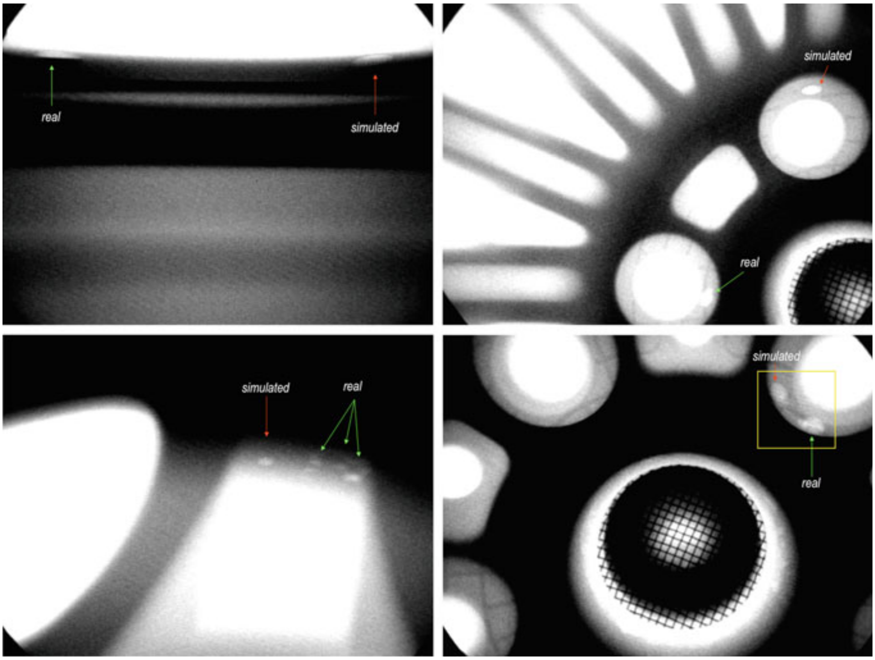


Fig. 7.11 Simulated ellipsoidal flaws using CAD models of a defect only. See details in Table 7.2. 3D profile of a *yellow square* is shown in Fig. 7.14 (Color figure online)

source is assumed as a source point; (ii) there is no consideration of noise in the model; (iii) there is no consideration of the solid angle $\Delta\Omega$ of the X-ray beam that is projected onto a pixel; and (iv) the spectrum of the radiation source is monochromatic.

In our approach we follow the geometric model illustrated in Fig. 7.12. This model is very similar to the geometric model we learned in Sect. 7.2.1, however, it includes a new coordinate system (X', Y', Z') attached to the center of the ellipsoid that is modeled as:

$$\frac{X'^2}{a^2} + \frac{Y'^2}{b^2} + \frac{Z'^2}{c^2} = 1, \tag{7.28}$$

where a, b and c are the half-axes of the ellipsoid as shown in Fig. 7.12. The location of the ellipsoid relative to the object coordinate system is defined by a 3×3 rotation matrix $\mathbf{R}_e$ and a 3×1 translation vector $\mathbf{t}_e$. They can be arranged in a 4×4 matrix $\mathbf{H}_e$ as in Eq. (7.2). Using (7.1), the coordinates in the ellipsoid coordinate system (X', Y', Z') can be expressed in the world coordinate system $(\bar{X}, \bar{Y}, \bar{Z})$ by:

$$\bar{\mathbf{M}} = \mathbf{H}\mathbf{H}_e\mathbf{M}', \tag{7.29}$$

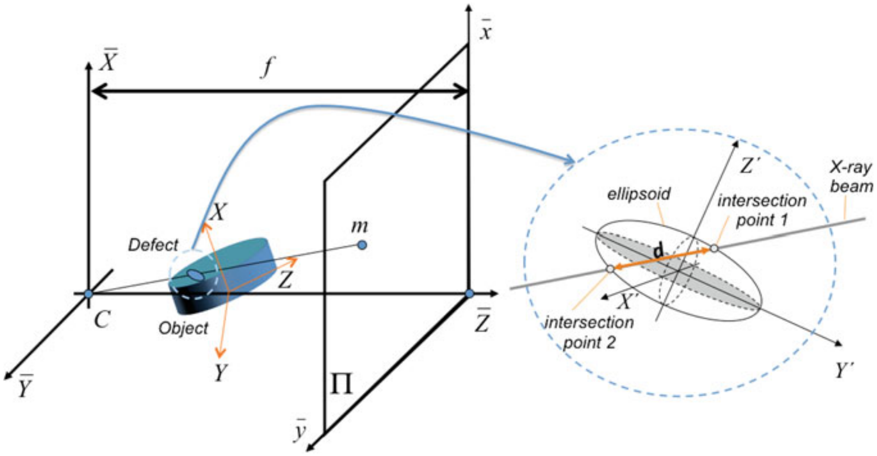


Fig. 7.12 Ellipsoid used by modeling a 3D flaw in coordinate system (X', Y', Z'). The two intersections of an X-ray beam with the surface of the ellipsoid define distance d

with $\mathbf{M}' = [X' \ Y' \ Z' \ 1]^T$ and $\bar{\mathbf{M}} = [\bar{X} \ \bar{Y} \ \bar{Z} \ 1]^T$. Now, we can write the ellipsoid in world coordinate system from (7.28) and (7.29) as:

$$\begin{aligned}&(s_{11}\bar{X} + s_{12}\bar{Y} + s_{13}\bar{Z} + s_{14})^2/a^2 + \\ &(s_{21}\bar{X} + s_{22}\bar{Y} + s_{23}\bar{Z} + s_{24})^2/b^2 + \\ &(s_{31}\bar{X} + s_{32}\bar{Y} + s_{33}\bar{Z} + s_{34})^2/c^2 = 1,\end{aligned} \tag{7.30}$$

where s_{ij} are the elements of the 4×4 matrix $\mathbf{S} = [\mathbf{H}\mathbf{H}_e]^{-1}$.

Suppose we have a pixel (u, v) of the X-ray image and we want to know if the X-ray beam, which produces a gray value in this pixel, intersects the modeled ellipsoid. Using $\mathbf{g}$, the inverse function of $\mathbf{f}$ (see (7.6)), we can calculate the corresponding coordinates of (u, v) in the projection coordinate systems $(\bar{x}, \bar{y})$:

$$\bar{\mathbf{m}} = \mathbf{g}(\mathbf{u}), \tag{7.31}$$

with $\mathbf{u} = [u \ v \ 1]^T$ and $\bar{\mathbf{m}} = [\bar{x} \ \bar{y} \ 1]^T$. Remember that for a linear perspective projection with no distortion, $\bar{\mathbf{m}} = \mathbf{K}^{-1}\mathbf{w}$, with $\mathbf{K}$ defined in (7.5). The X-ray beam in the world coordinate system is defined from (7.3) by:

$$\begin{cases} \bar{X} = x\bar{Z}/f \\ \bar{Y} = y\bar{Z}/f \end{cases} . \tag{7.32}$$

The intersection of the X-ray beam with the ellipsoid is shown in Fig. 7.12. A intersection point must satisfy (7.30) and (7.32) simultaneously. Substituting $\bar{X}$ and $\bar{Y}$ from (7.32) in (7.30) and after some slight rearranging we obtain:

$$A\bar{Z}^2 + B\bar{Z} + C = 0, \tag{7.33}$$

with

$$A = \frac{r_1^2}{a^2} + \frac{r_2^2}{b^2} + \frac{r_3^2}{c^2},$$

$$B = 2\left(\frac{r_1 s_{14}}{a^2} + \frac{r_2 s_{24}}{b^2} + \frac{r_3 s_{34}}{c^2}\right),$$

$$C = \frac{h_{14}^2}{a^2} + \frac{h_{24}^2}{b^2} + \frac{h_{34}^2}{c^2} - 1 \qquad \text{and}$$

$$r_i = s_{i1}\frac{x}{f} + s_{i2}\frac{y}{f} + s_{i3} \qquad \text{for} \qquad i = 1, 2, 3.$$

If $B^2 - 4AC > 0$ we obtain two intersection points of the X-ray beam with the ellipsoid given by:

$$\bar{X}_{1,2} = \frac{\bar{Z}_{1,2}}{f}x$$

$$\bar{Y}_{1,2} = \frac{\bar{Z}_{1,2}}{f}y$$

$$\bar{Z}_{1,2} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

The length of the X-ray beam that penetrates into the ellipsoid can be calculated as:

$$d = \sqrt{\left(\bar{X}_1 - \bar{X}_2\right)^2 + \left(\bar{Y}_1 - \bar{Y}_2\right)^2 + \left(\bar{Z}_1 - \bar{Z}_2\right)^2}, \tag{7.34}$$

that can be written as:

$$d = \frac{\sqrt{B^2 - 4AC}}{A}\sqrt{\frac{x^2}{f^2} + \frac{y^2}{f^2} + 1}. \tag{7.35}$$

The algorithm to simulate a flaw can be resumed as follows:

1. Calibration: Estimate the parameters of the mapping function $3D \rightarrow 2D$ (focal length f, matrix $\mathbf{H}$ and function $\mathbf{f}$).
2. Setting of X-ray imaging parameters: Define μ and x_{max} according to the energy used by the X-ray source.
3. Definition of the 3D flaw: Define the size of the flaw (parameters a, b and c) and the location of the flaw in the object (matrix $\mathbf{H}_e$).
4. Location of the superimposed area: Find the pixels (u, v) where the modeled 3D flaw is projected.[3]
5. Computation of intersection length d: For each determined pixel (u, v) find the length of the intersection between the X-ray beam and ellipsoid given by Eq. (7.35).
6. Change of the gray value: For each determined pixel (u, v) change the original gray value using (7.27).

Matlab Example 7.3 In this example, we simulate a defect as an ellipsoid using the method outlined in this section.

Listing 7.3 : Simulation of a defect in an aluminum casting

```
% EllipsoidSimulation.m
close all
I = imread('wheel.png');                     % input image

xmax = 400; var_mu = 0.1;                    % maximal thickness and mu

% Transformation (x,y)->(u,v)
u0 = 235; v0 = 305; ax = 1.1; ay = 1.1;       % translation and scaling
K  = [ax 0 u0; 0 ay v0; 0 0 1];               % transformation matrix

% Transformation (Xb,Yb,Zb)->(x,y)
f  = 1500;                                    % focal length

% Transformation (X,Y,Z)->(Xb,Yb,Zb)
R  = Xmatrixr3(0,0,0);                        % rotation
t  = [-36 40 1000]';                          % translation
S  = [R t; 0 0 0 1];                          % transformation matrix

% Transformation (Xp,Yp,Zp)->(X,Y,Z)
Re = Xmatrixr3(0,0,pi/3);                     % rotation
te = [0 0 0]';                                % translation
Se = [Re te; 0 0 0 1];                        % transformation matrix

% Ellipsoid's axes
abc = [3.5 4.5 2.5];                          % a, b, c

I = Xsimdefect(I,K,S*Se,f,abc,var_mu,xmax,1); % simulation
```

The output of this code is shown in Fig. 7.13. In this example, the simulated defects seems to be real. The defect was simulated using command Xsimdefect (see Appendix B) of 𝕏VIS Toolbox. □

[3] These pixels are defined where $B^2 - 4AC > 0$ in Eq. (7.33).

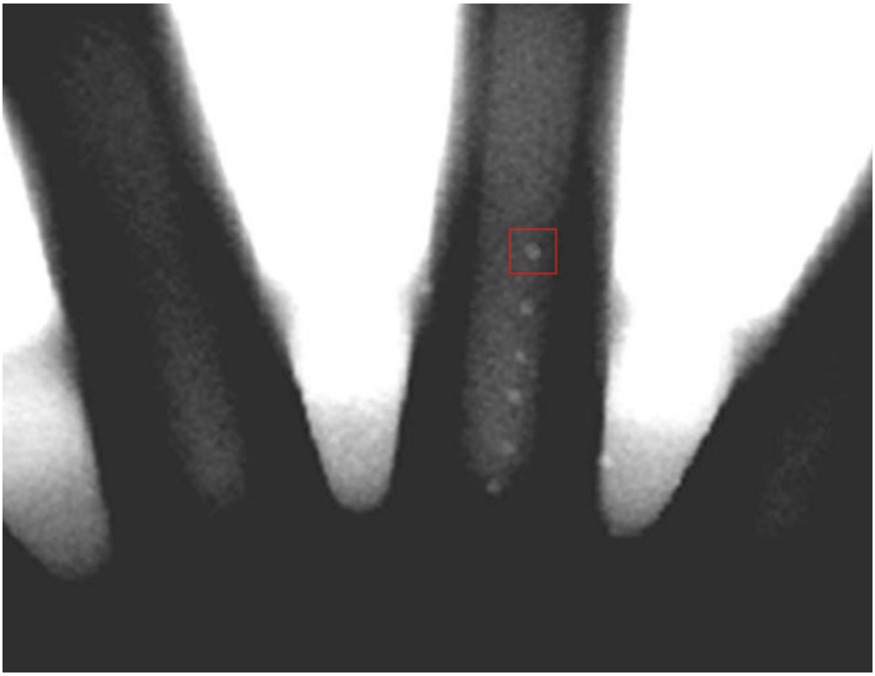


Fig. 7.13 Comparison of real defects with a simulated one (see *red square*) using proposed method (→ Example 7.3 (Color figure online))

In the following, the results of the simulation of flaws in cast aluminum wheels using our approach are presented. The dimensions of the wheels used in our experiments were approximately 48 cm in diameter and 20 cm in height. The focal length (distance between X-ray source and entrance screen of the image intensifier) was 90 cm. The projection model of the X-ray imaging system was calibrated using a hyperbolic model [19, 20].

In Fig. 7.11, experimental results on four X-ray images are shown. The values used to simulate the flaws in each image are summarized in Table 7.2. We can

Table 7.2 Values used in the simulations of Fig. 7.11

Image no.	E (keV)	μ (1/cm)	x_{max} (cm)	a (mm)	b (mm)	c (mm)
1	54	0.8426	4.0	8	2	4
2	58	0.7569	3.8	4	2	1.5
3	50	0.9500	4.5	4	2	1.7
4	57	0.7765	3.85	6	3	2.5

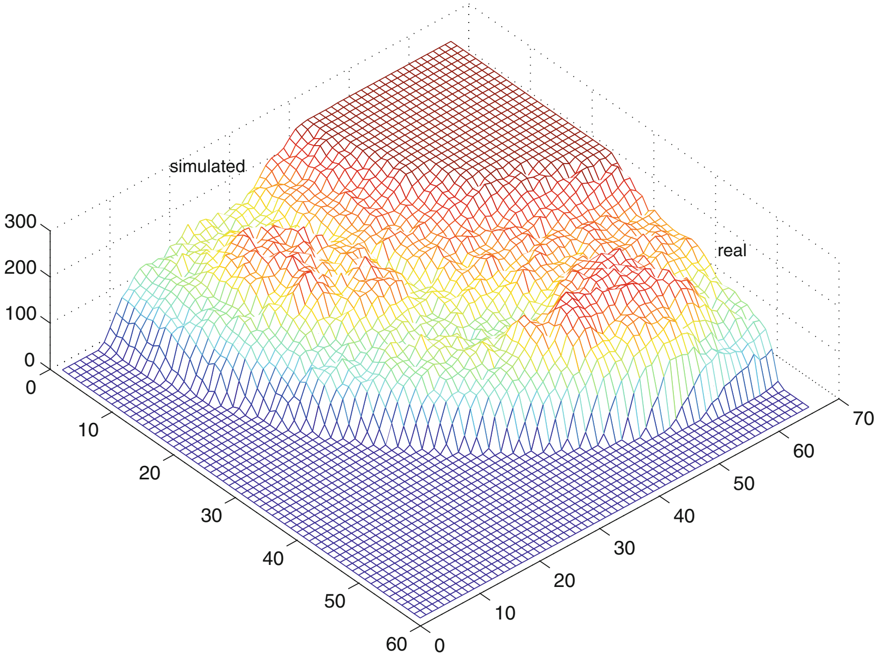


Fig. 7.14 3D plot of the *gray* values in the vicinity of flaws of the last X-ray image of Fig. 7.11

compare real and simulated flaws. It was shown that the simulation results are almost identical with real flaws. In Fig. 7.14 a 3D plot of the gray values in the vicinity of the flaws shown in last X-ray image of Fig. 7.11 is illustrated. Due to the irregularity of the gray values of the simulated flaw, it seems to be real.

Other complex defect shapes can be simulated using CAD models [21]. This general approach follows the block diagram of Fig. 7.15, where a 3D defect needs to be modeled as a manifold 3D mesh as illustrated in Fig. 7.16. Crack simulation can be obtained by superimposing a depth map computed from a single manifold (see for example Fig. 7.16a). However, a real crack corresponds to a more complex 3D representation. For this reason, we simulated another crack by superimposing several single cracks onto a real X-ray image. An example of this simulation is illustrated in Fig. 7.17. Due to the irregularity of the gray values of the simulated flaw, it can be seen that both real and simulated flaws show similar patterns.

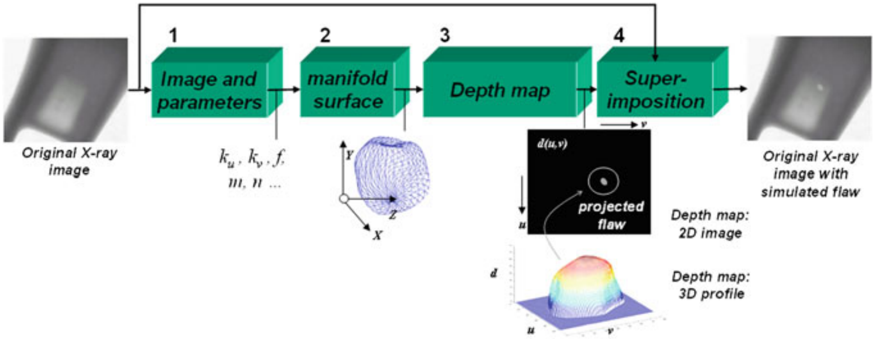


Fig. 7.15 Flaw simulation process using complex CAD models of the 3D defect

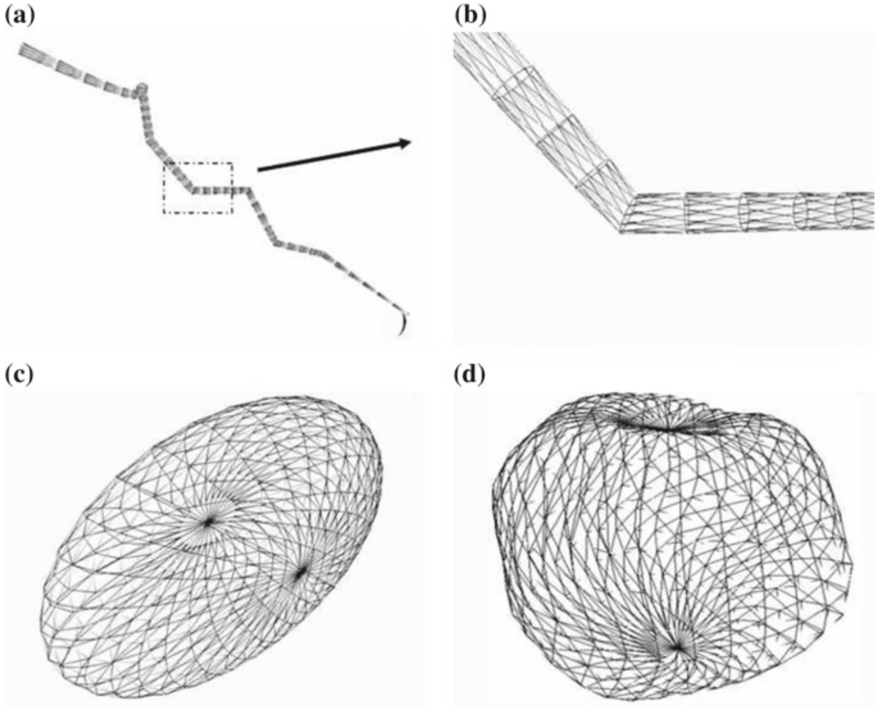


Fig. 7.16 Manifold surfaces from the 3D modeling software: **a** crack, **b** zoom of (**a**), **c** ellipsoid and **d** amorphous surface

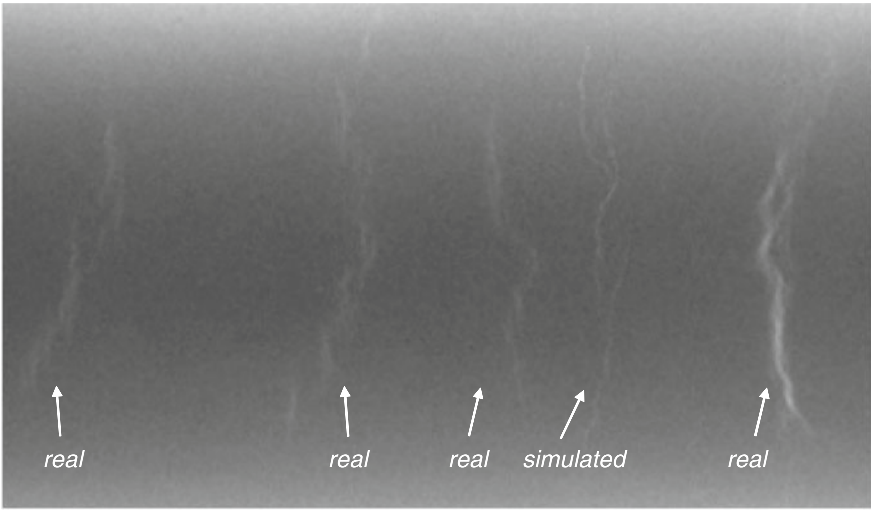


Fig. 7.17 Simulated and real cracks

7.5 Summary

To evaluate the performance of computer vision techniques, it is convenient to examine simulated data. This offers the possibility of tuning the parameters of the computer vision algorithm and to testing how it works in critical cases.

A simulation tool should model the physics of the X-ray formation (generation, interaction and detection) and handle complex 3D objects efficiently. State-of-the-art of computer modeling of X-ray testing methods are able to simulate different X-ray spectrum and X-ray source size, varied photon-matter interactions, and several X-ray detector responses. Thus, a computer simulator for X-ray testing should include the following modules: X-ray source model, ray-tracing engine, material data base, straight line attenuation model and detector model.

In this chapter we reviewed some basic concepts of simulation of X-ray images. We gave simple geometric and imaging models that can be used in the simulation. We explained the basic simulation principles and we addressed some techniques to simulate defects (that can be used to assess the performance of a computer vision method for automated defect recognition). The chapter has some Matlab examples that the reader can run and follow. Examples of simulated defects in castings and welds are also given.

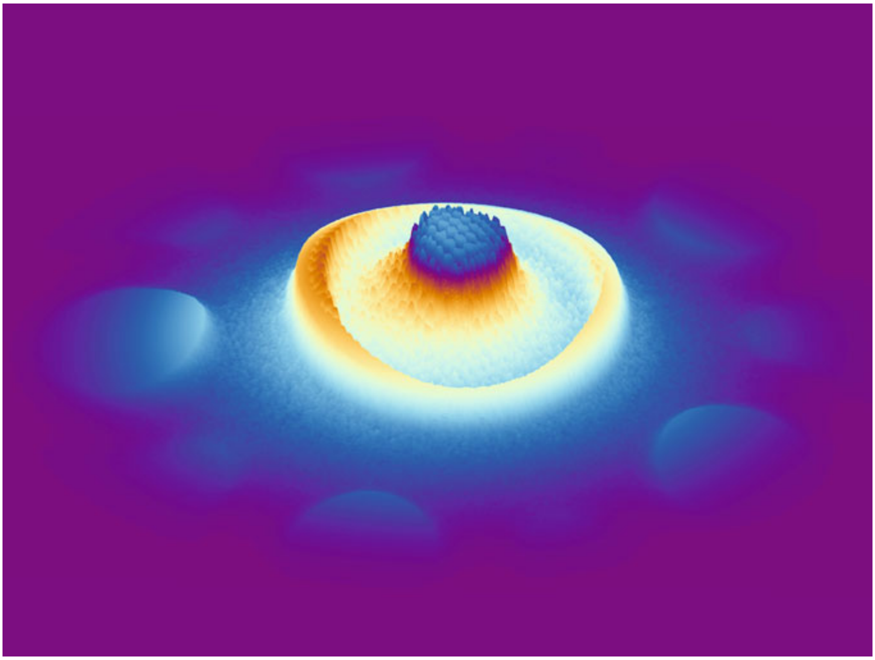
References

1. Duvauchelle, P., Freud, N., Kaftandjian, V., Babot, D.: A computer code to simulate X-ray imaging techniques. Nucl. Instrum. Methods Phys. Res. B **2000**(170), 245–258 (2000)
2. Huang, Q., Wu, Y., Baruch, J., Jiang, P., Peng, Y.: A template model for defect simulation for evaluating nondestructive testing in X-radiography. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **39**(2), 466–475 (2009)
3. Tabary, J., Hugonnard, P., Mathy, F.: SINDBAD: a realistic multi-purpose and scalable X-ray simulation tool for NDT applications. In: International Symposium on DIR and CT, Lyon, vol. 1, pp. 1–10 (2007)
4. Rebuffel, V., Tabary, J., Tartare, M., Brambilla, A., Verger, L.: SINDBAD: a simulation software tool for multi-energy X-ray imaging. In: Proceeding 11th European Conference on Non Destructive Testing, Prague (2014)
5. Salvat, F., Fernández-Varea, J.M., Sempau Roma, J.: PENELOPE-2008: a code system for Monte Carlo simulation of electron and photon transport. In: Workshop Proceedings, Barcelona, 30 June–3 July 2, 2008. OECD (2009)
6. Yao, M., Duvauchelle, P., Kaftandjian, V., Peterzol-Parmentier, A., Schumm, A.: X-ray imaging plate performance investigation based on a Monte Carlo simulation tool. Spectrochim. Acta Part B: At. Spectrosc. **103**, 84–91 (2015)
7. Schumm, A., Duvauchelle, P., Kaftandjian, V., Jaenisch, R., Bellon, C., Tabary, J., Mathy, F., Legoupil, S.: Modelling of radiographic inspections. In: Nondestructive Testing of Materials and Structures, pp. 697–702. Springer (2013)
8. Hecker, H.: A new method to process X-ray images in the automated inspection of castings. Ph.D. thesis, Institute for Measurement and Automation, Faculty of Electrical Engineering, Technical University of Berlin (1995) (in German)
9. Mery, D.: A new algorithm for flaw simulation in castings by superimposing projections of 3D models onto X-ray images. In: Proceedings of the XXI International Conference of the Chilean Computer Science Society (SCCC-2001), pp. 193–202. IEEE Computer Society Press, Punta Arenas (2001)
10. Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Lévy, B.: Polygon Mesh Processing. CRC Press, Florida (2010)
11. Carvajal, K., Chacón, M., Mery, D., Acuna, G.: Neural network method for failure detection with skewed class distribution. Insight **46**(7), 399–402 (2004)
12. Mery, D., Filbert, D.: Automated flaw detection in aluminum castings based on the tracking of potential defects in a radioscopic image sequence. IEEE Trans. Robot. Autom. **18**(6), 890–901 (2002)
13. Bavendiek, K.: Prüfkörper für die automatischen überprüfung der Bildqualität und der Messung der Erkennungssicherheit bei ADR Systemen. In: German Conference on Nondestructive Testing. Berlin (2001) (in German)
14. Mery, D.: Flaw simulation in castings inspection by radioscopy. Insight **43**(10), 664–668 (2001)
15. Heinrich, W.: Automated inspection of castings using X-ray testing. Ph.D. thesis, Institute for Measurement and Automation, Faculty of Electrical Engineering, Technical University of Berlin (1988) (in German)
16. Filbert, D., Klatte, R., Heinrich, W., Purschke, M.: Computer aided inspection of castings. In: IEEE-IAS Annual Meeting, pp. 1087–1095. Atlanta, (1987)
17. Tillack, G.R., Nockemann, C., Bellon, C.: X-ray modelling for industrial applications. NDT & E Int **33**(1), 481–488 (2000)
18. Freud, N., Duvauchelle, P., Babot, D.: Simulation of X-ray NDT imaging techniques. In: Proceedings of the 15th World Conference on Non-Destructive Testing (WCNDT–2000). Rome (2000)
19. Mery, D., Filbert, D.: The epipolar geometry in the radioscopy: theory and application. at -. Automatisierungstechnik **48**(12), 588–596 (2000) (in German)

20. Mery, D.: Automated Flaw Detection in Castings from Digital Radioscopic Image Sequences. Verlag Dr. Köster, Berlin (2001). Ph.D. thesis in German
21. Mery, D., Hahn, D., Hitschfeld, N.: Simulation of defects in aluminum castings using CAD models of flaws and real x-ray images. Insight **47**(10), 618–624 (2005)

Chapter 8
Applications in X-ray Testing

Abstract In this chapter, relevant applications on X-ray testing are described. We cover X-ray testing in (i) castings, (ii) welds, (iii) baggage, (iv) natural products, and (v) others (like cargos and electronic circuits). For each application, the state of the art is presented. Approaches in each application are summarized showing how they use computer vision techniques. A detailed approach is shown in each application and some examples using Matlab are given in order to illustrate the performance of the methods.



Cover image: *3D representation of the X-ray image of a wheel (X-ray image* `C0023_0001` *colored with 'sinmap' colormap).*

D. Mery, *Computer Vision for X-Ray Testing*, DOI 10.1007/978-3-319-20747-6_8

8.1 Introduction

In this chapter, we review some relevant applications in X-ray testing such as (i) castings, (ii) welds, (iii) baggage, (iv) natural products, and (v) others (like cargos and electronic circuits). For the first four application applications, in which the author has been undertaking research over the last decades, we will present a description, the state of the art, a detailed approach and an example in Matlab. For the last application, different techniques are mentioned.

8.2 Castings

Light-alloy castings produced for the automotive industry, such as wheel rims, steering knuckles, and steering gear boxes are considered important components for overall roadworthiness. Nonhomogeneous regions can be formed within the work piece in the production process. These are manifested, for example, by bubble-shaped voids, fractures, inclusions, or slag formation. To ensure the safety of construction, it is necessary to check every part thoroughly using X-ray testing. In casting inspection, automated X-ray systems have not only raised quality, through repeated objective inspections and improved processes, but have also increased productivity and consistency by reducing labor costs. Some examples are illustrated in Fig. 8.1.

8.2.1 State of the Art

Different methods for the automated detection of casting discontinuities using computer vision have been described in the literature over the 30 years. One can see that approaches to detecting can be divided into three groups: (i) approaches where an error-free reference image is used; (ii) approaches using pattern recognition, expert

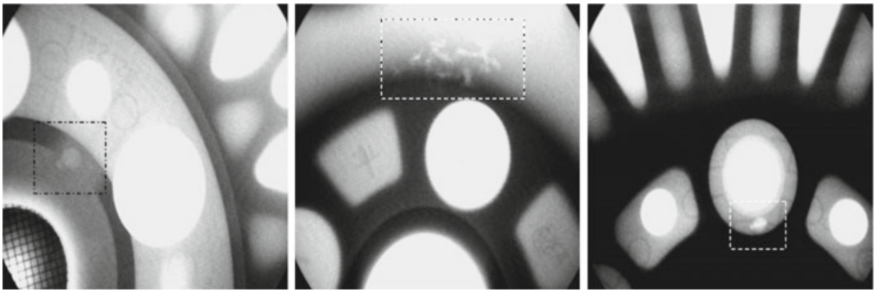


Fig. 8.1 Real defects in X-ray images of wheels

systems, artificial neural networks, general filters, or multiple view analyzes to make them independent of the position and structure of the test piece; and (iii) approaches using computer tomography to make a reconstruction of the cast piece and thereby detect discontinuities [1]. Selected approaches are summarized in Table 8.1. In this area, the automated systems are very effective, because the inspection task is fast and obtains a high performance.

8.2.2 An Application

In this section, we present a method for the automated detection of flaws based on *tracking principle* in an X-ray image sequence, i.e., first, it identifies potential defects in each image of the sequence, and second, it matches and tracks these from image to image. The key idea is to consider as false alarms those potential defects which cannot be tracked in the sequence [4]. The method for automated flaw detection presented here has basically two steps (see Fig. 8.2): *identification* and *tracking of potential flaws*. These will be described in this section.

Identification of Potential Flaws

A digital X-ray image sequence of the object test is acquired (see for example series `C0001` of $\mathbb{GDX}$ray). In order to ensure the tracking of flaws in the X-ray images, similar projections of the specimen must be achieved along the sequence. For this reason, the sequence consists of X-ray images taken by the rotation of the casting at small intervals (e.g., 5^0). Since many images are captured, the time of the data acquisition is reduced by taking the images without frame averaging. The position of the casting, provided online by the manipulator is registered at each X-ray image to calculate the perspective projection matrix $\mathbf{P}_p$ (for details see Sect. 3.3.4 and Example 3.5). An X-ray image sequence is shown in Figs. 8.3 and 8.4.

The detection of potential flaws identifies regions in X-ray images that may correspond to real defects. This process takes place in each X-ray image of the sequence without considering information about the correspondence between them. Two general characteristics of the defects are used for identification purposes: (i) a flaw can be considered as a connected subset of the image, and (ii) the gray-level difference between a flaw and its neighborhood is significant. However, as the signal-to-noise ratio in our X-ray images is low, the flaws signal is slightly greater than the background noise, as illustrated in Fig. 8.5. In our experiments, the mean gray level of the flaw signal (without background) was between 2.4 and 28.8 gray values with a standard deviation of 6.1. Analyzing a homogeneous background in different areas of interest of normal parts, we found that the noise signal was within ± 13 gray values with a standard deviation of 2.5. For this reason, the identification of real defects with poor contrast can also involve the detection of false alarms.

According to the mentioned characteristics of the real flaws, our method of identification has the following two steps (see Fig. 8.6):

Table 8.1 Applications on castings

References	Energy		Geometric model[a]	Single views[b]			Active vision	Multiple views[b]		
	Mono	Dual		①	②	③		①	②	③
Carrasco (2011) [2]	×		N	×	×	×	–	×	×	×
Li (2006) [3]	×		–	×	×	–	–	–	–	–
Mery (2002) [4]	×		C	×	×	×	–	×	×	×
Mery (2011) [5]	×		N	×	×	×	–	×	×	×
Pieringer (2010) [6]	×		C	×	×	×	–	×	×	×
Pizarro (2008) [7]	×		N	×	×	×	–	×	×	×
Ramirez (2013) [8]	×		N	×	×	×	–	–	–	–
Tang (2009) [9]	×		–	×	–	–	–	–	–	–
Zhao (2014) [10]	×		–	×	×	×	–	–	–	–
Zhao (2015) [11]	×		–	×	×	×	–	–	–	–

[a] *C* Calibrated, *N* Not calibrated, – Not used.

[b] See ①, ②, ③ in Fig. 1.21

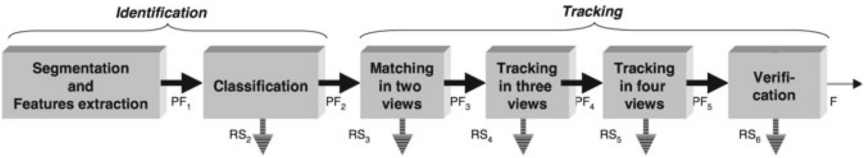


Fig. 8.2 Automated flaw detection in aluminum castings based on the tracking of potential defects in an X-ray image sequence: PF = potential flaws, RS = potential flaws classified as regular structures, F = detected flaws [4]

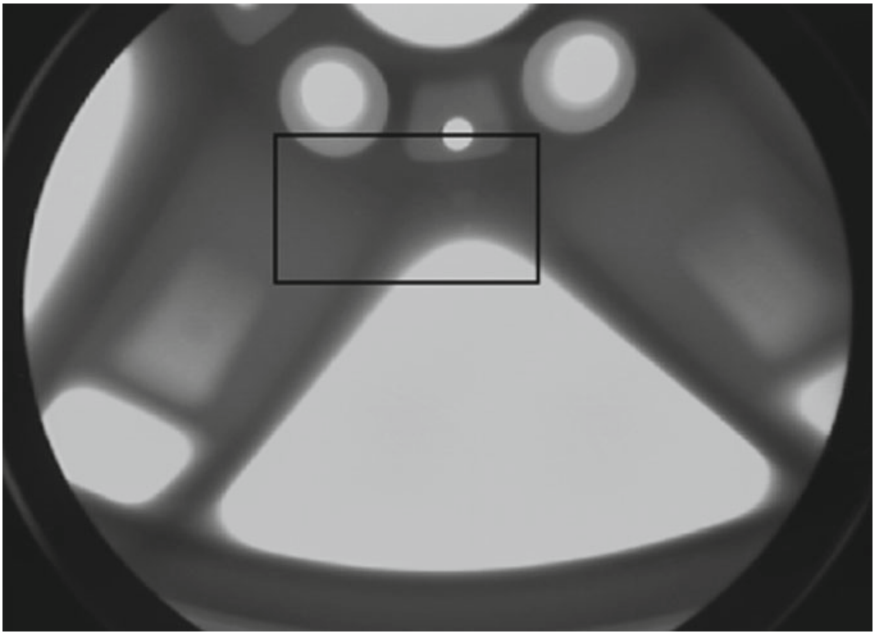


Fig. 8.3 X-ray image `C0001_0030` of an aluminum wheel (see zoom in Fig. 8.5)

Edge Detection: A Laplacian-of-Gaussian (LoG) kernel and a zero-crossing algorithm [12] are used to detect the edges of the X-ray images. The LoG-operator involves a Gaussian low-pass filter which is a good choice for the pre-smoothing of our noisy images. The resulting binary edge image should produce at real flaws closed and connected contours which demarcate *regions*. However, a flaw may not be perfectly enclosed if it is located at an edge of a regular structure as shown in Fig. 8.6c. In order to complete the remaining edges of these flaws, a thickening of the edges of the regular structure is performed as follows: (a) the gradient image[1] of the original image is computed (see Fig. 8.6d); (b) by thresholding the gradient

[1] The gradient image is computed by taking the square root of the sum of the squares of the gradient in a horizontal and vertical direction. These are calculated by the convolution of the X-ray image with the first derivative (in the corresponding direction) of the Gaussian low-pass filter used in the LoG-filter.

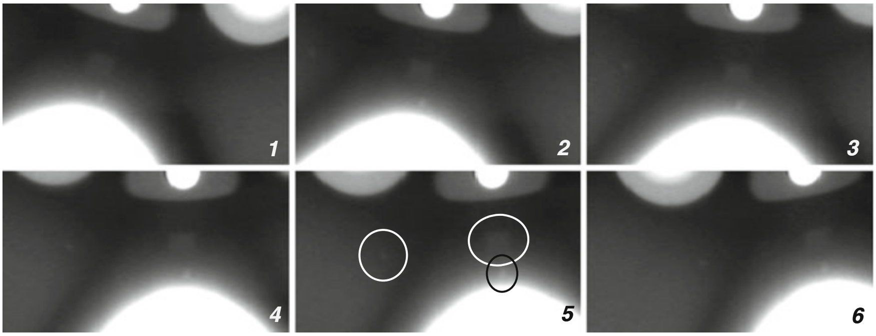


Fig. 8.4 X-ray image sequence with three flaws (image 5 is shown in Fig. 8.5)

Fig. 8.5 Zoom of Fig. 8.3 and gray-level profile along three rows crossing defects

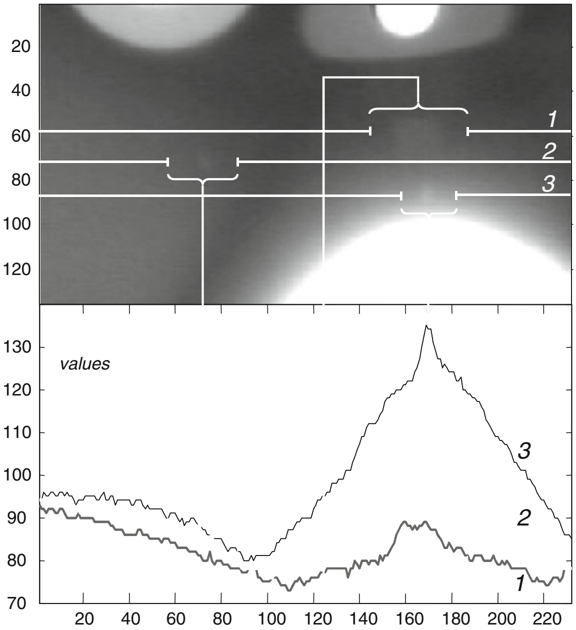


image at a high gray level a new binary image is obtained; and (c) the resulting image is added to the zero-crossing image (see Fig. 8.6e).

Segmentation and Classification of Potential Flaws: Afterwards, each closed region is segmented and classified as a potential flaw if (a) its mean gray level is 2.5 % greater than the mean gray level of its surroundings (to ensure the detection of the flaws with a poor contrast); and (b) its area is greater than 15 pixels (very small flaws are permitted). A statistical study of the classification of potential flaws using more than 70 features can be found in [13].

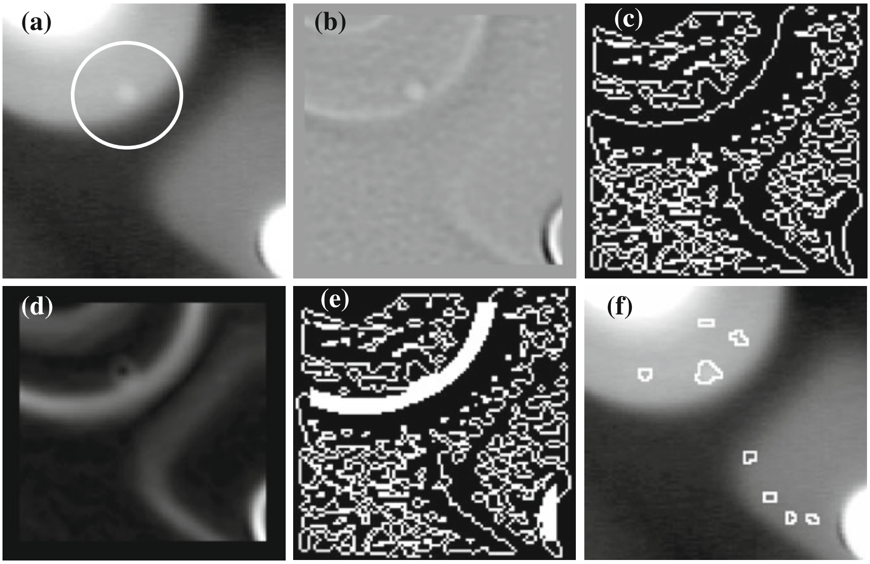


Fig. 8.6 Identification of potential flaws: **a** X-ray image with a small flaw at an edge of a regular structure, **b** Laplacian-filtered image with σ = 1.25 pixels (kernel size = 11 × 11), **c** zero-crossing image, **d** gradient image, **e** edge detection after adding high gradient pixels, and **f** potential flaws

This is a very simple detector of potential flaws (see implementation in Example 5.6). However, the advantages are as follows: (a) it is a single detector (it is the same detector for each image) and (b) it is able to identify potential defects independent of the placement and the structure of the specimen.

Using this method, some real defects cannot be identified in all X-ray images in which they appear if the contrast is very poor or the flaw is not enclosed by edges. For example, in Fig. 8.7 one can observe that the biggest real flaw was identified in images 1, 2, 3, 4, and 6, but not in image 5 where only two of the three real flaws were identified (compare with Fig. 8.5). Additionally, if a flaw is overlapped by edges of the structure of the casting, not all edges of the flaw can be detected. In this case, the flaw will not be enclosed and therefore not be segmented. Furthermore, a small flaw that moves in front (or behind) a thick cross-section of the casting, in which the X-rays are highly absorbed, may cause an occlusion. In our experiments, this detector identified the real flaws in four or more (not necessarily consecutive) images of the sequence.

Multiple View Detection

In the previous step, n_1 potential regions were segmented and described in the entire image sequence $\mathbb{I}$. Each segmented region is labeled with a unique number $r \in \mathbf{T}_1 = \{1, \ldots, n_1\}$. In view i, there are m_i segmented regions that are arranged in a subset $\mathbf{t}_i = \{r_{i,1}, r_{i,2}, \ldots, r_{i,m_i}\}$, i.e., $\mathbf{T}_1 = \mathbf{t}_1 \cup \mathbf{t}_2 \cup \cdots \mathbf{t}_m$.

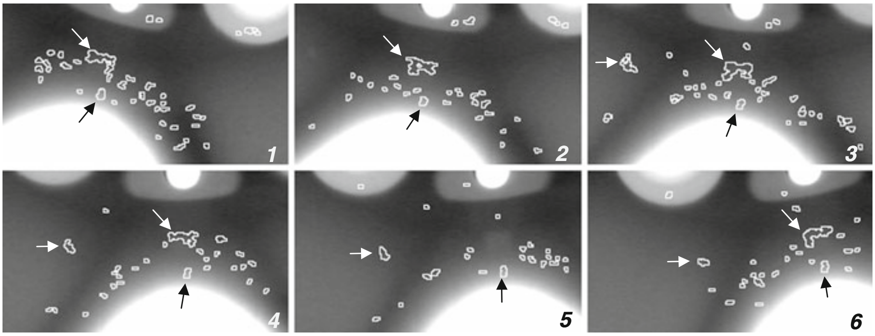


Fig. 8.7 Identification of potential flaws (the *arrows* indicate real flaws)

The matching and tracking algorithms combine all regions to generate consistent tracks of the object's parts of interest across the image sequence. The algorithm has the following steps:

Matching in Two Views: All regions in view i that have corresponding regions in the next p views are searched, i.e., regions $r_1 \in \mathbf{t}_i$ that have corresponding regions $r_2 \in \mathbf{t}_j$ for $i = 1, \ldots, m-1$ and $j = i+1, \ldots, \min(i+p, m)$. In our experiments, we use $p = 3$ to reduce the computational cost. The matched regions (r_1, r_2) are those that meet *similarity* and *location* constraints. The similarity constraint means that corresponding descriptors $\mathbf{y}_{r_1}$ and $\mathbf{y}_{r_2}$ must be similar enough such that

$$||\mathbf{y}_{r_1} - \mathbf{y}_{r_2}|| < \varepsilon_1. \tag{8.1}$$

The location constraint means that the corresponding locations of the regions must meet the epipolar constraint. In this case, the Sampson distance between $\mathbf{x}_{r_1}$ and $\mathbf{x}_{r_2}$ is used, i.e., the first-order geometric error of the epipolar constraint must be small enough such that

$$|\mathbf{x}_{r_2}^{\mathsf{T}} \mathbf{F}_{ij} \mathbf{x}_{r_1}| \left(\frac{1}{\sqrt{a_1^2 + a_2^2}} + \frac{1}{\sqrt{b_1^2 + b_2^2}} \right) < \varepsilon_2, \tag{8.2}$$

with $\mathbf{F}_{ij}\mathbf{x}_{r_1} = [a_1\ a_2\ a_3]^{\mathsf{T}}$ and $\mathbf{F}_{ij}^{\mathsf{T}}\mathbf{x}_{r_2} = [b_1\ b_2\ b_3]^{\mathsf{T}}$. In this case, $\mathbf{F}_{ij}$ is the fundamental matrix between views i and j calculated from projection matrices $\mathbf{P}_i$ and $\mathbf{P}_j$ [14] (see Sect. 3.5.1). In addition, the location constraint used is as follows:

$$||\mathbf{x}_{r_1} - \mathbf{x}_{r_2}|| < \rho(j - i), \tag{8.3}$$

because the translation of corresponding points in these sequences is smaller than ρ pixels in consecutive frames.

If we have 3D information about the space where our test object should be, it is worth to evaluating whether the 3D point reconstructed from the centers of mass of the regions must belong to the space occupied by the casting. From $\mathbf{m}_p^a$ and $\mathbf{m}_q^b$, the corresponding 3D point $\hat{\mathbf{M}}$ is estimated using the linear approach of Hartley in [14]. For two views, this approach is faster than the least squares technique. It is necessary to examine if $\hat{\mathbf{M}}$ resides in the volume of the casting, the dimensions of which are usually known a priori (e.g., a wheel is assumed to be a cylinder).[2]

Finally, a new matrix $\mathbf{T}_2$ sized $n_2 \times 2$ is obtained with all matched duplets (r_1, r_2), one per row. If a region is found to have no matches, it is eliminated. Multiple matching, i.e., a region that is matched with more than one region, is allowed. Using this method, problems like nonsegmented regions or occluded regions in the sequence can be solved by tracking if a region is not segmented in consecutive views.

Matching in 3 Views: Based on the matched regions stored in matrix $\mathbf{T}_2$, we look for triplets (r_1, r_2, r_3), with $r_1 \in \mathbf{t}_i, r_2 \in \mathbf{t}_j, r_3 \in \mathbf{t}_k$ for views i, j and k. We know that a row a in matrix $\mathbf{T}_2$ has a matched duplet $[T_2(a, 1)\ T_2(a, 2)] = [r_1\ r_2]$. We then look for rows b in $\mathbf{T}_2$ in which the first element is equal to r_2, i.e., $[T_2(b, 1)\ T_2(b, 2)] = [r_2\ r_3]$. Thus, a matched triplet (r_1, r_2, r_3) is found if the regions r_1, r_2 and r_3 meet the trifocal constrain:

$$||\hat{\mathbf{x}}_{r_3} - \mathbf{x}_{r_3}|| < \varepsilon_3, \tag{8.4}$$

This means that $\mathbf{x}_{r_3}$ must be similar enough to the reprojected point $\hat{\mathbf{x}}_{r_3}$ computed from the points in views i and j ($\mathbf{x}_{r_1}$ and $\mathbf{x}_{r_2}$), and the trifocal tensors T_i^{jk} of views i, j, k calculated from projection matrices $\mathbf{P}_i$, $\mathbf{P}_j$ and $\mathbf{P}_k$ [14] (see (3.76)). A new matrix $\mathbf{T}_3$ sized $n_3 \times 3$ is built with all matched triplets (r_1, r_2, r_3), one per row. Regions in which the three views do not match are eliminated.

The results of our example are shown in Fig. 8.8.

Matching in More Views: For $v = 4, \ldots, q \leq m$ views, we can build the matrix recursively $\mathbf{T}_v$, sized $n_v \times v$, with all possible v-tuplets $(r_1, r_2, \ldots, r_v)$ that fulfill $[T_{v-1}(a, 1)\ ...\ T_{v-1}(a, v-1)] = [r_1\ r_2\ \ldots\ r_{v-1}]$ and $[T_{v-1}(b, 1)\ \ldots\ T_{v-1}(b, v-1)] = [r_2\ \ldots\ r_{l-1}\ r_v]$, for $j, k = 1, \ldots, n_{v-1}$. No more geometric constraints are required because it is redundant. The final result is stored in matrix $\mathbf{T}_q$. For example, for $q = 4$ we store in matrix $\mathbf{T}_4$ the matched quadruplets (r_1, r_2, r_3, r_4) with $r_1 \in \mathbf{t}_i$, $r_2 \in \mathbf{t}_j, r_3 \in \mathbf{t}_k, r_4 \in \mathbf{t}_l$ for views i, j, k and l.

Figure 8.9 shows the tracked regions of our example that fulfill this criterion. Only two false trajectories are observed (see arrows).

[2] It is possible to use a CAD model of the casting to evaluate this criterion more precisely. Using this model, we could discriminate a small hole of the regular structure that is identified as a potential flaw. Additionally, the CAD model can be used to inspect the casting geometry, as shown in [15].

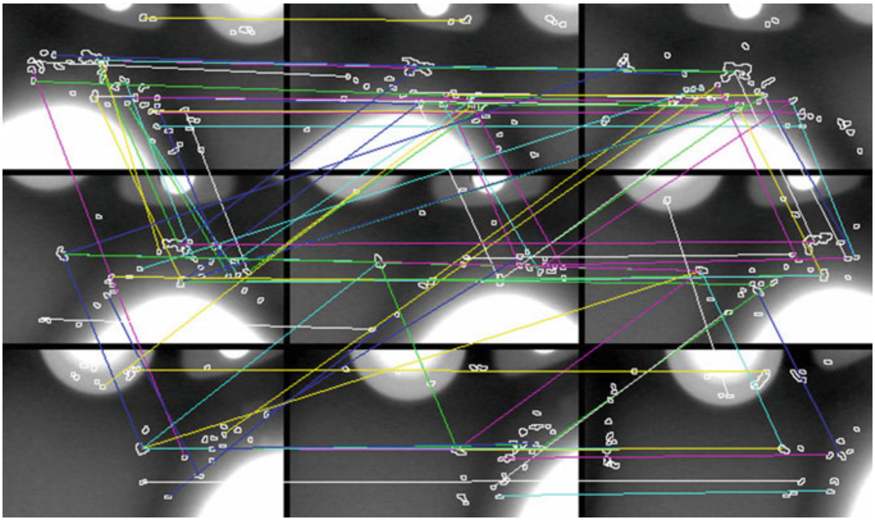


Fig. 8.8 Matching of potential flaws in two views

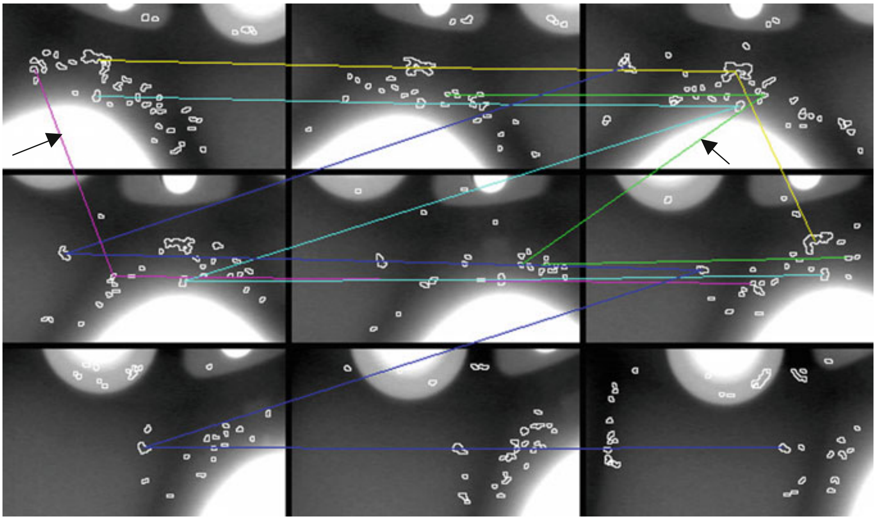


Fig. 8.9 Tracking in more views (the *arrows* indicate false detections)

As our detector cannot guarantee the identification of all real flaws in more than four views, a tracking in five views could lead to the elimination of those real flaws that were identified in only four views. However, if a potential flaw is identified in more than four views, more than one quadruplet can be detected. For this reason, these corresponding quadruplets are joined in a trajectory that contains more than four potential flaws (see trajectory with arrows in Fig. 8.9).

The matching condition for building matrix $\mathbf{T}_i$, $i = 3, \ldots, q$, is efficiently evaluated (avoiding an exhaustive search) using a k-d tree structure [16] to search the nearest neighbors for zero Euclidean distance between the first and last $i-2$ columns in $\mathbf{T}_{i-1}$.

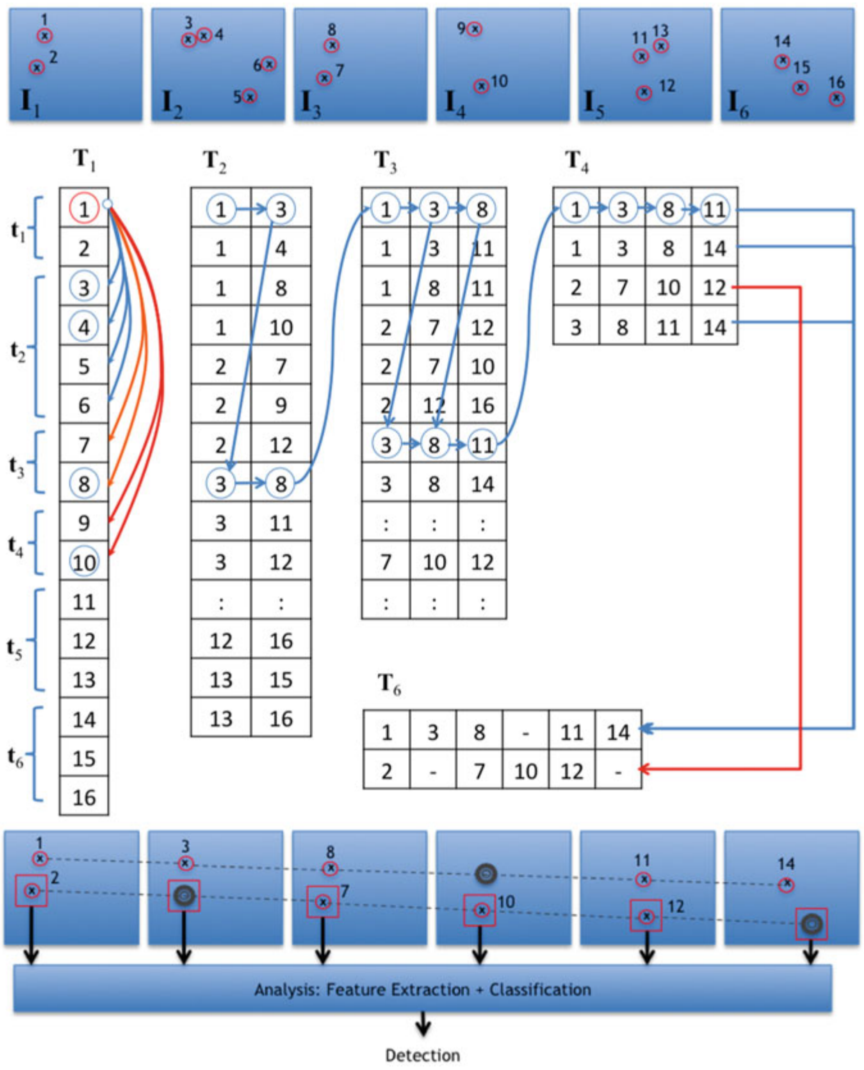
Merging Tracks: Matrix $\mathbf{T}_q$ defines tracks of regions in q views. It can be observed that some of these tracks correspond to the same region. For this reason, it is possible to merge tracks that have $q - 1$ common elements. In addition, if a new track has more than one region per view, we can select the region that shows the minimal reprojection error after computing the corresponding 3D location. In this case, a 3D reconstruction of $\hat{\mathbf{X}}$ is estimated from tracked points [14]. Finally, matrix $\mathbf{T}_m$ is obtained with all merged tracks in the m views. See an example of the whole tracking algorithm in Fig. 8.10.

Analysis: The 3D reconstructed point $\hat{\mathbf{X}}$ from each set of tracked points of $\mathbf{T}_m$ can be reprojected in views where the segmentation may have failed to obtain the complete track in all views. The reprojected points of $\hat{\mathbf{X}}$ should correspond to the centroids of the nonsegmented regions. It is then possible to calculate the size of the projected region as an average of the sizes of the identified regions in the track. In each view, a small window centered in the computed centroids is defined. These corresponding small windows, referred to as *tracked part*, will be denoted as $\mathbb{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_m\}$. In each view, a small window is defined with the estimated size in the computed centers of gravities (see Fig. 8.11). Afterwards, the corresponding windows are averaged. Thus, the attempt is made to increase the signal-to-noise ratio by the factor $\sqrt{n}$, where n is the number of averaged windows. As flaws must appear as contrasted zones relating to their environment, we can verify if the contrast of each averaged window is greater than 2.5 %. With this verification, it is possible to eliminate all remaining false detections. Figure 8.11 shows the detection in our sequence using this method. Our objective is then achieved: the real defects were separated from the false ones.

Experimental Results

In this section, results of automatic inspection of cast aluminum wheels using the outlined approach are presented. These results have been achieved recently on synthetic flaws and real data. The parameters of our method have been manually tuned, giving $\sigma = 1.25$ pixels (for LoG-operator), $\varepsilon_2 = 0.75$ mm, $\varepsilon_s = 0.7$, and $\varepsilon_3 = 0.9$ mm. These parameters were not changed during these experiments. A wheel was considered to be a cylinder with the following dimensions: 470 mm diameter and 200 mm height. The focal length (distance between X-ray source and entrance screen of the image intensifier) was 884 mm. The bottom of a wheel was 510 mm from the X-ray source. Thus, a pattern of 1 mm in the middle of the wheel is projected in the X-ray projection coordinate system as a pattern of 1.73 mm, and in the image coordinate system as a pattern of 2.96 pixels. The sequences of X-ray images were taken by rotation of the casting at 5^0.

The detection performance will be evaluated by computing the number of true positives (TP) and false positives (FP). They are, respectively, defined as the number of flaws that are correctly classified and the number of misclassified regular



$\mathbf{T}_1$

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

$\mathbf{T}_2$

1	3
1	4
1	8
1	10
2	7
2	9
2	12
3	8
3	11
3	12
:	:
12	16
13	15
13	16

$\mathbf{T}_3$

1	3	8
1	3	11
1	8	11
2	7	12
2	7	10
2	12	16
3	8	11
3	8	14
:	:	:
7	10	12
:	:	:

$\mathbf{T}_4$

1	3	8	11
1	3	8	14
2	7	10	12
3	8	11	14

$\mathbf{T}_6$

1	3	8	-	11	14
2	-	7	10	12	-

Fig. 8.10 Tracking example with $m = 6$ views. In each view there are 2, 4, 2, 2, 3, and 3 segmented regions, i.e., there are $n_1 = 16$ regions in total. For each region, we seek corresponding regions in the next $p = 3$ views (see *matching arrows* in $\mathbf{T}_1$: region 1 with regions (3, 4, 5, 6) in view 2, regions (7, 8) in view 3, and (9, 10) in view 4). We observe that after tracking in 2, 3, and 4 views there are only two tracks in $\mathbf{T}_6$ that could be tracked in 5 and 4 views, respectively. The regions that were not segmented can be recovered by reprojection (see *gray circles* in views 2, 4 and 6). Finally, each set of tracked regions are analyzed in order to take the final decision (Color figure online)

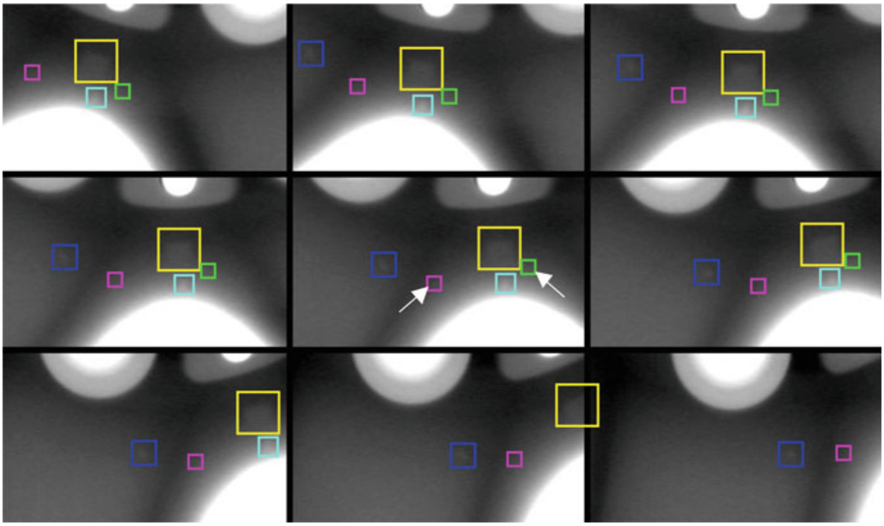


Fig. 8.11 Reconstruction and verification: the false detections (indicated by the *arrows*) are eliminated after the verification in all images of the sequence

structures. The TP and FP will be normalized by the number of existing flaws (E) and the number of identified potential flaws (I). Thus, we define the following percentages: TPP = TP / E ×100 and FPP = FP / I ×100. Ideally, TPP = 100 % and FPP = 0 %.

Synthetic Flaws: To evaluate the performance of our method in critical cases, real data in which synthetic flaws have been added were examined (see Sect. 7.4.3). A simple 3D modeled flaw (a spherical bubble) was projected and superimposed on real X-ray images of an aluminum wheel according to the law of X-ray absorption [17]. In our experiment, a flaw is simulated in 10 X-ray images of a real casting, in an area that included an edge of the structure (see Fig. 8.12a). In this area, the synthetic flaw was located in 24 different positions in a regular grid manner. At each position, TPP and FPP were tabulated. This test was repeated for different sizes of the flaws ($\emptyset = 1.5 \sim 7.5$ mm) which are illustrated in Fig. 8.12b. The results are shown in Fig. 8.12c. It was observed that the FPP was always zero. The TPP was 100 % for $\emptyset \geq 2.5$ mm, and greater than 95 % for $\emptyset \geq 2.1$ mm. However, the identification of the flaw may fail (and therefore also its detection) if it is very small and is located at the edge of the structure of the casting. In this case, one may choose a smaller value of the parameter σ in the LoG-operator of the edge detection, which will unfortunately increment the FPP. Other noncritical experiments, where the area of the simulation does not include an edge of the structure, have led to perfect results (TPP = 100 %, FPP = 0 %) for $\emptyset \geq 1.5$ mm ($\geq$4.4 pixels). Usually, the minimum detectable defect size according to inspection specifications is in the order of $\emptyset = 2$ mm. In X-ray testing, smaller flaws can be detected by decreasing the distance of the object test to the X-ray source.

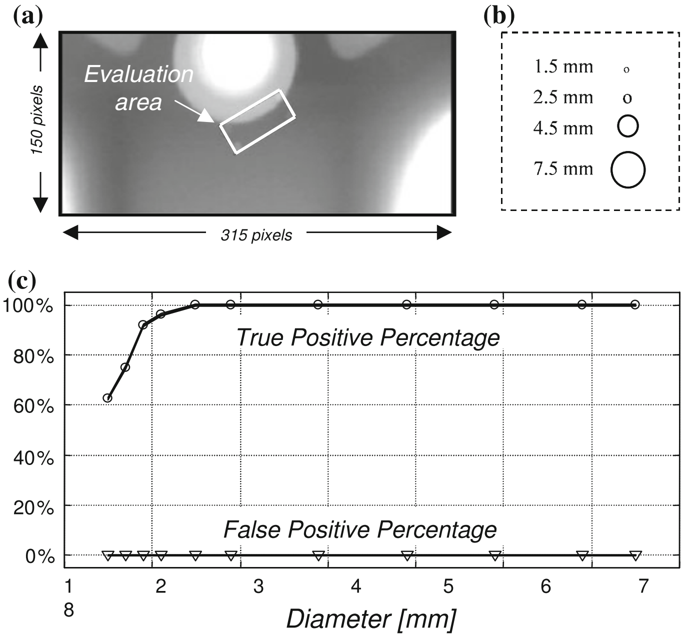


Fig. 8.12 Detection on synthetic flaws: **a** X-ray image and evaluated area, **b** flaw sizes, and **c** TPP and FPP

Real Data: Fourteen X-ray image sequences of aluminum wheels with twelve known flaws were inspected. Three of these defects were existing blow holes (with Ø = 2.0 ∼ 7.5 mm). They were initially detected by a visual (human) inspection. The remaining nine flaws were produced by drilling small holes (Ø = 2.0 ∼ 4.0 mm) in positions of the casting which were known to be difficult to detect. Casting flaws are present only in the first seven sequences. The results are summarized in Table 8.2, Figs. 8.13 and 8.14. In the identification of potential flaws, it was observed that the FPP was 98 % (4,310/4,381). Nevertheless, the TPP in this experiment was good, and it was possible to identify 85 % (71/84) of all projected flaws in the sequences (13 of the existing 84 flaws were not identified because the contrast was poor or they were located at edges of regular structures). It was observed that in the next steps, the FPP was reduced to nil. The detection of the real flaws was successful in all cases. The first six images of sequence 3 and its results were already illustrated in Figs. 8.4, 8.7, 8.8, 8.9, 8.10 and 8.11. The results on the other sequences with flaws are shown in Fig. 8.13.

Comparison with Other Methods: In this section, we present a comparison of our proposed algorithm with other methods that can be used to detect defects in aluminum castings. In this comparison, we evaluate the same real 14 sequences used in the previous section. The results are summarized in Table 8.3.

Table 8.2 Detection of flaws on real data

Seq.	X-ray images	Flaws in the sequence	Flaws in the images (E)	Identification			Detection	
				TP	FP	Total (I)	TP	FP
1	10	2	12	12	249	261	2	0
2	9	1	9	8	238	246	1	0
3	9	3	23	19	253	272	3	0
4	8	1	8	4	413	417	1	0
5	6	1	6	6	554	560	1	0
6	8	1	8	8	196	204	1	0
7	6	3	18	14	445	459	3	0
8	6	0	0	0	178	178	0	0
9	9	0	0	0	256	256	0	0
10	8	0	0	0	150	150	0	0
11	8	0	0	0	345	345	0	0
12	6	0	0	0	355	355	0	0
13	6	0	0	0	365	365	0	0
14	9	0	0	0	313	313	0	0
Total	108	12	84	71	4,310	4,381	12	0
Percentage				85 %	98 %		100 %	0 %

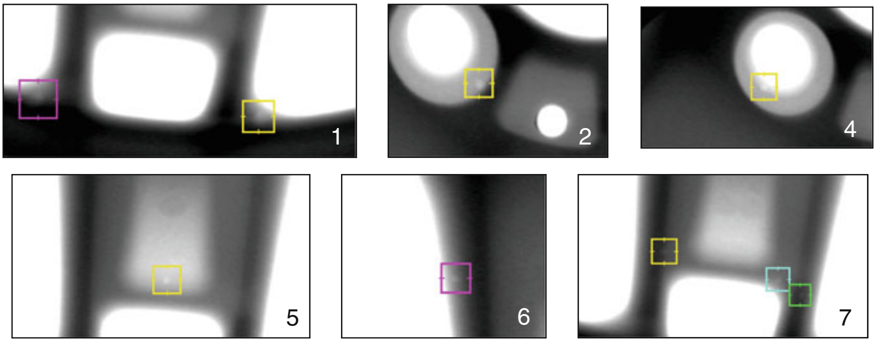


Fig. 8.13 Detected flaws in sequences 1, 2, 4, 5, 6, and 7 (sequence 3 is shown in Fig. 8.11)

First, we compared the first step of our method (*identification of potential flaws*). The objective of this step is the use of a single filter, instead of a set of filters adapted to the regular structure of the specimen. We evaluated the well-known Canny filter (see for example [12]). As this filter detects sparse edge pixels that not necessarily produce at real flaws closed and connected contours, the TPP of this detector was unacceptable, only 4% of the real flaws were identified (‘Canny I’ in Table 8.3). In order to increase the number of closed regions, a dilation of the edges using a 3 × 3 mask was performed. Although the TPP is improved to 40 % (‘Canny II’

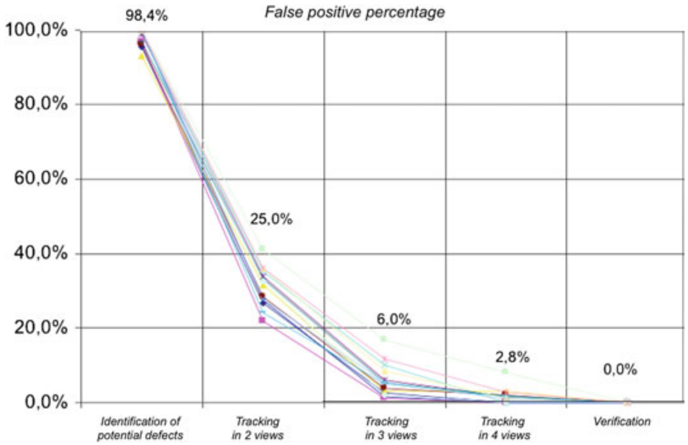


Fig. 8.14 False positive percentage on real data in the 14 real sequences (the number of identified potential flaws corresponds to 100 %). The mean of each step is given over the fourteen curves

Table 8.3 Comparison with other methods

Method	Identification		Detection	
	TPP (%)	FPP (%)	TPP (%)	FPP (%)
Proposed	85	98	100	0
Canny I	4	97	0	–
Canny II	40	99	17	40
Median I	55	85	33	36
Median II	88	98	92	45
Tracking in 3	85	98	100	25
Tracking in 5	85	98	83	0
PXV-5000	–	–	100	0

in Table 8.3), many flaws were not detected in any of the images of the sequence. For this reason, only 17 % of the real flaws were detected after the tracking and verification.

Another detection of potential flaws can be performed using a region-based segmentation. Median filtering is normally used to generate an error-free image, since defect structures are essentially eliminated, while design features of the test piece are normally preserved [18]. Once the error-free reference image is computed, an error difference image between original and error-free images is calculated. Casting defects are then identified when a sufficiently large gray level in the error difference image occurs. The best results were obtained using a median filter with a 11×11 mask. We evaluated two thresholds: $\theta = 6$ and $\theta = 2$—by 256 gray levels—(see 'Median I' and 'Median II' in Table 8.3). In the first case, the TPP was only 55 %. By decreasing the threshold value, we increased the TPP to 88 %, that is slightly better than our detector (85 %). However, systematic false alarms were detected at the corners of the regular structures. Since these false alarms satisfy the multifocal conditions, they can be tracked in the sequence. For this reason, this detector can only be used if the median filter is adapted to the regular structures of the specimen using a priori information. Normally, a set of median filters is used for each X-ray image [19–21].

In order to evaluate the second step of our method (*tracking of potential flaws*), we tested the method by tracking the potential flaws in 3 and in 5 views, instead of 4 views (see 'Tracking in 3', 'Tracking in 5' and 'Proposed' in Table 8.3). By considering only three views, we obtained so many false alarms that the verification step detected four false alarms (25 %). In the other case, by tracking the potential flaws in five views, real flaws that were segmented in only four views of the sequences were not tracked. For this reason, only 83 % of the real flaws were detected.

Finally, we inspected the test castings using a classic image-processing method. In our experiments, we used the industrial software PXV-5000 [22]. The results were excellent: 100 % of the real flaws were detected without false alarms. As a result of its peak detection performance, the classic image-processing methods have become the most widely established in industrial applications. However, these method suffer from the complicated configuration of the filtering, which is tailored to the test piece. In our experiments, the configuration process has taken two weeks. Nevertheless, as our method requires only a few number of parameters, the configuration could be carried out in hours.

Conclusions

A new method for automated flaw detection in aluminum castings using multiple view geometry has been developed. Our method is very efficient because it is based on a two-step analysis: identification and tracking. The idea was to try to imitate the way a human inspector inspects X-ray images: first, relevant details (potential defects) are detected, followed by tracking them in the X-ray image sequence. In this way, the false detections can be eliminated without discriminating the real flaws.

The great advantage of our first step is the use of a single filter to identify potential defects, which is independent of the structure of the specimen. Nevertheless, its disadvantages are as follows: (a) the false positive percentage is enormous; (b) the true positive percentage could be poor if the flaws to be detected are very small and located at the edge of a structure; and (c) the identification of regions is time consuming. Contrarily, the second step is highly efficient in both discrimination of false detections and tracking of real defects, and is not time consuming, due to the use of the multiple view tensors.

To inspect a whole wheel, our method requires approximately 100 views of 256 × 256 pixels, that can be processed in one minute. The required computing time is acceptable for practical applications because a typical inspection process takes about 1 min, independently of whether it is performed manually or automatically.

We have shown that these preliminary results are promising. However, given that the performance of the method has been verified on only a few X-ray image sequences, an evaluation on a broader data base is necessary.

It is possible to combine our second step with existing defect detection technologies, which use a priori information of the regular structures of the casting to detect flaws in single images (see for example [22]). This method could also be used in the automated flaw detection of other objects. In the adaptation of our method, one must determine the number of views in which a flaw must be tracked. If the false positive percentage by identifying potential flaws is low (or high), one may track a flaw in fewer (or more) views of the sequence. However, one must guarantee that the real flaws will be identified as potential flaws in these views.

8.2.3 An Example

In this section, an implementation that can be used for defect detection of castings in single views is presented. It consists of features that are extracted from positive class (the defects) and negative class (the background).

An example of using detection in multiple views can be found in Sect. 8.4.3.

Matlab Example 8.1 In this example, we show how to implement a classifier that is able to defect casting defects in single X-ray images. For this end, we use series C0002 that contains small images with and without defects. In addition, for this series we have the ground truth for all defects. The strategy of this example is to extract LBP features of all annotated defects, and LBP features of random patches that do not contain defects. After that we test a KNN classifier using cross-validation.

Listing 8.1 : Defect detection in castings

```
% CastingTraining.m
sdir              = Xgdxdir('C',2);                      % directory of the images
sfmt              = 'png';                               % format of the images
GT                = 'ground_truth.txt';                  % ground truth file
opx.opf.b         = Xfxbuild({'lbpri'});                 % LBP rotation invariant
opx.m             = 32;                                  % size of patches 0: 32x32
opx.n0            = 15;                                  % number of patche 0 per image
opx.th0           = 0.02;                                % threshold for patch 0
opx.segmentation  = 'Xsegbimodal';                       % wheel segmentation
opx.resize        = [32 32];                             % resize of patches 1: 32x32
[X0,d0,Xn]        = Xfxseqpatches(sdir,sfmt,GT,opx,0);    % Extracting patches 0
[X1,d1,Xn]        = Xfxseqpatches(sdir,sfmt,GT,opx,1);    % Extracting patches 1

X                 = [X0;X1];                             % features if both classes
d                 = [d0;d1];                             % labels of both classes

c.name            = 'knn';   c.options.k = 5;            % KNN with 5 neighbors
op.strat=1; op.c = c; op.v = 10; op.show = 1; op.p = 0.95;% 10 fold–cross–validation
[acc,ci]          = Xcrossval(X,d,op);
```

The output of this code is the estimated accuracy

```
» knn, 5 97.99% in (97.15, 98.82%) with CI=95%
```

That means with 95 % of confidence, the accuracy of this classifier is between 97.15 and 98.82 %. In this code, we used Xfxseqpatches (see Appendix B) of 𝕏vis Toolbox, that is able to extract features of class ‘1’ and class ‘0’. The reader can use additional series of 𝔾𝔻𝕏ray, that contain annotated defects in aluminum wheels. □

8.3 Welds

In welding process, a mandatory inspection using X-ray testing is required in order to detect defects like porosity, inclusion, lack of fusion, lack of penetration, and cracks. Industrial X-ray images of welds is widely used for detecting those defects in the petroleum, chemical, nuclear, naval, aeronautics, and civil construction industries, among others. An example is illustrated in Fig. 8.19.

8.3.1 State of the Art

Over the last three decades, substantial research has been performed on automated detection and classification of welding defects in continuous welds using X-ray imaging [23, 24]. Typically, the approaches follow a classical computer vision schema: (i) image acquisition—an X-ray digital image is taken and stored in the computer, (ii) preprocessing—the digital image is improved in order to enhance the details, (iii) segmentation—potential welding defects are found and isolated, (iv) feature extraction/selection—significant features of the potential welding defects and their surroundings are quantified, and (v) classification—the extracted features are interpreted automatically using a priori knowledge of the welding defects in order to separate potential defects into detected welding defects or false alarms. In the last few years, some methods based on the tracking principle (as explained in Sect. 8.2.2) have been developed [25, 26].

Selected approaches are summarized in Table 8.4. As we can see there is much research on weld inspection. Achieved performance of the developed algorithms is still not high enough, thus it is not suitable for fully automated inspection.

8.3.2 An Application

In computer vision, many object detection and classification problems have been solved without classic segmentation using *sliding-windows*. Sliding-window approaches have established themselves as state of the art in computer vision problems where an object must be separated from the background (see for example successful applications in face detection [44] and human detection [45]). In sliding-window methodology, a detection window (see black square in Fig. 8.15) is sledded over an input image in both horizontal and vertical directions, and for each localization of the detection window a classifier decides to which class the corresponding portion of the image belongs to according to its features. In this section, an approach to detect defects based on sliding-windows in welds is presented [36].

Table 8.4 Applications on welds

References	Energy		Geometric model[a]	Single views[b]			Active vision	Multiple views[b]		
	Mono	Dual		①	②	③		①	②	③
Anand (2009) [27]	×		–	×	×	×	–	–	–	–
Baniukiewicz (2014) [28]	×		–	×	×	×	–	–	–	–
Gao (2014) [29]	×		–	×	×	×	–	–	–	–
Kaftandjian (2003) [30]	×		–	×	×	×	–	–	–	–
Kumar (2014) [31]	×		–	×	×	×	–	–	–	–
Kumar (2014) [32]	×		–	×	×	×	–	–	–	–
Liao (2008) [33]	×		–	×	×	×	–	–	–	–
Liao (2009) [34]	×		–	×	×	×	–	–	–	–
Lindgren (2014) [25]	×		C	×	×	×	–	×	×	×
Mery (2003) [35]	×		–	×	×	×	–	–	–	–

(continued)

Table 8.4 (continued)

References	Energy		Geometric model[a]	Single views[b]			Active vision	Multiple views[b]		
	Mono	Dual		①	②	③		①	②	③
Mery (2011) [36]	×		–	×	×	×	–	–	–	–
Mu (2011) [37]	×		–	×	×	×	–	–	–	–
Shao (2014) [26]	×		–	×	×	×	–	×	×	×
Shi (2007) [38]	×		–	×	×	×	–	–	–	–
da Silva (2009) [39]	×		–	×	×	×	–	–	–	–
Vilar (2009) [40]	×		–	×	×	×	–	–	–	–
Wang (2008) [41]	×		–	×	×	×	–	–	–	–
Yiron (2015) [42]	×		–	×	×	×	–	×	×	×
Zapata (2008) [43]	×		–	×	×	×	–	–	–	–

[a] *C* Calibrated, *N* Not calibrated, – Not used.

[b] See ①, ②, ③ in Fig. 1.21

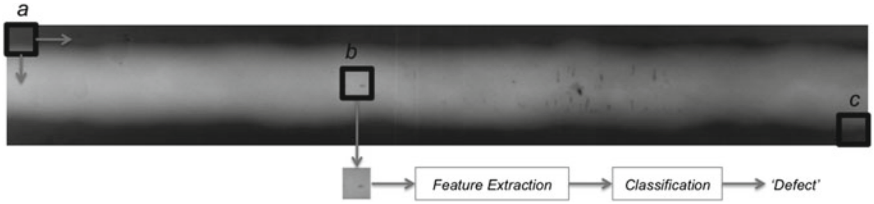


Fig. 8.15 Sliding-window approach: a detection window (see *black square*) is sledded over the X-ray image starting at place 'a' and ending at 'c'. For each position, e.g., at 'b', features are extracted only from the subimage defined by the *square*, and a classifier determines the class of this portion of the image

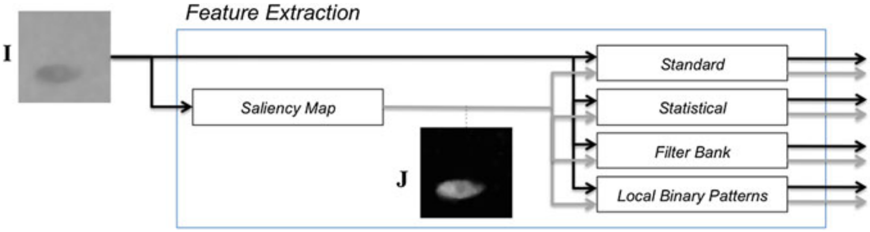


Fig. 8.16 Feature extraction: from each detection window several features are extracted (see *black path*). Additionally, the same features are extracted from a saliency map of the subwindow (see *gray path*)

Overview

We developed an X-ray computer vision approach to detect welding defects using this methodology yielding promising results. We will differentiate between the 'detection of defects' and the 'classification of defects' [46]. In the detection problem, the classes that exist are only two: 'defects' and 'no-efects', whereas the recognition of the type of the defects (e.g., porosity, slag, crack, lack of penetration, etc.) is known as classification of flaw types. This section describes our approach on detection only and the corresponding validation experiments. The classification of defects can be developed by the reader using a similar methodology.

The key idea of this example is to use a computer vision methodology, as shown in Figs. 8.15 and 8.16, to automatically detect welding defects. In the following, feature extraction, feature selection, classification, and validation will be explained in further detail.

Feature Extraction, Selection, and Classification

Features provide information about the intensity of a subimage. In our approach, p features per *intensity* channel were extracted. The used intensity channels in our work are only two: the grayscale X-ray image (**I**) and a saliency map (**J**) computed from **I**, i.e., , $p \times 2$ features for two intensity channels. In order to reduce the computational time, we restricted the feature extraction for these only two channels, however, other channels, like Harris transform [47] or other saliency maps, can be used.

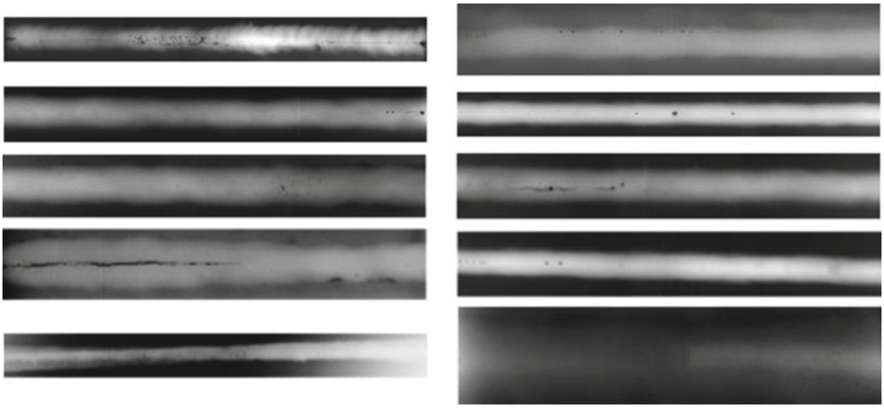


Fig. 8.17 X-ray images used in our experiments (series W0001 of 𝔾𝔻𝕏ray)

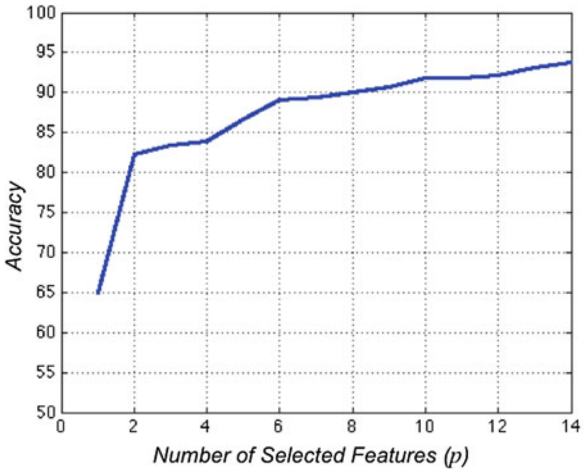
The saliency map **J** is obtained using a center-surround saliency mechanism based on a biologically inspired attention system [48].[3] In order to achieve faster processing, this theory proposes that the human visual system uses only a portion of the image, called *focus of attention*, to deal with complex scenes. In our approach, we use the *off-center* saliency map that measures the different dark areas surrounded by a bright background, as shown in Fig. 8.16.

In a training phase, using a priori knowledge of the welding defects, the detection windows are manually labeled as one of two classes: 'defects' and 'no-defect'. The first class corresponds to those regions where the potential welding defects are indeed welding defects. Alternatively, the second class corresponds to false alarms. For this end, we use series W0001 and W0002 of 𝔾𝔻𝕏ray. In the first series, we have the X-ray images, whereas in the second one we have the corresponding binary images representing the ground truth. Thus, the ideal segmentation of image `W0001_00i.png` is binary image `W0002_00i.png`, for `i` = `01 ... 10`.. Intensity features of each channel are extracted for both classes. Features extracted from each area of an X-ray image region are divided into four groups: basic intensity features (see Sect. 5.3.1), statistical features (see Sect. 5.3.5), Fourier and DCT features (see Sect. 5.3.7), Gabor features (see Sect. 5.3.6) and Local Binary Patterns (see Sect. 5.4.1). Afterwards, the extracted features are selected using feature selection approaches (see Sect. 5.6, and several classifiers (see Sect. 6.2) were evaluated using cross-validation (see Sect. 6.3.2).

Experiments

We experimented with 10 representative X-ray images (see Fig. 8.17). The average size of the image was 1.35 mega-pixels. For each X-ray image, 250 detection windows with detects and 250 without defects were selected, yielding $2 \times 250 \times 10 = 5{,}000$ detection windows. Each detection window was labeled with '1' for class *defects* and '0' for *no-defects*. The size of the detection windows were 24×24 pixels.

[3] The saliency function is implemented in Xsaliency (see Appendix B) of 𝕏VIS Toolbox.



m	Name	Channel
1	LBP(36)	Saliency
2	LBP(1)	Saliency
3	LBP(13)	Saliency
4	Int-Hu-Moment(1)	Saliency
5	LBP(32)	Saliency
6	Tx12,d1(range)	Saliency
7	Tx14,d2(range)	Saliency
8	LBP(58)	Saliency
9	LBP(54)	Saliency
10	LBP(42)	Gray
11	Tx 1,d1(mean)	Saliency
12	LBP(14)	Gray
13	Fourier(1,1)	Gray
14	LBP(4)	Gray

Selected Features

Fig. 8.18 Classification performance using the first p features

For each detection window 586 features were extracted. This means that 586 features were extracted from 5,000 samples (2,500 with defects and 2,500 without defects).

After the feature extraction, 75 % of the samples from each class were randomly chosen to perform the feature selection. The best performance was achieved using sequential forward selection. The best 14 features are shown in Fig. 8.18 in ascending order.

The performance of the classification using the SVM classifier and the first p selected features was validated using an average of ten cross-validation with 10-folds. The results are shown in Fig. 8.18. We observe that using 14 features, the performance was almost 94 % with a 95 % confidence interval between 93.0 and 94.5 %.

In order to test this methodology on X-ray images, the technique was implemented using a sliding-window sized 24 × 24 pixels that was shifted by 4 pixels. Thus, in each position, a subwindow of 24 × 24 pixels was defined and the corresponding features were extracted. The subwindow was marked if the trained classifier detected it as a discontinuity. Using a size of 24 × 24 pixel and a shift of 4 pixels, an image pixel could be marked from 0 to 36 times. Finally, if a pixel of the image was marked more than 24 times, then the pixel was considered as a discontinuity. The aforementioned parameters were set using an exhaustive search. The described steps are shown in Fig. 8.19 for one X-ray image. The results on other X-ray images are shown in Fig. 8.20. From these, one can see the effectiveness of the proposed technique.

Conclusions

In this section, we presented a new approach to detecting weld defects without segmentation based on sliding-windows and novel features. The promising results outlined in our work show that we achieved a very high classification rate in the

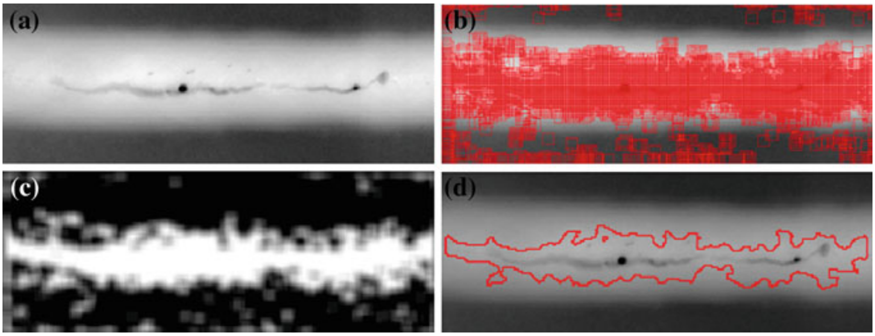


Fig. 8.19 Weld inspection using a sliding-window: **a** X-ray image, **b** detected windows, **c** activation map, and **d** detection [36]

detection of welding defects using a large number of features combined with efficient feature selection and classification algorithms. The key idea of the proposed method was to select, from a large universe of features, namely 572 features, only those features that were relevant for the separation of the two classes. We tested our method on 10 representative X-ray images yielding a performance of 94 % in accuracy using only 14 features and support vector machines. It is important to note that local binary pattern features extracted from the saliency map play an important role in the performance of the classifier. The method was implemented and tested on real X-ray images showing high effectiveness.

8.3.3 An Example

In this section, we present a Matlab code that can be used to detect defects in welds according to sliding-windows approach explained above.

Matlab Example 8.2 In this example, we show how to implement—for a simple perspective—the strategy explained in the previous section. We will use one part of image `W0001_0001.png` as training, and another part as testing. We will extract only a few number of features, and we will test only one feature selection technique and only one classifier. The reader can modify this code in order to achieve better results. The reader will note that this example has pedagogical purposes only. In order to develop a real application, more training images must be taken into account.

Listing 8.2 : Sliding-Windows

```
% SlidingWindows.m

% X-ray image and Ground Truth for training and testing
I             = Xloadimg('W',1,1,1);
J             = Xloadimg('W',2,1,1);
```

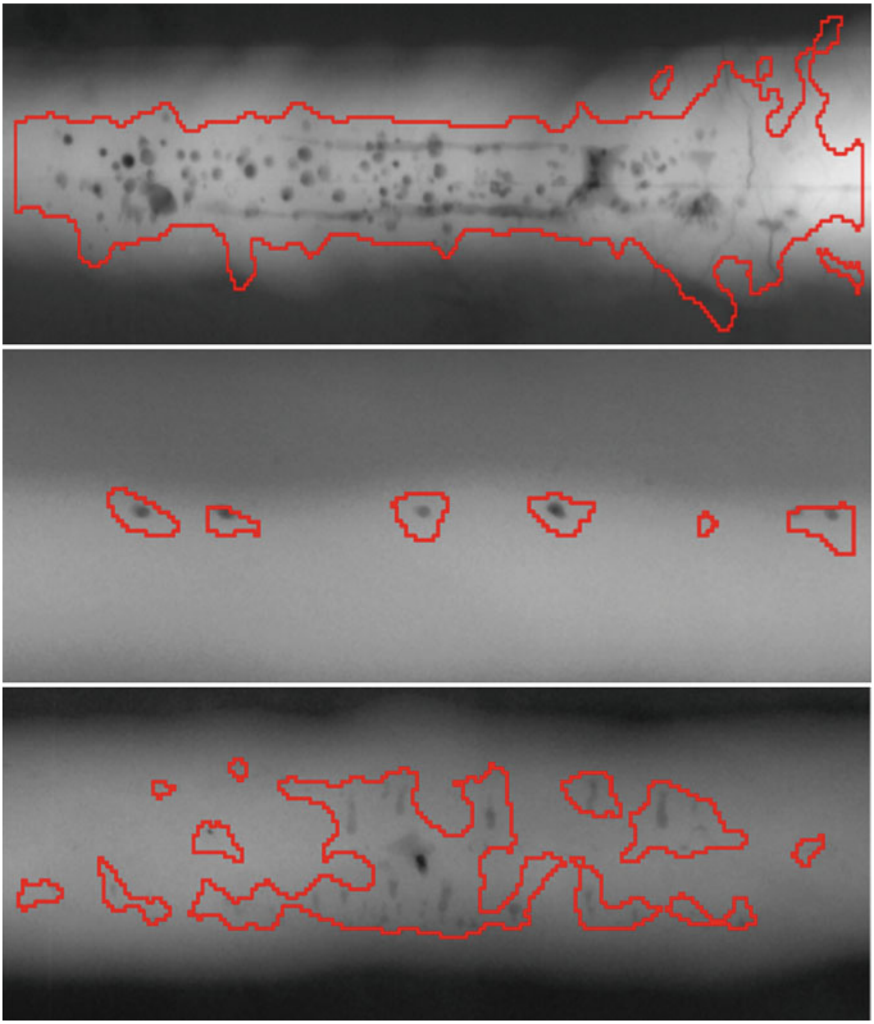


Fig. 8.20 Detection of defects on X-ray images

```
Itrain        = I(:,1650:2399);         % Training image
GTtrain       = J(:,1650:2399);         % Ground truth of training image
Itest         = I(:,900:1649);          % Testing image
GTtest        = J(:,900:1649);          % Ground truth of testing image

% Feature Extraction
options.opf.b = Xfxbuild({'basicint',...
                          'lbpri'});    % Basic intensity features
options.selec = 0;                      % all features
options.m     = 24;                     % size of a window mxm
options.n0    = 400;                    % number of 0 windows
options.n1    = 400;                    % number of 1 windows
options.th0   = 0.02;                   % threshold for 0
options.th1   = 0.02;                   % threshold for 1
options.show  = 1;
options.roi   = Xsegbimodal(Itrain);    % weld segmentation
[X,d,Xn]      = Xfxrandompatches(...    % extraction of patches
                Itrain,GTtrain,options);% and feature extraction

% Feature Selection
sc            = Xfclean(X);             % delete constant and correlated features
Xc            = X(:,sc);                % sc = indices of selected features
Xcn           = Xn(sc,:);
opsfs.show    = 1;                      % display results
opsfs.p       = 15;                     % 15 features will be selected
sx            = Xsfs(Xc,d,opsfs);       % using SFS
selec         = sc(sx);
fx            = X(:,selec);             % selec = indices of selected features

% Training
opc.name      = 'qda';                  % LDA classifier
opc.options.p = [];
opc           = Xclassify(fx,d,opc);

% Detection
options.opc   = opc;
options.nm    = 6;                     % shifted by 24/6=4 pixels
options.Dth   = 24;                    % 24x24 pixels
options.selec = selec;
options.roi   = Xsegbimodal(Itest);    % weld segmentation
[Dmap,Dbin]   = Xsegsliwin(Itest,options);
figure                                 % output
imshow(Itest);title('Detection');
hold on
[yd,xd]    = find(bwperim(Dbin));
plot(xd,yd,'r.')
[ygt,xgt]    = find(bwperim(GTtest));
plot(xgt,ygt,'g.')
legend({'detection','ground truth'})
```

The output of this code is shown in Fig. 8.21. We can see the positive and negative samples that are used for training the classifier and the final detection on testing image. In this example, we use Xfxrandompatches (see Appendix B) to extract the features of the random patches. The patches are extracted only in the region of interest (`options.roi`) defined by the segmentation of the weld. The detection based on sliding-windows is performed by command Xsegsliwin (see Appendix B) of 𝕏VIS Toolbox. The reader can observe the effectiveness of this strategy. It is clear that better results can be achieved by considering more features, classifiers and training images. □

8.4 Baggage

Since the September 11 attacks, automated (or semiautomated) 3D recognition using X-ray images have become a very important element in baggage screening. The inspection process, however, is complex, basically because threatening items are very difficult to detect when placed in close-packed bags, superimposed by other objects, and/or rotated showing an unrecognizable view [67]. In baggage screening, where human security plays an important role and inspection complexity is very

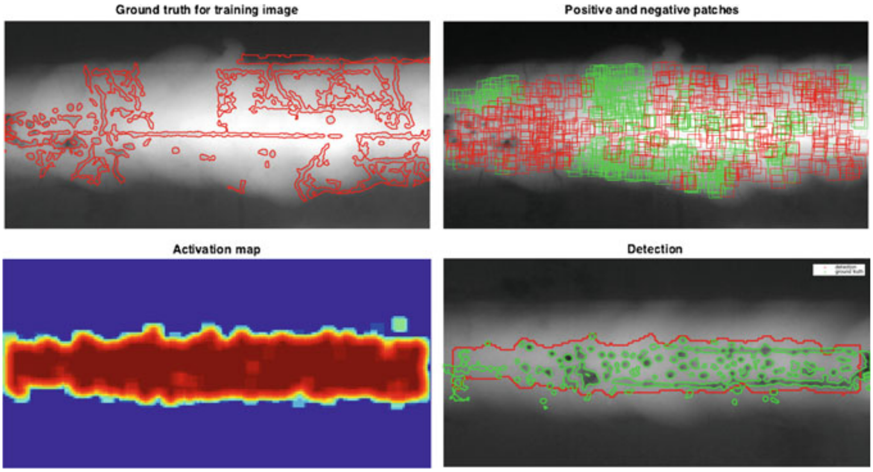

Ground truth for training image
Positive and negative patches
Activation map
Detection
detection
ground truth

Fig. 8.21 Detection of defects on X-ray images using sliding-windows. *Top* Training image (ground truth and patches). *Bottom* Testing image (activation map and detection with ground truth) (→ Example 8.2)

high, human inspectors are still used. Nevertheless, during peak hours in airports, human screeners have only a few seconds to decide whether a bag contains or not a prohibited item, and detection performance is only about 80–90 % [68].

8.4.1 *State of the Art*

Before 9/11, the X-ray analysis of luggage mainly focused on capturing the images of their content: the reader can find in [69] an interesting analysis carried out in 1989 of several aircraft attacks around the world, and the existing technologies to detect terrorist threats based on Thermal-Neutron Activation (TNA), Fast-Neutron Activation (FNA), and dual-energy X-rays (used in medicine since the early 1970s). In the 1990s, Explosive Detection Systems (EDS) were developed based on X-ray imaging [70], and computed tomography through elastic scatter X-ray (comparing the structure of irradiated material, against stored reference spectra for explosives and drugs) [71]. All these works were concentrated on image acquisition and simple image processing; however, they lacked advanced image analysis to improve detection performance. Nevertheless, the 9/11 attacks increased the security measures taken at airports, which in turn stimulated the interest of the scientific community in the research of areas related to security using advanced computational techniques. Over the last decade, the main contributions were: analysis of human inspection [72], pseudo-coloring of X-ray images [73, 74], enhancement and segmentation of X-ray images [75] and detection of threatening items in X-ray images, based on texture features (detecting a 9 mm Colt Beretta automatic (machine) pistol) [76], neural networks and fuzzy rules (yielding about 80 % of performance) [77], and SVM classifier (detecting guns in real time) [78].

In baggage screening, the use of multiple view information yields a significant improvement in performance as certain items are difficult to recognize using only one viewpoint. As reported in a study that measures the human performance in baggage screening [79], (human) multiple view X-ray inspection leads to a higher detection performance of prohibited items under difficult conditions, however, there are no significant differences between the detection performance (single vs. multiple view) for difficult–easy multiple view conditions, i.e., two *difficult* or two *easy* views are redundant. We observed that for intricate conditions, multiple view X-ray inspection is required.

Recently, some algorithms based on multiple X-ray views were reported in the literature. For example, synthesis of new X-ray images obtained from Kinetic Depth Effect X-ray (KDEX) images based on SIFT features in order to increase detection performance [49]; an approach for object detection in multiview dual-energy X-ray with promising preliminary results [54]; X-ray active vision that is able to adequate the viewpoint of the target object in order to obtain better X-ray images to analyze [61]; and tracking across multiple X-ray views in order to verify the diagnoses performed using a single view [5, 57–59].

An example is illustrated in Fig. 8.22. A survey on explosives detection can be found in [80, 81]. Selected approaches are summarized in Table 8.5. In baggage screening, where human security plays an important role and inspection complexity is very high, human inspectors are still used. For intricate conditions, multiple view X-ray inspection using dual-energy is required.

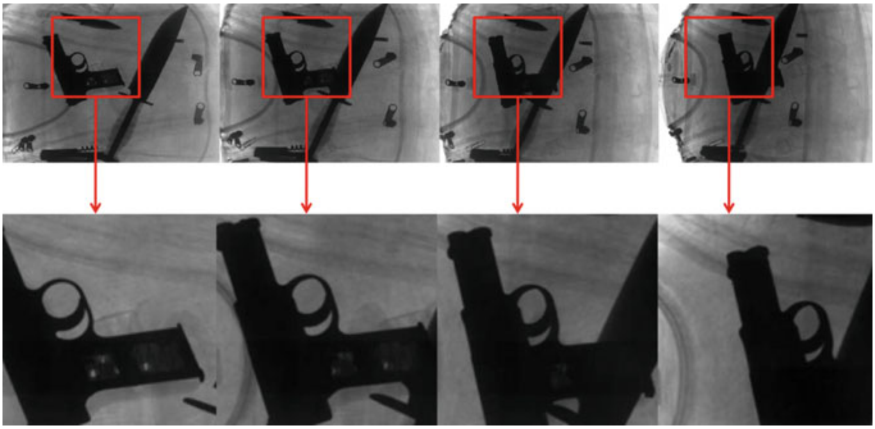


Fig. 8.22 Detection of a handgun based on the trigger identification in multiple views [57]

Table 8.5 Applications on baggage screening

References	Energy		Geometric model[a]	Single views[b]			Active vision	Multiple views[b]		
	Mono	Dual		①	②	③		①	②	③
Abusaeeda (2011) [49]	×	×	C	–	×	–	–	–	×	–
Baştan (2012) [50]	×	×	–	×	×	×	–	–	–	–
Baştan (2013) [51]	×	×	–	×	×	×	–	×	×	×
Chen (2005) [52]	×	×	–	×	×	–	–	–	–	–
Ding (2006) [53]	×	×	–	×	×	×	–	–	–	–
Franzel (2012) [54]	×	×	C	×	×	×	–	×	×	×
Heitz (2010) [55]	×	×	–	×	×	×	–	×	×	×
Mansoor (2012) [56]	×	×	–	×	×	×	–	–	–	–
Mery (2011) [5]	×		N	×	×	×	–	×	×	×
Mery (2013) [57]	×		N	×	×	×	–	×	×	×
Mery (2013) [58]	×		N	×	×	×	–	×	×	×

(continued)

Table 8.5 (continued)

References	Energy		Geometric model[a]	Single views[b]			Active vision	Multiple views[b]		
	Mono	Dual		①	②	③		①	②	③
Mery (2015) [59]	×		N	×	×	×	–	×	×	×
Qiang (2006) [60]	×	×	–	×	×	–	–	–	–	–
Riffo (2011) [61]	×		N	×	×	–	×	×	×	×
Riffo (2015) [62]	×		–	×	×	×	–	–	–	–
Schmidt (2012) [63]	×	×	–	×	×	×	–	–	–	–
Turcsany (2013) [64]	×	×	–	×	×	×	–	–	–	–
Uroukov (2015) [65]	×	×	–	×	×	×	–	–	–	–
Zhang (2015) [66]	×	×	–	×	×	×	–	–	–	–

[a] *C* Calibrated, *N* Not calibrated, – Not used.

[b] See ①, ②, ③ in Fig. 1.21

8.4.2 An Application

In this section, we present the use of an automated method based on multiple X-ray views to recognize certain regular objects with highly defined shapes and sizes. The method consists of two steps: 'monocular analysis', to obtain possible detections in each view of a sequence, and 'multiple view analysis', to recognize the objects of interest using matchings in all views. The search for matching candidates is efficiently performed using a lookup table that is computed offline. In order to illustrate the effectiveness of the proposed method, experimental results on recognizing regular objects (clips, springs and razor blades) in pen cases are shown. In this section, we explain in further detail the proposed method. The strategy consists of two main stages: *offline* and *online*.

Off Line Stage

The first stage, performed offline, consists of two main steps: (i) learning a model that is used for the recognition and (ii) estimation of a multiple view geometric model that is used for data association.

Learning: In this step, we learn a classifier h to recognize parts of the objects that we are attempting to detect. It is assumed that there are $C + 1$ classes (labeled as '0' for nonobject class, and '1', '2', ... 'C' for C different objects). Images are taken of representative objects of each class from different points of view. In order to model the details of the objects from different poses, several keypoints per image are detected, and for each keypoint a descriptor $\mathbf{d}$ is extracted using, for example, LBP, SIFT, HOG, and SURF, among others (see Sect. 5.4). In this supervised approach, each descriptor $\mathbf{d}$ is manually labeled according to its corresponding class $c \in \{0, 1, \ldots C\}$. Given the training data $(\mathbf{d}_t, c_t)$, for $t = 1, \ldots, N$, where N is the total number of descriptors extracted in all training images, a classifier h is designed which maps $\mathbf{d}_t$ to their classification label c_t, thus, $h(\mathbf{d}_t)$ should be c_t. This classifier will be used in the online stage by monocular and multiple view analysis.

Geometry: Our strategy deals with multiple monocular detections in multiple views. In this problem of data association, the aim is to find the correct correspondence among different views. For this reason, we use multiple view geometric constraints to reduce the number of matching candidates between monocular detections. For an image sequence with n views $\mathbf{I}_1 \ldots \mathbf{I}_n$, the fundamental matrices $\{\mathbf{F}_{ij}\}$ between consecutive frames $\mathbf{I}_i$ and $\mathbf{I}_{j=i+1}$ are computed for $i = 1, \ldots, n-1$. In our approach, the fundamental matrix $\mathbf{F}_{ij}$ is calculated from projection matrices $\mathbf{P}_i$ and $\mathbf{P}_j$ that can be estimated using calibration (see Sect. 3.4) or bundle adjustment algorithms (see Sect. 8.4.3).

The geometric constraints are expressed in homogeneous coordinates. Therefore, given a point $\mathbf{m}_i = [x_i \ y_i \ 1]^\mathsf{T}$ in image $\mathbf{I}_i$, a corresponding point $\mathbf{m}_j = [x_j \ y_j \ 1]^\mathsf{T}$ in image $\mathbf{I}_j$ must fulfill: (i) epipolar constraint (see Sect. 3.5.1): $\mathbf{m}_j$ must lie near the epipolar line $\ell = \mathbf{F}_{ij}\mathbf{m}_i$, and (ii) location constraint: for small variations of the point of views between $\mathbf{I}_i$ and $\mathbf{I}_j$, $\mathbf{m}_j$ must lie near $\mathbf{m}_i$. Thus, a candidate $\mathbf{m}_j$ must fulfill:

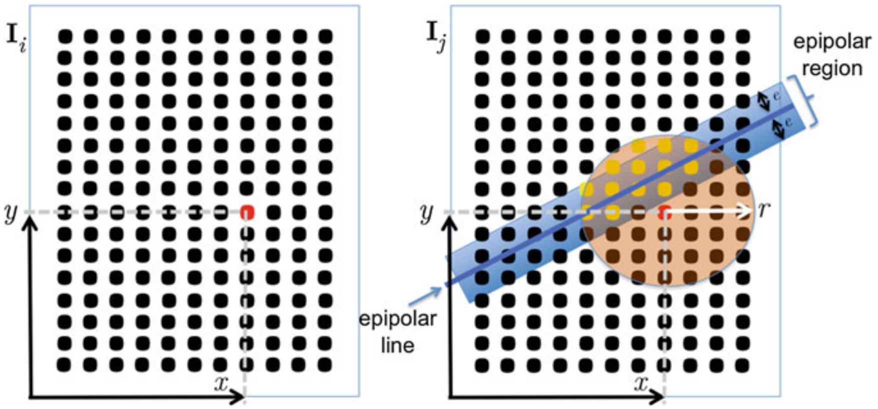


Fig. 8.23 Given the grid point illustrated as the *red point* at (x, y), in image $\mathbf{I}_i$, the set of possible corresponding points in image $\mathbf{I}_j$ can be those grid points (*yellow points*) represented by the intersection of the epipolar region (*blue rectangle*) and neighborhood around (x, y) (*orange circle* with radius r centered at *red point*). The use of grid points allows us to use a lookup table in order to search the matching candidates in $\mathbf{I}_j$ efficiently (Color figure online)

$$\frac{|\mathbf{m}_j^{\mathsf{T}}\mathbf{F}_{ij}\mathbf{m}_i|}{\sqrt{\ell_1^2+\ell_2^2}} < e \text{ and } ||\mathbf{m}_i - \mathbf{m}_j|| < r. \tag{8.5}$$

In order to accelerate the search of candidates, we propose the use of a lookup table as follows: Points in images $\mathbf{I}_i$ and $\mathbf{I}_j$ are arranged in a grid format with rows and columns. For each grid point (x, y) of image $\mathbf{I}_i$, we look for the grid points of image $\mathbf{I}_j$ that fulfill (8.5), as illustrated in Fig. 8.23. Therefore, the possible corresponding points of (x, y) will be the set $\mathbf{S}_{xy} = \{(x_p, y_p)\}_{p=1}^{q}$, where $x_p = X(x, y, p)$, $y_p = Y(x, y, p)$ and $q = Q(x, y)$ are stored (offline) in a lookup table. In the online stage, given a point $\mathbf{m}_i$ (in image $\mathbf{I}_i$), the matching candidates in image $\mathbf{I}_j$ are those that lie near to $\mathbf{S}_{xy}$, where (x, y) is the nearest grid point to $\mathbf{m}_i$. This search can be efficiently implemented using k-d tree structures [16].

In a controlled and calibrated environment, we can assume that the fundamental matrices are stable and we do not need to estimate them in each new image sequence, i.e., the lookup tables are constant. Additionally, when the relative motion of the point of view between consecutive frames is the same, the computed fundamental matrices are constant, i.e., $\mathbf{F}_{ij} = \mathbf{F}$, and we need to store only one lookup table.

Online Stage

The online stage is performed in order to recognize the objects of interest in a test image sequence of n images $\{\mathbf{I}_i\}$, for $i = 1, \ldots, n$. The images are acquired by rotation of the object being tested at β degrees (in our experiments we used $n = 4$, and $\beta = 10^0$). This stage consisted of two main steps: monocular and multiple view analysis that will be described in further detail as follows.

Monocular Analysis: This step is performed in each image $\mathbf{I}_i$ of the test image sequence, as illustrated in Fig. 8.24 in a real case. The whole object contained

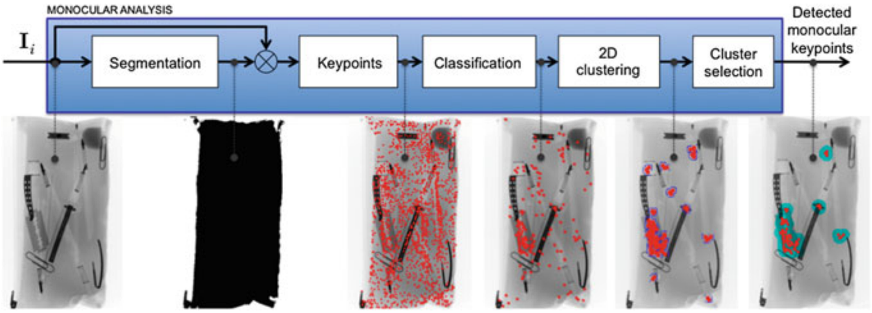


Fig. 8.24 Monocular analysis for each image of the sequence, i.e., for $i = 1, \ldots, n$. In this example, the class of interest is 'razor blade'

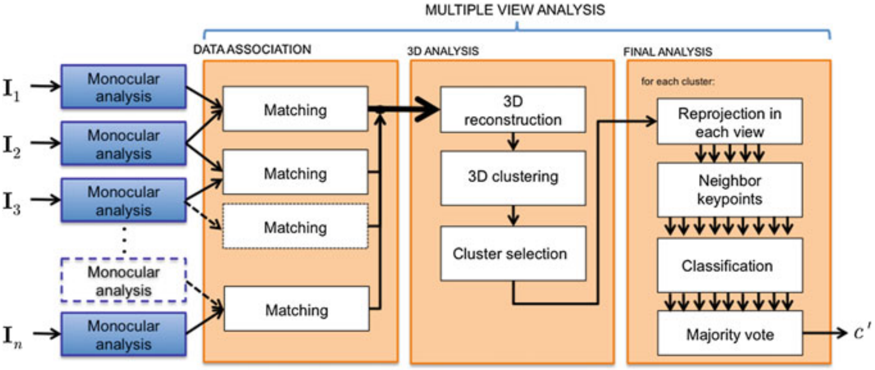


Fig. 8.25 Multiple view analysis. An explanation of last step (final analysis) is illustrated in Fig. 8.26

in image $\mathbf{I}_i$ is segmented from the background using threshold and morphological operations. SIFT-keypoints—or other descriptors–are only extracted in the segmented portion. The descriptor **d** of each keypoint is classified using classifier $h(\mathbf{d})$ trained in the offline stage, and explained above. All keypoints classified as class c, where c is the class of interest, with $c \in \{1 \ldots C\}$ are selected. As we can see in Fig. 8.24 for the classification of 'razor blade', there are many keypoints misclassified. For this reason, neighbor keypoints are clustered in the 2D space using Mean Shift algorithm [82]. Only those clusters that have a large enough number of keypoints are selected. They will be called *detected monocular keypoints*.

Multiple View Analysis: Multiple view analysis performs the recognition of objects of interest in three steps (see Fig. 8.25): (i) data association, (ii) 3D analysis, and (iii) final analysis. The input is the detected monocular keypoints obtained by the mentioned monocular analysis explained above. The output is c', the assigned class for each detected object.

- Data Association: In this step, we find matchings for all detected monocular keypoints in all consecutive images $\mathbf{I}_i$ and $\mathbf{I}_{j=i+1}$, for $i = 1, \ldots, n-1$, as follows:

- For each detected monocular keypoint in image $\mathbf{I}_i$ (located at position (x_i, y_i) with descriptor $\mathbf{d}_i$), we seek in a dense grid of points, the nearest point (x, y) (see red point in Fig. 8.23-left) using a k-d tree structure.
- We determine $\mathbf{S}_{xy}$, the set of matching candidates in image $\mathbf{I}_{j=i+1}$ arranged in a grid manner by reading the lookup table explained above (see yellow points in Fig. 8.23-right).
- We look for the detected monocular keypoints in image $\mathbf{I}_j$ that are located in the neighborhood of $\mathbf{S}_{xy}$, again using a k-d tree structure. They will be called *neighbor keypoints*. When no neighbor keypoint is found, no match is established for (x_i, y_i).
- From neighbor keypoints, we select that one (located at position (x_j, y_j) with descriptor $\mathbf{d}_j$) with minimum distance $||\mathbf{d}_i - \mathbf{d}_j||$. In order to ensure the similarity between matching points, the distance should be less than a threshold ε. If this constraint is not satisfied, again no match is established for (x_i, y_i).

- 3D analysis: From each pair of matched keypoints (x_i, y_i) in image $\mathbf{I}_i$ and (x_j, y_j) in image $\mathbf{I}_{j=i+1}$ established in the previous step, a 3D point is reconstructed using the projection matrices $\mathbf{P}_i$ and $\mathbf{P}_j$ of our geometric model (see Sect. 3.6). Similarly to the monocular detection approach, neighbor 3D points are clustered in the 3D space using Mean Shift algorithm [82], and only those clusters that have a large enough number of 3D points are selected.
- Final analysis: For each selected 3D cluster, all 3D reconstructed points belonging to the cluster are reprojected onto all images of the sequence using the projection matrices of geometric model (see Fig. 8.26). The extracted descriptors of the keypoints located near these reprojected points are classified individually using classifier h. The cluster will be classified as class c' if there is a large number of keypoints individually classified as c', and this number represents a majority in the cluster.

  This majority vote strategy can overcome the problem of false monocular detections when the classification of the minority fails. A cluster can be misclassified if the part that we are trying to recognize is occluded by a part of another class. In this case, there will be keypoints in the cluster assigned to both classes; however, we expect that the majority of keypoints will be assigned to the true class if there are a small number of keypoints misclassified.

Experiments and Results

In our experiments, the task was to recognize three different classes of objects that are present in a pencil case (see for example a sequence in Fig. 8.27a). These classes are: ‘clips’, ‘springs’ and ‘razor blades’. We followed the recognition approach explained above.

In the offline stage, we used a structure-from-motion algorithm in order to estimate the projection matrices of each view.[4] Additionally, in the learning phase, we used only 16 training images of each class. Due to the small intraclass variation

[4] We use in our experiments a fast implementation of multiple view geometry algorithms from Balu Toolbox [83].

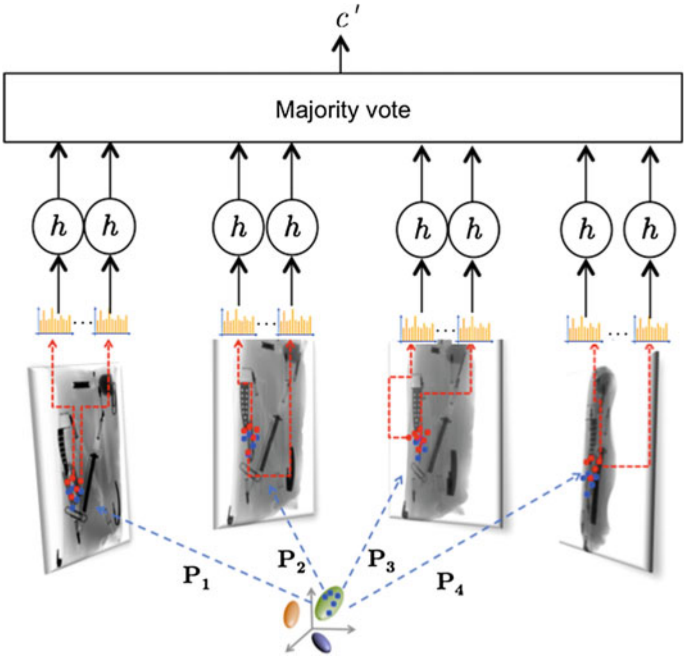


Fig. 8.26 Final analysis: using the geometric model, the reconstructed 3D points in each cluster are reprojected in each view (*blue points*). The keypoints that are near to the reprojected points are identified (*red points*). The descriptors of these keypoints (*orange histograms*) are classified using trained classifier h. The class c' of this cluster is determined by majority vote. In this example of $n = 4$ views, only the *green cluster* is represented

of our classes, this number of training images was deemed sufficient. The training objects were posed at different angles. SIFT descriptors were extracted as explained in [84], and a k-Nearest Neighbor (KNN) classifier with $k = 3$ neighbors was ascertained using the SIFT descriptors of the four classes.[5] Other descriptors (like LBP and HOG) and other classifiers (like SVM or KNN with other values of k) were also tested, although the best performance was achieved with the aforementioned configuration.

In order to illustrate step by step the online stage, the recognition of a razor blade is illustrated in Fig. 8.27a–d for monocular analysis and in Fig. 8.27e–g for multiple view analysis.[6] It is worth mentioning that in monocular detection there are false alarms, however, they can be filtered out after multiple view analysis. The reason is because false alarms cannot be tracked in the sequence or because the tracked points, when validating the corresponding points in other views of the sequence, do not belong to the class of interest. Other results with some degree of overlap, where the task was the recognition of springs and clips, are illustrated in Fig. 8.28.

[5] We used in our experiments fast implementations of SIFT and KNN (based on k-d tree) from VLFeat Toolbox [85].

[6] We used in our experiments a fast implementation of Mean Shift from PMT Toolbox [86].

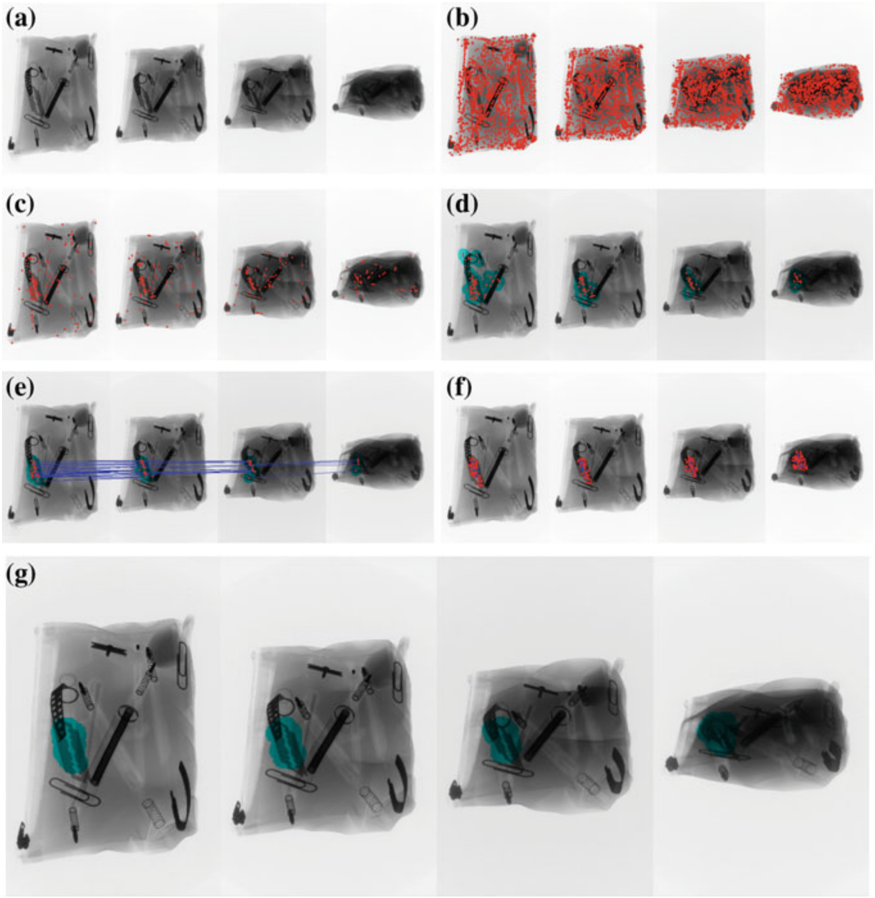


Fig. 8.27 Recognition of a razor blade using our approach. **a** original sequence, **b** keypoints, **c** classified keypoints, **d** detected monocular keypoints, **e** matched keypoints, **f** reprojected 3D points (*blue*) and neighbor keypoints (*red*), and **g** final detection

Testing experiments were carried out by recognizing the three mentioned classes (‘clips’, ‘springs’, and ‘razor blades’) in 45 different sequences of 4 views (15 sequences for each class).[7] The size of an individual image was 1,430 × 900 pixels. In these experiments there were 30 clips, 75 springs, and 15 razor blades to be recognized. A summary of the results using the proposed algorithm is presented in Table 8.6, in which the performance in the recognition of each class is presented in two different parts of our algorithm: after monocular analysis (Mono) and after multiple view analysis (Multi). These parts are illustrated in Fig. 8.27d, g, respectively, for a razor blade. In this table, ground truth (GT) is the number of existing objects to be recognized. The number of detected objects by our algorithm is

[7] The images tested in our experiments come from public GDXray database [87].

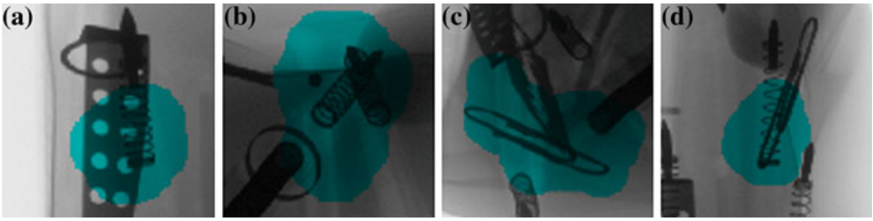


Fig. 8.28 Recognition using our approach in cases with some degree of overlap: **a** one spring, **b** two springs, **c** one clip, and **d** one clip. Each figure shows a part of one image of the whole sequence

Table 8.6 Recognition performance

Class	Mono			Multi		
	TP	FP	GT	TP	FP	GT
Clip	114	127	120	26	2	30
Spring	263	30	300	71	3	75
Blade	59	18	60	14	0	15
Total	436	175	480	111	5	120
PR [%]		71.4			95.7	
RE [%]		90.8			92.5	

$D = TP + FP$, including false positives (FP) and true positives (TP). Ideally, $FP = 0$ and $TP = GT$. In our experiments, precision (PR), computed as $PR = TP/D$, is 71.4 and 95.7 % in each part; and recall (RE), computed as $RE = TP/GT$, is 90.8 and 92.5 % in each step. If we compare single versus multiple view detection, both precision and recall are incremented. Precision, however, is drastically incremented because our approach achieves good discrimination from false alarms.

The amount of time required in our experiments was about 15 min for the offline stage and about 16 s for testing each sequence on a iMac OS X 10.7.3, processor 3.06 GHz Intel Core 2 Duo, 4 GB 1,067 MHz DDR3 memory. The code of the program—implemented in Matlab—is available on our website.

Conclusions

In this section, we presented a new method that can be used to recognize certain parts of interest in complex objects using multiple X-ray views. The proposed method filters out false positives resulting from monocular detection performed on single views by matching information across multiple views. This step is performed efficiently using a lookup table that is computed offline. In order to illustrate the effectiveness of the proposed method, experimental results on recognizing regular objects—clips, springs and razor blades—in pen cases are shown achieving around 93 % accuracy in the recognition of 120 objects. We believe that it would be possible to design an automated aid in a target detection task using the proposed algorithm. In our future work, the approach will be tested in more complex scenarios recognizing objects with a larger intraclass variation.

8.4.3 An Example

In this example, we show how to detect objects in a noncalibrated image sequence as illustrated in Fig. 8.30. The approach has two parts: *structure estimation* and *parts detection*. The approach follows the same strategy of method explained in Sect. 8.2.2.

Structure Estimation
In case the X-ray imaging system is not calibrated, a geometric model must be estimated. The estimation of the geometric model is based on well-known structure-from-motion (SfM) methodologies. For the sake of completeness, a brief description of this model is presented here. In our work, SfM is estimated from a sequence of m images taken from a rigid object at different viewpoints. The original image sequence is stored in m images $\mathbf{J}_1, \ldots, \mathbf{J}_m$.

Keypoints: For each image, SIFT keypoints are extracted because they are very robust against scale, rotation, viewpoint, noise, and illumination changes [84]. Thus, not only a set of 2D image positions $\mathbf{x}$, but also descriptors $\mathbf{y}$, are obtained. Although this method is based on SIFT descriptors, there is no limitation to use other descriptors, e.g., SURF [88].

Image Sorting: If the images are not sorted, a visual vocabulary tree is constructed for fast image indexing. Thus, a new image sequence $\mathbf{I}_1, \ldots, \mathbf{I}_m$ is established from $\mathbf{J}_1, \ldots, \mathbf{J}_m$ by maximizing the total similarity defined as $\sum \text{sim}(\mathbf{I}_i, \mathbf{I}_{i+1})$, for $i = 1, \ldots, m-1$, where the similarity function 'sim' is computed from a normalized scalar product obtained from the visual words of the images [89]. See an example in Fig. 8.29a, b.

Matching Points: For two consecutive images, $\mathbf{I}_i$ and $\mathbf{I}_{i+1}$, SIFT keypoints are matched using the algorithm suggested by Lowe [84] that rejects too ambiguous matches. Afterwards, the Fundamental Matrix between views i and $i+1$, $\mathbf{F}_{i,i+1}$, is estimated using RANSAC [14] to remove outliers. If keypoint k of $\mathbf{I}_i$ is matched with keypoint k' of $\mathbf{I}_{i+1}$, the match will be represented as $\mathbf{x}_{i,k} \rightarrow \mathbf{x}_{i+1,k'}$.

Structure Tracks: We look for all possible structure tracks—with one keypoint in each image of sequence—that belong to a family of the following matches:

$$\mathbf{x}_{1,k_1} \rightarrow \mathbf{x}_{2,k_2} \rightarrow \mathbf{x}_{3,k_3} \rightarrow \cdots \rightarrow \mathbf{x}_{m,k_m}.$$

There are many matches that are eliminated using this approach, however, having a large number of keypoints there are enough tracks to perform the bundle adjustment. We define n as the number of tracks.

Bundle Adjustment: The determined tracks define n image point correspondences over m views. They are arranged as $\mathbf{x}_{i,j}$ for $i = 1, ..., m$ and $j = 1, \ldots, n$. Bundle adjustment estimates 3D points $\hat{\mathbf{X}}_j$ and camera matrices $\mathbf{P}_i$ so that $\sum ||\mathbf{x}_{i,j} - \hat{\mathbf{x}}_{i,j}||$ is minimized, where $\hat{\mathbf{x}}_{i,j}$ is the projection of $\hat{\mathbf{X}}_j$ by $\mathbf{P}_i$. If $n \geq 4$, we can use the

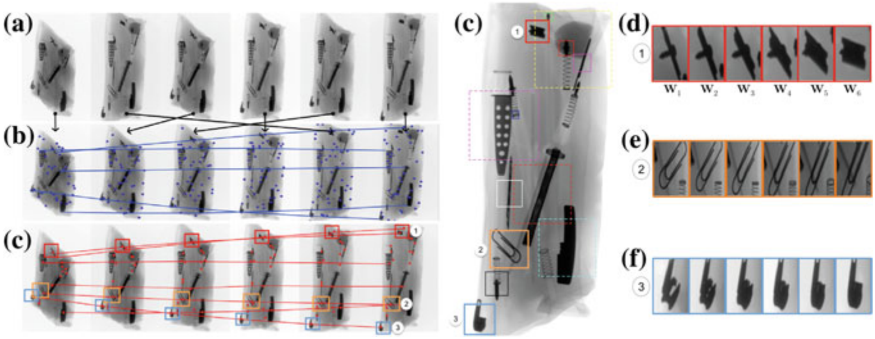


Fig. 8.29 Detection of objects in a pencil case using the proposed method: **a** Unsorted sequence with six X-ray images. The images are sorted according to their similarity (see *arrows*). **b** Sorted sequence, keypoints (points) and structure-from-motion (lines across the sequence). **c** Detection in the sequence and tracked regions. **d** Detection of parts of interest in the last image in the sequence (three of them are used in this example to illustrate the next subfigures). **e** Tracked example regions in each view of the sequence (1: pencil sharpener, 2: clip and 3: zipper slider body and pull-tab) (→ Example 8.3)

factorization algorithm [14] to perform an affine reconstruction because for our purposes the affine ambiguity of 3D space is irrelevant.[8] This method gives a fast and closed-form solution using SVD decomposition. A RANSAC approach is used to remove outliers.

Multiple View Tensors: Bundle adjustment provides a method for computing bifocal and trifocal tensors from projection matrices $\mathbf{P}_i$ [14], that will be used in the next section.

Parts Detection

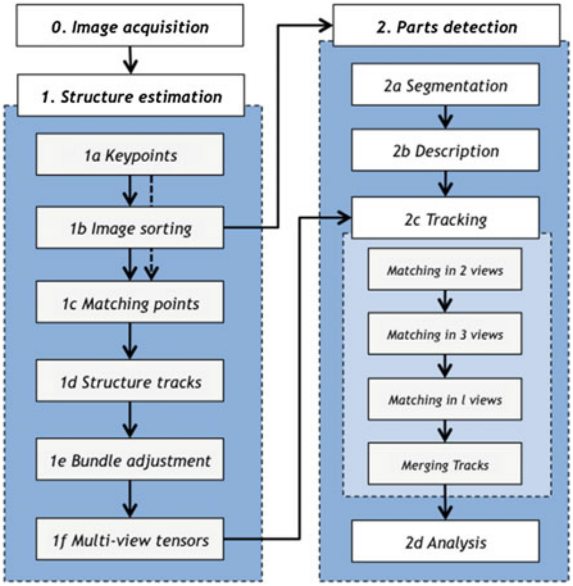
In this section, we give details of the algorithm that detects the object parts of interest. The algorithm consists of four steps: segmentation, description, tracking, and analysis as shown in Fig. 8.30.

Segmentation: Potential regions of interest are segmented in each image $\mathbf{I}_i$ of the sequence. It is an *ad-hoc* procedure that depends on the application. For instance, one can be interested in detecting razor blades or pins in a bag, or flaws in a material, etc. This step ensures the detection of the object parts of interest allowing false detections. The discrimination between these two classes takes place by tracking them across multiple views (see steps 2c and 2d). In our experiments, we tested three segmentation approaches.

- Spots detector: The X-ray image is filtered using a 2D median filter. The difference between original and filtered images is thresholded obtaining a binary image. A potential region r is segmented if size, shape, and contrast criteria are fulfilled.

[8] In this problem, the projective factorization can be used as well [14], however, our simplifying assumption is that only small depth variations occur and an affine model may be used.

Fig. 8.30 Block diagram of the proposed approach



This approach was used to detect small parts (like pen tips or pins in a pencil case).

- Crossing line profile (CLP): Laplacian-of-Gaussian edges are computed from the X-ray image. The closed and connected contours of the edge image define region candidates. Gray-level profiles along straight lines crossing each region candidate in the middle are extracted. A potential region r is segmented if the profile that contains the most similar gray levels in the extremes fulfills contrast criteria [90]. This approach was used to detect discontinuities in a homogeneous material, e.g., flaws in automotive parts.
- SIFT matching: SIFT descriptors are extracted from the X-ray image. They are compared with SIFT descriptors extracted from the image of a reference object of interest. A potential region r is segmented if the descriptors fulfill similarity criteria [84, 91]. This approach was used to detect razor blades in a bag.

Other general segmentation approaches can be used as well. For example, methods based on saliency maps [48], Haar basis features [44], histogram of oriented gradients [45], corner detectors [47], SURF descriptors [88], Maximally Stable regions [92], Local Binary Patterns [93], etc.

Description: Each segmented potential region r is characterized using a SIFT descriptor. The scale of the extracted descriptor, i.e., the width in pixels of the spatial histogram of 4×4 bins is set to $\sqrt{A_r}$, where A_r is the corresponding area of the region r.

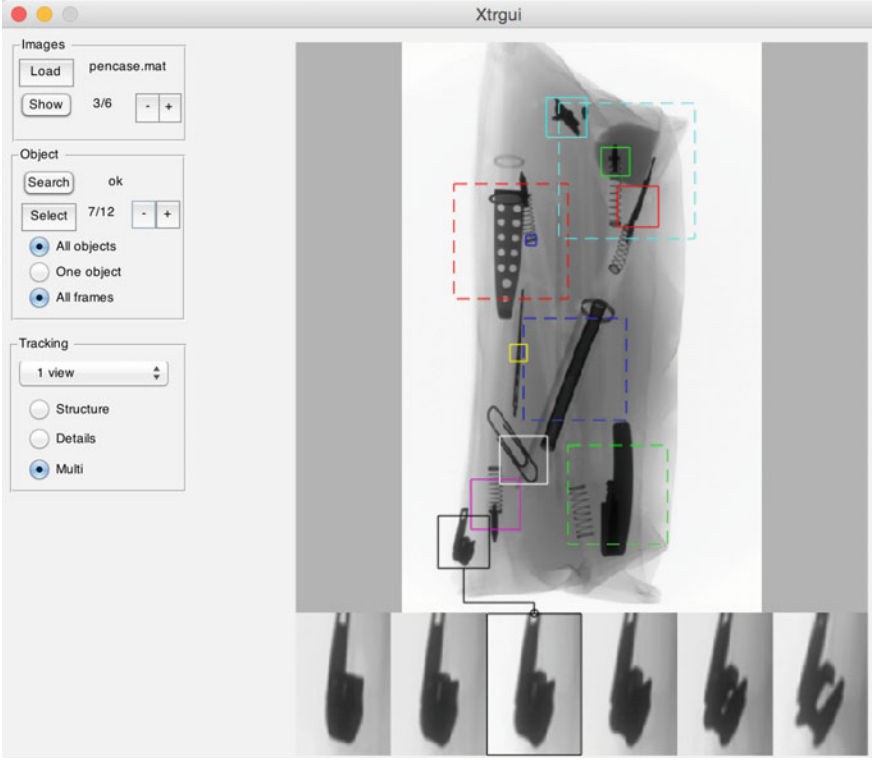
Tracking and Analysis: The tracking and analysis algorithms were covered in detail in Sect. 8.2.2. Results are shown in Fig. 8.29.

Matlab Example 8.3 The approach explained in this section is implemented in Graphic User Interface (Xtrgui (see Appendix B)) of 𝕏vis Toolbox. In this example, we show how to detect objects in a pen case. For this end we have six different views of the object. The parameters of the algorithm are stored in file `pencase.mat`. Thus, we use an MSER approach to segment the potential objects.

Listing 8.3 : Detection of objects in a pen case

```
% DemoXtrgui.m
% 1. Run Xtrgui
% 2. Load pencase.mat
% 3. Press botton 'Search'
% 4. Select 'all objects' and 'all frames'
```

The output of this code is shown in Figs. 8.31 and 8.29. □



Xtrgui

Images

Load pencase.mat

Show 3/6 - +

Object

Search ok

Select 7/12 - +

● All objects

○ One object

● All frames

Tracking

1 view

○ Structure

○ Details

● Multi

Fig. 8.31 Detection of objects in a pen case using graphic user interface Xtrgui (see Appendix B) of 𝕏vis Toolbox. In this example, we can see the zipper slider body and pull-tab in six different views (→ Example 8.3)

8.5 Natural Products

In order to ensure food safety inspection, several applications have been developed by the natural products industry. The difficulties inherent in the detection of defects and contaminants in food products have limited the use of X-ray into the packaged foods sector. However, the need for NDT has motivated a considerable research effort in this field spanning many decades [95].

8.5.1 *State of the Art*

The most important advances are: detection of foreign objects in packaged foods [96]; detection of fish bones in fishes [94]; identification of insect infestation in citrus [97]; detection of codling moth larvae in apples [95]; fruit quality inspection like split-pits, water content distribution and internal structure [98]; and detection of larval stages of the granary weevil in wheat kernels [99]. In these applications, only single view analysis is required. An example is illustrated in Fig. 8.32. A survey can be found in [95]. In Table 8.7, some applications are summarized.

8.5.2 *An Application*

In countries where fish is often consumed, fish bones are some of the most frequently ingested foreign bodies encountered in foods. In the production of fish fillets, fish bone detection is performed by human inspection using their sense of touch and vision which can lead to misclassification. Effective detection of fish bones in the quality control process would help avoid this problem. For this reason, an X-ray machine vision approach to automatically detect fish bones in fish fillets was

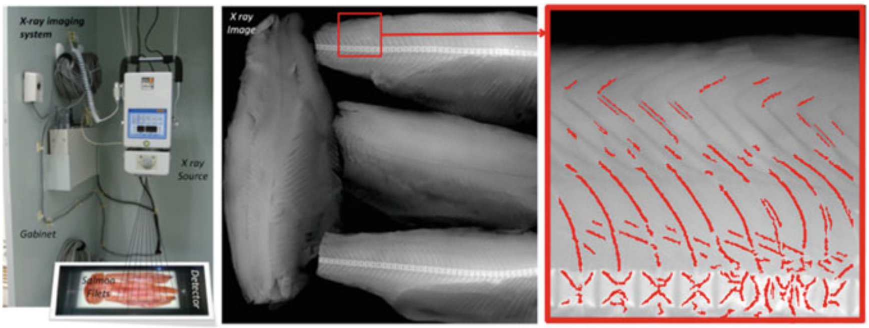


Fig. 8.32 Detection of fish bones using sliding-windows [94]

Table 8.7 Applications on natural products

References	Energy		Geometric model [a]	Single views [b]			Active vision	Multiple views[b]		
	Mono	Dual		①	②	③		①	②	③
Bej (2015) [100]	×		–	×	×	×	–	–	–	–
Haff (2004) [99]	×		–	×	×	×	–	–	–	–
Jiang (2008) [97]	×		–	×	×	×	–	–	–	–
Ogawa (2003) [98]	×		–	×	×	×	–	–	–	–
Karunakaran (2004) [101]	×		–	×	×	×	–	–	–	–
Kelkar (2015) [102]	×		–	×	×	–	–	–	–	–
Kwon (2008) [96]	×		–	×	×	×	–	–	–	–
Mathanker (2011) [103]	×		–	×	×	×	–	–	–	–
Mery (2011) [94]	×		–	×	×	×	–	–	–	–
Neethirajan (2014) [104]	×		–	×	×	–	–	–	–	–
Nielsen (2014) [105]	×		–	×	×	–	–	–	–	–

[a] *C* Calibrated, *N* Not calibrated, – Not used.

[b] See ①, ②, ③ in Fig. 1.21

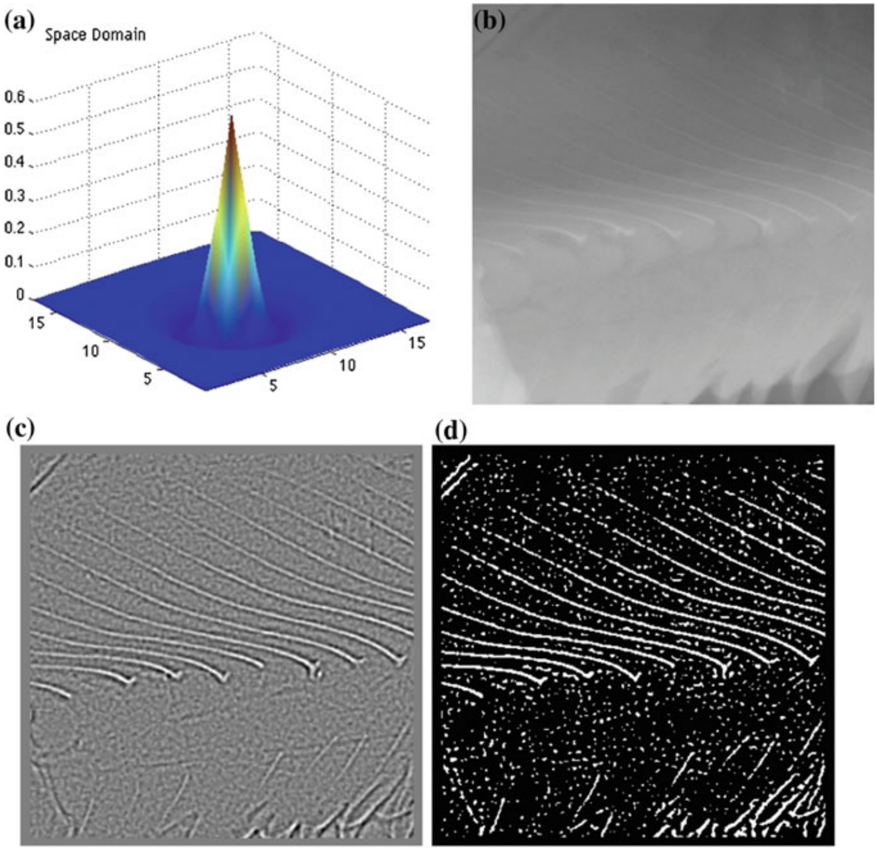


Fig. 8.33 Segmentation of potential fish bones: **a** Convolution mask **H** in space domain, **b** original X-ray image **X** of a salmon fillet, **c** filtered image **Z**, **d** potential fish bones image **P** after thresholding and removing objects deemed too small

developed. This section describes our approach to detect fish bones automatically and the corresponding experiments with salmon and trout fillets based on [94].

Preprocessing and Segmentation

The fish bones are only present in certain space frequencies of the spectrum: they are not too thin (minimal 0.5 mm) nor too thick (maximal 2 mm). The segmentation of potential fish bones is based on a band-pass filter to enhance the fish bones with respect to their surroundings as shown in Fig. 8.33. The proposed approach to detect potential fish bones has four steps:

Enhancement: The original X-ray image **X** (Fig. 8.33b) is enhanced linearly by modifying the original histogram in order to increase contrast [106]: The enhanced image **Y** is

$$\mathbf{Y} = a\mathbf{X} + b \tag{8.6}$$

Band-Pass Filtering: The enhanced image **Y** is filtered using a radial symmetric 17×17 pixels mask **H** (Fig. 8.33a). Mask **H** was estimated from twenty X-ray images by minimizing the error rate as mention in [107] and applied to fish bones (all fish bones should be found and there should be no false alarms). The filtered image **Z** (Fig. 8.33c) is then the convolution of **Y** with mask **H**:

$$\mathbf{Z} = \mathbf{Y} * \mathbf{H} \tag{8.7}$$

Thresholding: Those pixels in **Z** that have gray values greater than a certain threshold θ are marked in a binary image **B**. The threshold is defined to ensure that all fish bones are detected, i.e., false alarms are allowed in this step. The pixels of **B** are defined as

$$B_{ij} = \begin{cases} 1 \text{ if } Z_{ij} > \theta \\ 0 \text{ else} \end{cases} \tag{8.8}$$

Removal of Small Objects: All connected pixels in **B** containing fewer than A pixels are removed as shown in Fig. 8.33d. This image, called **P**, defines the potential fish bones.

Feature Extraction, Selection, and Classification

The segmented potential fish bones—contained in image **P**—are divided into small 10×10 pixels windows called *detection windows*. In a training phase, using a priori knowledge of the fish bones, the detection windows are manually labeled as one of two classes: *bones* and *no-bones*. The first class corresponds to those regions where the potential fish bones are indeed fish bones. Alternatively, the second class corresponds to false alarms. Intensity features of the enhanced X-ray image **Y** are extracted for both classes. We use enhanced image **Y**, instead of preprocessed image **X**, because after our experiments the detection performance was higher. Features extracted from each area of an X-ray image region are divided into four groups as shown in Sect. 8.3.2. In these experiments, 279 features are extracted from each detection window. Afterwards, the features are selected in order to decide on the relevant features for the two defined classes. In addition, a classifier is designed. The best results, after evaluation a 10-fold cross-validation was achieved by Sequential Forward Selection (as feature selection technique) and Support Vector Machine with RBF kernel (as classifier).

Experimental Results

First, the proposed method was tested with 20 representative salmon fillets obtained at a local fish market. The average size of these fillets was 15×10 cm^2. According to preprocessing and segmentation techniques explained above, several regions of interest were obtained where fish bones could be located. The area occupied by these regions of interest corresponds to approx. 12 % of the salmon fillets as shown in Fig. 8.33.

From the mentioned regions of interest 7,697, detection windows of 10×10 pixels were obtained (available in series N0003 of 𝔾𝔻𝕏ray). Each window was labeled with '1' for class *bones* and '0' for *no–bones* (see file `labels.txt` in directory of N0003). From each window, 279 features were extracted. After the feature extraction, 75 % of the samples from each class were randomly chosen to perform the feature selection. The best performance was achieved using Sequential Forward Selection. 24 features were selected. The features gave information about the spatial distribution of pixels, i.e., how coarse or fine the texture is. The selected features correspond mainly to statistical features (12) and filter banks (7), however, it is worth nothing that the two most discriminative features are LBP features (in this case LBP 48 and LBP 11). On the other hand, from the standard features there is only one feature (standard deviation of the intensity) (Fig. 8.34).

In order to investigate the sensibility (S_n) and 1-specificity ($1 - S_p$) of the fish bones depending on their largeness, three size groups were constructed: *large* for fish bones larger than 12 mm, *small* for fish bones smaller than 8.5 mm, and *medium* for fish bones between both sizes. In this experiment, 3,878 fish bones were manually selected. The performance was calculated using a cross-validation with 5-folds. The results are summarized in Fig. 8.35. All medium and large fish bones were detected (with $1 - S_p = 0$ and 3 %, respectively), whereas 93 % of small fish bones were correctly detected with $1 - S_p = 6$ %. This means that cross-validation yielded a detection performance of 100, 98.5 and 93.5 % (computed using $(S_n + S_p)/2$) for large, medium and small fish bones, respectively.

Finally, in order to validate the proposed methodology, the last experiment was carried out using representative fish bones and representative trout fillets provided by a Chilean salmon industry. The size of the fish bones were between 14 and 47 mm (larger than the small-size and mid-size groups considered above). The fish bones were arranged in strips that were superimposed onto trout fillets. Thus, the number of fish bones to be detected was a priori known. According to the absorption law, an X-ray image of a fillet with a fish bone inside, and an X-ray image with a fish bone laid on the fillet top are almost identical. Similar methodologies are used in industrial X-ray inspection of materials in order to simulate discontinuities [17]. The only difference could be that the position of a real fish bone (inside of a fillet) achieves a more realistic location related to the fish tissues, however, after our experience, the obtained images were found to be very similar. Figure 8.36 shows the detection of one fish bone strip on a trout fillet. Using the same classifier trained in the last experiment, i.e., no new training was necessary, the proposed method was able to detect all fish bones with a 1 % false positive rate. In this case, 15 X-ray images were tested, with 459 *bones* and 10,413 *no-bones*.

Conclusion

The need for more information on the quality control of several fish types by means of quantitative methods can be satisfied using X-ray testing, a nondestructive technique that can be used to objectively measure intensity and geometric patterns in nonuniform surfaces. In addition the method can also determine other physical features such as image texture, morphological elements, and defects in order to automatically determine the quality of a fish fillet. The promising results outlined in this work show that a very high classification rate was achieved in the quality control

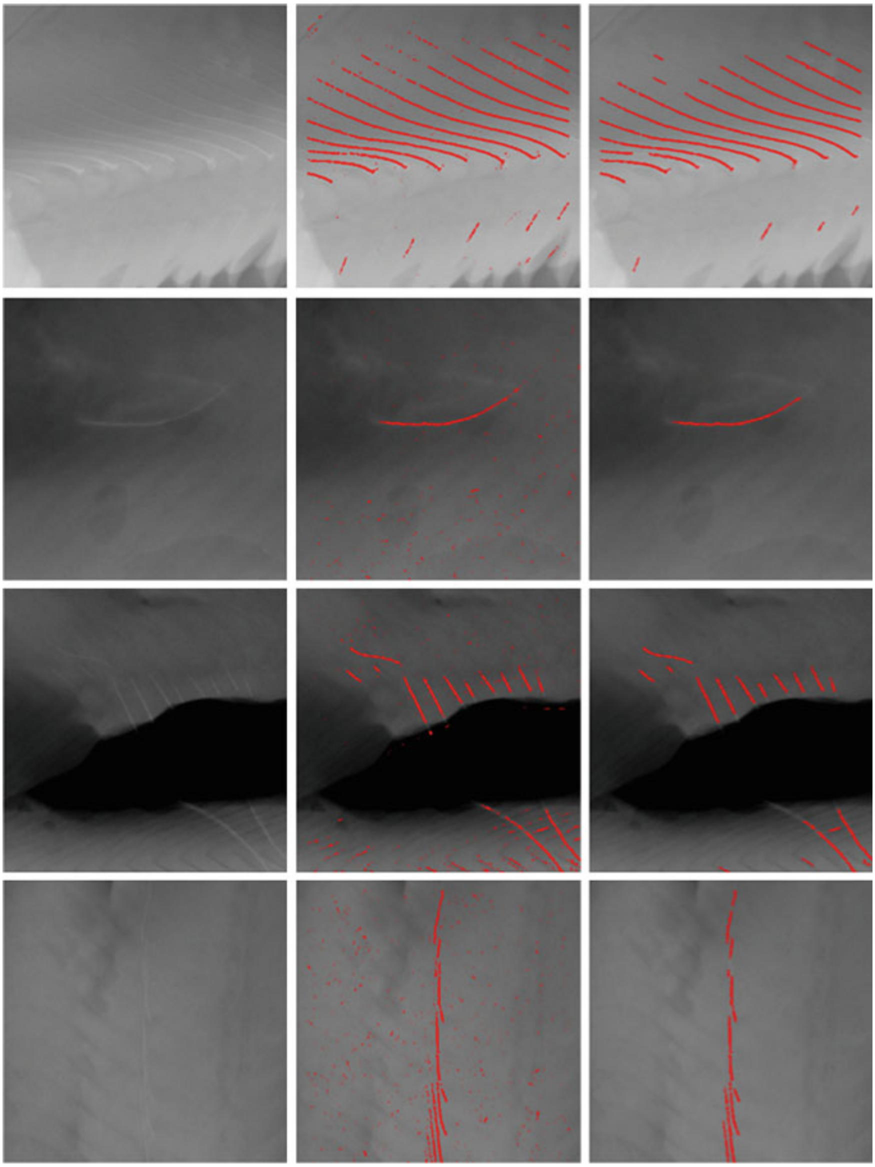
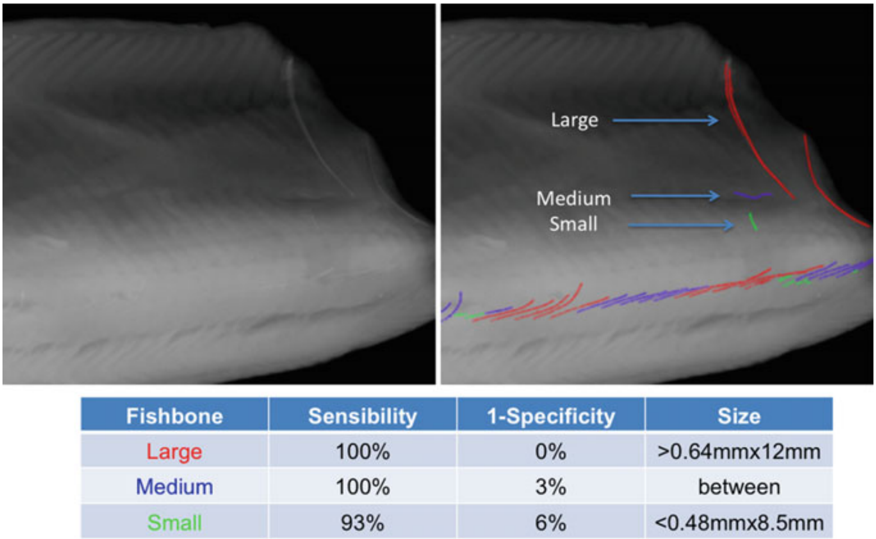


Fig. 8.34 Results obtained in four X-ray images. The columns correspond to enhanced images, class ified fish bones and post processed fish bones. The *first row* corresponds to the example shown in Fig. 8.33

of salmon and trout when using a large number of features combined with efficient feature selection and classification. The key idea of the proposed method was to select, from a large universe of features, only those features that were relevant for the



Fishbone	Sensibility	1-Specificity	Size
Large	100%	0%	>0.64mmx12mm
Medium	100%	3%	between
Small	93%	6%	<0.48mmx8.5mm

Fig. 8.35 Results obtained on 3,878 samples using cross-validation with five folds

separation of the classes. Cross-validation yielded a detection performance of 100, 98.5 and 93.5 % for large, medium, and small fish bones, respectively. The proposed method was validated on trout with representative fish bones provided by a Chilean salmon industry yielding a performance of 99 %. Although the method was validated with salmon and trout fillets only, we believe that the proposed approach opens new possibilities not only in the field of automated visual inspection of salmons and trout, but also in other similar fish.

8.5.3 An Example

In order to illustrate the methodology explained in the previous section, the reader can see Example 6.9, where the whole process is presented. In this example, 200 small X-ray images (100 × 100 pixels) of salmon filets, 100 with fish bones, and 100 without fish bones are used. The images are available in series `N0002` of 𝔾𝔻𝕏ray.

8.6 Further Applications

There are many applications in which X-rays can be used as a NDT and E method. In this section, we mention only cargos and electronic circuits.

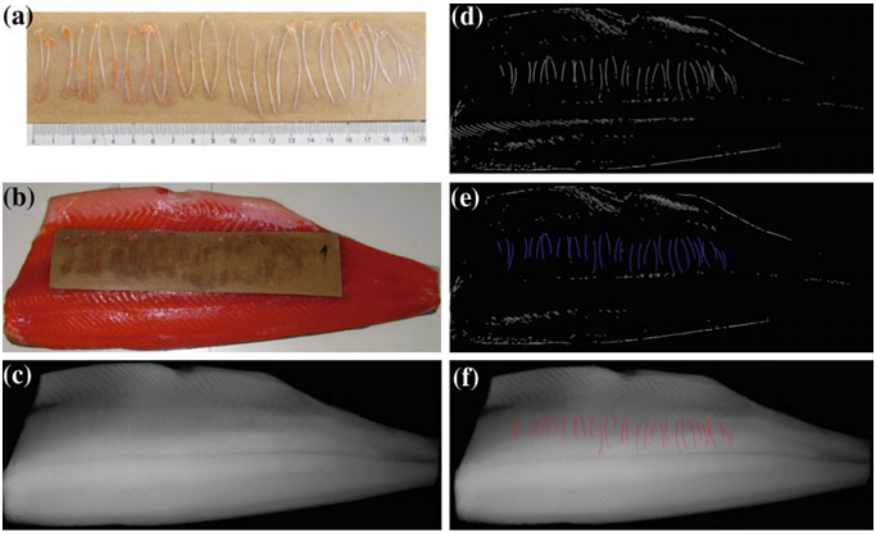


Fig. 8.36 Results obtained on a trout fillet using a fish bone strip with 33 fish bones: **a** strip, **b** strip over the fillet, **c** X-ray image, **d** segmentation, **e** classification, and **f** post-processing. All fish bones were detected ($S_n = 1$), in this example there was no false alarm ($1 - S_p = 0$)

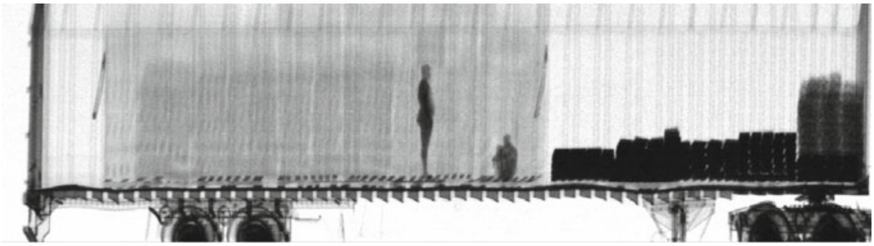


Fig. 8.37 X-ray image of a cargo. Collected by U.S. Customs and Border Protection a bureau of the United States Department of Homeland Security, via Wikimedia Commons

8.6.1 Cargo Inspection

With the ongoing development of international trade, cargo inspection becomes more and more important. X-ray testing has been used for the evaluation of the contents of cargo, trucks, containers, and passenger vehicles to detect the possible presence of many types of contraband. See an example in Fig. 8.37. Some approaches are presented in Table 8.8. There still is not much research on cargo inspection, and the complexity of this inspection task is very high. For this reason, X-ray systems are still only semiautomatic, and they require human supervision.

Table 8.8 Applications on cargo inspection

References	Energy		Geometric Model[a]	Single views[b]			Active vision	Multiple views[b]		
	Mono	Dual		①	②	③		①	②	③
Duan (2008) [108]	×		–	×	×	×	–	×	×	–
Frosio (2011) [109]	×		–	×	×	×	–	×	×	–
Kolkoori (2014) [110]	×		–	×	×	–	–	–	–	–
Zhu (2008) [111]	×		–	×	×	×	–	×	×	–
Zhu (2010) [112]	×		–	×	×	×	–	×	×	–

[a] *C* Calibrated, *N* Not calibrated, – Not used.

[b] See ①, ②, ③ in Fig. 1.21

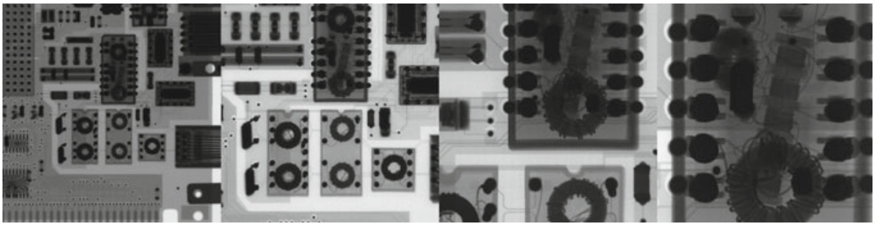


Fig. 8.38 X-ray image of a printed circuit board. By SecretDisc (Own work) via Wikimedia Commons

8.6.2 Electronic Circuits

In this industrial application of X-rays, the idea is to inspect circuit boards or integrated circuits in order to detect flaws in manufacturing, e.g., broken traces, missing components, cracks, dilapidations, etc. An example is shown in Fig. 8.38. Some approaches are presented in Table 8.9. In this area, automated systems are very effective, and the inspection task is very fast and obtains a high performance.

8.7 Summary

In this chapter, relevant applications on X-ray testing were described. We covered X-ray testing in:

- Castings: To ensure the safety of the construction of automative parts, it is necessary to check every part thoroughly using X-ray testing. We presented the state of the art, a defect detection approach based on a tracking principle, and a Matlab implementation of a classifier that is able to defect casting defects in single X-ray images.
- Welds: In welding processes, a mandatory inspection using X-ray testing is required in order to detect defects like porosity, inclusion, lack of fusion, lack of penetration and cracks. We presented the state of the art, a defect detection approach based on sliding-windows, and a Matlab implementation of a classifier that is able to detect defects using sliding-windows methodology in single X-ray images.
- Baggage: In baggage screening, every piece of luggage must be inspected using X-ray testing in order to detect dangerous objects. We presented the state of the art, a recognition approach based on multiple view analysis, and a Matlab implementation of tracking principle that is able to detect objects in the sequence X-ray images of a pen case.
- Natural products: We presented some applications of X-ray testing in natural products, such as inspection of fruit, identification of infections and detection of fish bones. We reviewed the state of the art, a fish bones detection approach based

Table 8.9 Applications on electronic circuit boards

References	Energy		Geometric model[a]	Single views[b]			Active vision	Multiple views[b]		
	Mono	Dual		①	②	③		①	②	③
Mahmood (2015) [113]	×		–	×	×	×	–	–	–	–
Uehara (2013) [114]	×		–	×	×	×	–	–	–	–
Wang (2014) [115]	×		–	×	×	×	–	–	–	–
Wu (2014) [116]	×		–	×	×	×	–	–	–	–

[a] *C* Calibrated, *N* Not calibrated, – Not used.

[b] See ①, ②, ③ in Fig. 1.21

on sliding-windows, and a Matlab implementation of a classifier that is able to detect fish bones in cropped images with and without fish bones.

- Others: There are several industrial applications that use X-ray testing. We mentioned only cargos and electronic circuits giving some references of the state of art.

References

1. Mery, D.: Automated radioscopic testing of aluminum die castings. Mater. Eval. **64**(2), 135–143 (2006)
2. Carrasco, M., Mery, D.: Automatic multiple view inspection using geometrical tracking and feature analysis in aluminum wheels. Mach. Vis. Appl. **22**(1), 157–170 (2011)
3. Li, X., Tso, S.K., Guan, X.P., Huang, Q.: Improving automatic detection of defects in castings by applying wavelet technique. IEEE Trans. Ind. Electron. **53**(6), 1927–1934 (2006)
4. Mery, D., Filbert, D.: Automated flaw detection in aluminum castings based on the tracking of potential defects in a radioscopic image sequence. IEEE Trans. Robot. Autom. **18**(6), 890–901 (2002)
5. Mery, D.: Automated detection in complex objects using a tracking algorithm in multiple X-ray views. In: Proceedings of the 8th IEEE Workshop on Object Tracking and Classification Beyond the Visible Spectrum (OTCBVS 2011), in Conjunction with CVPR 2011, pp. 41–48. Colorado Springs (2011)
6. Pieringer, C., Mery, D.: Flaw detection in aluminium die castings using simultaneous combination of multiple views. Insight **52**(10), 548–552 (2010)
7. Pizarro, L., Mery, D., Delpiano, R., Carrasco, M.: Robust automated multiple view inspection. Pattern Anal. Appl. **11**(1), 21–32 (2008)
8. Ramírez, F., Allende, H.: Detection of flaws in aluminium castings: a comparative study between generative and discriminant approaches. Insight-Non-Destructive Test. Cond. Monit. **55**(7), 366–371 (2013)
9. Tang, Y., Zhang, X., Li, X., Guan, X.: Application of a new image segmentation method to detection of defects in castings. Int. J. Adv. Manuf. Technol. **43**(5–6), 431–439 (2009)
10. Zhao, X., He, Z., Zhang, S.: Defect detection of castings in radiography images using a robust statistical feature. JOSA A **31**(1), 196–205 (2014)
11. Zhao, X., He, Z., Zhang, S., Liang, D.: A sparse-representation-based robust inspection system for hidden defects classification in casting components. Neurocomputing **153**(0), 1–10 (2015)
12. Faugeras, O.: Three-Dimensional Computer Vision: A Geometric Viewpoint. The MIT Press, Cambridge (1993)
13. Mery, D., Filbert, D.: Classification of potential defects in automated inspection of aluminium castings using statistical pattern recognition. In: 8th European Conference on Non-Destructive Testing (ECNDT 2002), pp. 1–10. Barcelona (2002)
14. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2003)
15. Noble, A., Gupta, R., Mundy, J., Schmitz, A., Hartley, R.: High precision X-ray stereo for automated 3D CAD-based inspection. IEEE Trans. Robot. Autom. **14**(2), 292–302 (1998)
16. Bentley, J.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)
17. Mery, D.: Flaw simulation in castings inspection by radioscopy. Insight **43**(10), 664–668 (2001)
18. Mery, D., Filbert, D., Jaeger, T.: Image processing for fault detection in aluminum castings. In: MacKenzie, D., Totten, G. (eds.) Analytical Characterization of Aluminum and Its Alloys. Marcel Dekker, New York (2003)

19. Filbert, D., Klatte, R., Heinrich, W., Purschke, M.: Computer aided inspection of castings. In: IEEE-IAS Annual Meeting, pp. 1087–1095. Atlanta (1987)
20. Heinrich, W.: Automated inspection of castings using X-ray testing. Ph.D. thesis, Institute for Measurement and Automation, Faculty of Electrical Engineering, Technical University of Berlin (1988) (in German)
21. Hecker, H.: Ein neues Verfahren zur robusten Röntgenbildauswertung in der automatischen Gußteilprüfung. Ph.D. thesis, vom Fachbereich Elektrotechnik, Technische Universität Berlin (1995)
22. Mery, D., Filbert, D., Parspour, N.: Improvement in automated aluminum casting inspection by finding correspondence of potential flaws in multiple radioscopic images. In: Proceedings of the 15th World Conference on Non-Destructive Testing (WCNDT-2000). Rome (2000)
23. da Silva, R., Mery, D.: State-of-the-art of weld seam inspection using X-ray testing: part I—image processing. Mater. Eval. **65**(6), 643–647 (2007)
24. da Silva, R., Mery, D.: State-of-the-art of weld seam inspection using X-ray testing: part II—pattern recognition. Mater. Eval. **65**(9), 833–838 (2007)
25. Lindgren, E.: Detection, 3-D positioning, and sizing of small pore defects using digital radiography and tracking. EURASIP J. Adv. Signal Process. **2014**(1), 1–17 (2014)
26. Shao, J., Du, D., Chang, B., Shi, H.: Automatic weld defect detection based on potential defect tracking in real-time radiographic image sequence. NDT E Int. **46**, 14–21 (2012)
27. Anand, R., Kumar, P., et al.: Flaw detection in radiographic weldment images using morphological watershed segmentation technique. NDT E Int. **42**(1), 2–8 (2009)
28. Baniukiewicz, P.: Automated defect recognition and identification in digital radiography. J. Nondestruct. Eval. **33**(3), 327–334 (2014)
29. Gao, W., Hu, Y.H.: Real-time X-ray radiography for defect detection in submerged arc welding and segmentation using sparse signal representation. Insight-Non-Destructive Test. Cond. Monit. **56**(6), 299–307 (2014)
30. Kaftandjian, V., Dupuis, O., Babot, D., Zhu, Y.M.: Uncertainty modelling using Dempster-Shafer theory for improving detection of weld defects. Pattern Recognit. Lett. **24**(1), 547–564 (2003)
31. Kumar, J., Anand, R., Srivastava, S.: Flaws classification using ANN for radiographic weld images. In: 2014 International Conference on Signal Processing and Integrated Networks (SPIN), pp. 145–150 (2014)
32. Kumar, J., Anand, R., Srivastava, S.: Multi-class welding flaws classification using texture feature for radiographic images. In: 2014 International Conference on Advances in Electrical Engineering (ICAEE), pp. 1–4 (2014)
33. Liao, T.: Classification of weld flaws with imbalanced class data. Expert Syst. Appl. **35**(3), 1041–1052 (2008)
34. Liao, T.W.: Improving the accuracy of computer-aided radiographic weld inspection by feature selection. NDT E Int. **42**, 229–239 (2009)
35. Mery, D., Berti, M.A.: Automatic detection of welding defects using texture features. Insight-Non-Destructive Test. Cond. Monit. **45**(10), 676–681 (2003)
36. Mery, D.: Automated detection of welding defects without segmentation. Mater. Eval. **69**(6), 657–663 (2011)
37. Mu, W., Gao, J., Jiang, H., Wang, Z., Chen, F., Dang, C.: Automatic classification approach to weld defects based on PCA and SVM. Insight-Non-Destructive Test. Cond. Monit. **55**(10), 535–539 (2013)
38. Shi, D.H., Gang, T., Yang, S.Y., Yuan, Y.: Research on segmentation and distribution features of small defects in precision weldments with complex structure. NDT E Int. **40**, 397–404 (2007)
39. da Silva, R.R., Calôba, L.P., Siqueira, M.H., Rebello, J.M.: Pattern recognition of weld defects detected by radiographic test. NDT E Int. **37**(6), 461–470 (2004)
40. Vilar, R., Zapata, J., Ruiz, R.: An automatic system of classification of weld defects in radiographic images. NDT E Int. **42**(5), 467–476 (2009)

41. Wang, Y., Sun, Y., Lv, P., Wang, H.: Detection of line weld defects based on multiple thresholds and support vector machine. NDT E Int. **41**(7), 517–524 (2008)
42. Yirong, Z., Dong, D., Baohua, C., Linhong, J., Jiluan, P.: Automatic weld defect detection method based on Kalman filtering for real-time radiographic inspection of spiral pipe. NDT E Int. **72**, 1–9 (2015)
43. Zapata, J., Vilar, R., Ruiz, R.: Automatic inspection system of welding radiographic images based on ANN under a regularisation process. J. Nondestruct. Eval. **31**(1), 34–45 (2012)
44. Viola, P., Jones, M.: Robust real-time object detection. Int. J. Comput. Vis. **57**(2), 137–154 (2004)
45. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Conference on Computer Vision and Pattern Recognition (CVPR2005), vol. 1, pp. 886–893 (2005)
46. Liao, T.: Classification of welding flaw types with fuzzy expert systems. Fuzzy Sets Syst. **108**, 145–158 (2003)
47. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of 4th Alvey Vision Conferences, pp. 147–152 (1988)
48. Montabone, S., Soto, A.: Human detection using a mobile platform and novel features derived from a visual saliency mechanism. Image Vis. Comput. **28**(3), 391–402 (2010)
49. Abusaeeda, O., Evans, J., Downes, D., Chan, J.: View synthesis of KDEX imagery for 3D security X-ray imaging. In: Proceedings of 4th International Conference on Imaging for Crime Detection and Prevention (ICDP-2011) (2011)
50. Baştan, M., Yousefi, M.R., Breuel, T.M.: Visual words on baggage X-ray images. Computer Analysis of Images and Patterns, pp. 360–368. Springer, New York (2011)
51. Baştan, M., Byeon, W., Breuel, T.M.: Object recognition in multi-view dual X-ray images. In: British Machine Vision Conference BMVC (2013)
52. Chen, Z., Zheng, Y., Abidi, B.R., Page, D.L., Abidi, M.A.: A combinational approach to the fusion, de-noising and enhancement of dual-energy X-ray luggage images. In: Workshop of IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2005) (2005)
53. Ding, J., Li, Y., Xu, X., Wang, L.: X-ray image segmentation by attribute relational graph matching. In: 8th IEEE International Conference on Signal Processing, vol. 2 (2006)
54. Franzel, T., Schmidt, U., Roth, S.: Object detection in multi-view X-ray images. Pattern Recognit. **7476**, 144–154 (2012)
55. Heitz, G., Chechik, G.: Object separation in X-ray image sets. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2010), pp. 2093–2100 (2010)
56. Mansoor, M., Rajashankari, R.: Detection of concealed weapons in X-ray images using fuzzy K-NN. Int. J. Comput. Sci. Eng. Inf. Technol. **2**(2), 187–196 (2012)
57. Mery, D., Riffo, V., Mondragon, G., Zuccar, I.: Detection of regular objects in baggages using multiple X-ray views. Insight **55**(1), 16–21 (2013)
58. Mery, D., Riffo, V., Zuccar, I., Pieringer, C.: Automated X-ray object recognition using an efficient search algorithm in multiple views. In: Proceedings of the 9th IEEE CVPR Workshop on Perception Beyond the Visible Spectrum, Portland (2013)
59. Mery, D.: Inspection of complex objects using multiple-X-ray views. IEEE/ASME Trans. Mechatron. **20**(1), 338–347 (2015)
60. Lu, Q., Conners, R.: Using image processing methods to improve the explosive detection accuracy. IEEE Trans. Appl. Rev. Part C: Syst Man Cybern. **36**(6), 750–760 (2006)
61. Riffo, V., Mery, D.: Active X-ray testing of complex objects. Insight **54**(1), 28–35 (2012)
62. Riffo, V., Mery, D.: Automated detection of threat objects using Adapted Implicit Shape Model. IEEE Transactions on Systems, Man, and Cybernetics: Systems (2015). doi:10.1109/TSMC.2015.2439233
63. Schmidt-Hackenberg, L., Yousefi, M.R., Breuel, T.M.: Visual cortex inspired features for object detection in X-ray images. In: 2012 21st International Conference on Pattern Recognition (ICPR), pp. 2573–2576. IEEE (2012)
64. Turcsany, D., Mouton, A., Breckon, T.P.: Improving feature-based object recognition for X-ray baggage security screening using primed visualwords. In: IEEE International Conference on Industrial Technology (ICIT), pp. 1140–1145 (2013)

65. Uroukov, I., Speller, R.: A preliminary approach to intelligent X-ray imaging for baggage inspection at airports. Signal Process. Res. **4**, 1–11 (2015)
66. Zhang, N., Zhu, J.: A study of X-ray machine image local semantic features extraction model based on bag-of-words for airport security. Int. J. Smart Sens. Intell. Syst. **1**, 45–64 (2015)
67. Zentai, G.: X-ray imaging for homeland security. In: IEEE International Workshop on Imaging Systems and Techniques (IST 2008), pp. 1–6 (2008)
68. Michel, S., Koller, S., de Ruiter, J., Moerland, R., Hogervorst, M., Schwaninger, A.: Computer-based training increases efficiency in X-ray image interpretation by aviation security screeners. In: 2007 41st Annual IEEE International Carnahan Conference on Security Technology, pp. 201–206 (2007)
69. Murphy, E.: A rising war on terrorists. IEEE Spectr. **26**(11), 33–36 (1989)
70. Murray, N., Riordan, K.: Evaluation of automatic explosive detection systems. In: Proceedings. Institute of Electrical and Electronics Engineers 29th Annual International Carnahan Conference on Security Technology, pp. 175–179 (1995). doi:10.1109/CCST.1995.524908
71. Strecker, H.: Automatic detection of explosives in airline baggage using elastic X-ray scatter. Medicamundi **42**, 30–33 (1998)
72. Wales, A., Halbherr, T., Schwaninger, A.: Using speed measures to predict performance in X-ray luggage screening tasks. In: 43rd Annual 2009 International Carnahan Conference on Security Technology, pp. 212–215 (2009)
73. Abidi, B.R., Zheng, Y., Gribok, A.V., Abidi, M.A.: Improving weapon detection in single energy X-ray images through pseudocoloring. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. **36**(6), 784–796 (2006)
74. Chan, J., Evans, P., Wang, X.: Enhanced color coding scheme for kinetic depth effect X-ray (KDEX) imaging. In: 2010 IEEE International Carnahan Conference on Security Technology (ICCST), pp. 155–160 (2010)
75. Singh, M., Singh, S.: Optimizing image enhancement for screening luggage at airports. In: Proceedings of the 2005 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety, CIHSPS 2005, pp. 131–136 (2005)
76. Oertel, C., Bock, P.: Identification of objects-of-interest in X-Ray images. In: 35th IEEE Applied Imagery and Pattern Recognition Workshop, AIPR 2006, p. 17 (2006)
77. Liu, D., Wang, Z.: A united classification system of X-ray image based on fuzzy rule and neural networks. In: 3rd International Conference on Intelligent System and Knowledge Engineering, ISKE 2008, vol. 1, pp. 717–722 (2008)
78. Nercessian, S., Panetta, K., Agaian, S.: Automatic detection of potential threat objects in X-ray luggage scan images. In: 2008 IEEE Conference on Technologies for Homeland Security, pp. 504–509 (2008). doi:10.1109/THS.2008.4534504
79. von Bastian, C., Schwaninger, A., Michel, S.: Do Multi-View X-ray Systems Improve X-Ray Image Interpretation in Airport Security Screening?, vol. 52. GRIN Verlag, Munich (2010)
80. Singh, S., Singh, M.: Explosives detection systems (EDS) for aviation security. Signal Process. **83**(1), 31–55 (2003)
81. Wells, K., Bradley, D.: A review of X-ray explosives detection techniques for checked baggage. Appl. Radiat. Isot. **70**, 1729–1746 (2012)
82. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)
83. Mery, D.: BALU: a toolbox Matlab for computer vision, pattern recognition and image processing. http://dmery.ing.puc.cl/index.php/balu
84. Lowe, D.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
85. Vedaldi, A., Fulkerson, B.: VLFeat: an open and portable library of computer vision algorithms. In: Proceedings of the International Conference on Multimedia, pp. 1469–1472. ACM (2010)
86. Dollár, P.: Piotr's Image and Video Matlab Toolbox (PMT). http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html

87. Mery, D., Riffo, V., Zscherpel, U., Mondragon, G., Lillo, I., Zuccar, I., Lobel, H., Carrasco, M.: GDXray: the GRIMA database of X-ray images. http://dmery.ing.puc.cl/index.php/material/gdxray/ (2015)
88. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: 9th European Conference on Computer Vision (ECCV2006). Graz Austria (2006)
89. Sivic, J., Zisserman, A.: Efficient visual search of videos cast as text retrieval. IEEE Trans. Pattern Anal. Mach. Intell. **31**(4), 591–605 (2009)
90. Mery, D.: Crossing line profile: a new approach to detecting defects in aluminium castings. In: Proceedings of the Scandinavian Conference on Image Analysis (SCIA 2003). Lecture Notes in Computer Science, vol. 2749, pp. 725–732 (2003)
91. Gobi, A.F.: Towards generalized benthic species recognition and quantification using computer vision. In: 4th Pacific-Rim Symposium on Image and Video Technology (PSIVT2010), Singapore, 14–17 November 2010, pp. 94–100 (2010)
92. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide-baseline stereo from maximally stable extremal regions. Image Vis. Comput. **22**(10), 761–767 (2004)
93. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 971–987 (2002)
94. Mery, D., Lillo, I., Riffo, V., Soto, A., Cipriano, A., Aguilera, J.: Automated fish bone detection using X-ray testing. J. Food Eng. **2011**(105), 485–492 (2011)
95. Haff, R., Toyofuku, N.: X-ray detection of defects and contaminants in the food industry. Sens. Instrum. Food Qual. Saf. **2**(4), 262–273 (2008). doi:10.1007/s11694-008-9059-8
96. Kwon, J., Lee, J., Kim, W.: Real-time detection of foreign objects using X-ray imaging for dry food manufacturing line. In: Proceedings of IEEE International Symposium on Consumer Electronics, ISCE 2008, pp. 1–4 (2008)
97. Jiang, J., Chang, H., Wu, K., Ouyang, C., Yang, M., Yang, E., Chen, T., Lin, T.: An adaptive image segmentation algorithm for X-ray quarantine inspection of selected fruits. Comput. Electron. Agric. **60**, 190–200 (2008)
98. Ogawa, Y., Kondo, N., Shibusawa, S.: Inside quality evaluation of fruit by X-ray image. In: Proceedings. 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM 2003, vol. 2, pp. 1360–1365 (2003)
99. Haff, R., Slaughter, D.: Real-time X-ray inspection of wheat for infestation by the granary weevil, Sitophilus granarius (l.). Trans. Am. Soc. Agric. Eng. **47**, 531–537 (2004)
100. Bej, G., Akuli, A., Pal, A., Dey, T., Chaudhuri, A., Alam, S., Khandai, R., Bhattacharyya, N.: X-ray imaging and general regression neural network (GRNN) for estimation of silk content in cocoons. In: Proceedings of the 2nd International Conference on Perception and Machine Intelligence, pp. 71–76. ACM (2015)
101. Karunakaran, C., Jayas, D., White, N.: Identification of wheat kernels damaged by the red flour beetle using X-ray images. Biosyst. Eng. **87**(3), 267–274 (2004)
102. Kelkar, S., Boushey, C.J., Okos, M.: A method to determine the density of foods using X-ray imaging. J. Food Eng. **159**, 36–41 (2015)
103. Mathanker, S., Weckler, P., Bowser, T., Wang, N., Maness, N.: AdaBoost classifiers for pecan defect classification. Comput. Electron. Agric. **77**(1), 60–68 (2011)
104. Neethirajan, S., Karunakaran, C., Symons, S., Jayas, D.: Classification of vitreousness in durum wheat using soft X-rays and transmitted light images. Comput. Electron. Agric. **53**(1), 71–78 (2006)
105. Nielsen, M.S., Christensen, L.B., Feidenhans, R.: Frozen and defrosted fruit revealed with X-ray dark-field radiography. Food Control **39**, 222–226 (2014)
106. Gonzalez, R., Woods, R.: Digital Image Processing, 3rd edn. Pearson, Prentice Hall, Upper Saddle River (2008)
107. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-8**(6), 679–698 (1986)
108. Duan, X., Cheng, J., Zhang, L., Xing, Y., Chen, Z., Zhao, Z.: X-ray cargo container inspection system with few-view projection imaging. Nucl. Instrum. Methods Phys. Res. A **598**, 439–444 (2009)

109. Frosio, I., Borghese, N., Lissandrello, F., Venturino, G., Rotondo, G.: Optimized acquisition geometry for X-ray inspection. In: 2011 IEEE Instrumentation and Measurement Technology Conference (I2MTC), pp. 1–6 (2011)
110. Kolkoori, S., Wrobel, N., Deresch, A., Redmer, B., Ewert, U.: Dual high-energy X-ray digital radiography for material discrimination in cargo containers. In: 11th European Conference on Non-Destructive Testing (ECNDT 2014), 6–10 October 2014, Prague, Czech Republic (2014)
111. Zhu, Z., Zhao, L., Lei, J.: 3D measurements in cargo inspection with a gamma-ray linear pushbroom stereo system. In: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (2005)
112. Zhu, Z., Hu, Y.C., Zhao, L.: Gamma/X-ray linear pushbroom stereo for 3D cargo inspection. Mach. Vis. Appl. **21**(4), 413–425 (2010)
113. Mahmood, K., Carmona, P.L., Shahbazmohamadi, S., Pla, F., Javidi, B.: Real time automated counterfeit integrated circuit detection using X-ray microscopy. J. Appl. Opt. (2015). To be published
114. Uehara, M., Yashiro, W., Momose, A.: Effectiveness of X-ray grating interferometry for non-destructive inspection of packaged devices. J. Appl. Phys. **114**(13), 134901 (2013)
115. Wang, Y., Wang, M., Zhang, Z.: Microfocus X-ray printed circuit board inspection system. Optik-International J. Light Electron Opt. **125**(17), 4929–4931 (2014)
116. Wu, J.H., Yan, X.Y., Wang, G.: High-resolution PCB board defect detection system based on non-destructive detection. Instrum. Tech. Sens. **6**, 028 (2013)

Appendix A
$\mathbb{GDX}$ray Details

In this appendix we show the details of each series of $\mathbb{GDX}$ray. The database consists of 19,407 X-ray images. The images are organized in a public database called $\mathbb{GDX}$ray that can be used free of charge,[1] for research and educational purposes only. The database includes five groups of X-ray images: castings, welds, baggages, natural objects, and settings. Each group has several series and each series has several X-ray images. The most of the series are annotated or labeled. In those cases, the coordinates of the bounding boxes of the objects of interest or the labels of the images are available in standard text files. The size of $\mathbb{GDX}$ray is 3.5 GB and it can be downloaded from our website.

The details of each series are summarized in following tables: Table A.1 for natural objects, Table A.2 for castings, Table A.3 for baggages, Table A.4 for welds and Table A.5 for setting X-ray images. See more about $\mathbb{GDX}$ray in Chap. 2.

[1] Available on http://dmery.ing.puc.cl/index.php/material/gdxray.

D. Mery, *Computer Vision for X-Ray Testing*, DOI 10.1007/978-3-319-20747-6

Table A.1 Description of group 'Nature' of 𝔾𝔻𝕏ray

Series	Images	kpixels	Description	Additional
N0001	13	5935.1	Apples	
N0002	200	10.0	Cropped images of 100 × 100 pixels for fish bone detection	Labels
N0003	7,697	0.1	Cropped images of 10 × 10 pixels for fish bone detection	Labels
N0004	20	143.3	Static noisy images of a wood piece	
N0005	9	4076.7	Apples	Annotations for apples
N0006	27	5935.1	Cherries	Annotations for cherries
N0007	8	5935.1	Cherries	Annotations for cherries
N0008	3	5935.1	Kiwis	Annotations for cherries
N0009	39	585.0	Wood pieces	
N0010	99	83.6	Wood pieces	
N0011	163	5935.1	Salmon filets	
N0012	6	5935.1	Selected 6 images of N0011	Annotation for fish bones. See N0013
N0013	6	5935.1	Binary ideal segmentation of N0012	Original images in N0012

Table A.2 Description of group 'Castings' of 𝔾𝔻𝕏ray

Series	Images	kpixels	Description	Additional
C0001	72	439.3	Wheel: Rotation each 5 degrees	Annotations for defects, calibration
C0002	90	44.5	Crops of C0001 with and without defects	Annotations for defects
C0003	37	439.3	Wheel with slow rotation	
C0004	37	439.3	Wheel with medium rotation. No defects	
C0005	37	439.3	Wheel with fast rotation. No defects	
C0006	37	439.3	Wheel with medium rotation. No defects	
C0007	37	439.3	Wheel with medium rotation	
C0008	37	439.3	Wheel with medium rotation. Large defect	Annotations for defects
C0009	37	439.3	Wheel with medium rotation	
C0010	37	439.3	Wheel with medium rotation. Large defect	Annotations for defects

(continued)

Table A.2 (continued)

Series	Images	kpixels	Description	Additional
C0011	37	439.3	Wheel with medium rotation	
C0012	37	439.3	Wheel with medium rotation. No defect	
C0013	37	439.3	Wheel with medium rotation. No defect	
C0014	37	439.3	Wheel with medium rotation. Defect at axis	
C0015	37	439.3	Wheel with medium rotation. Defect at axis	Annotations for defects
C0016	37	439.3	Wheel with medium rotation. Defect at edge	
C0017	37	439.3	Wheel with medium rotation. Defect at edge	
C0018	37	439.3	Wheel with no defect	
C0019	37	439.3	Wheel with hidden defect	Annotations for defects
C0020	37	439.3	Wheel with possible defect at lateral side	
C0021	37	439.3	Wheel with many small drilled defects	Annotations for defects
C0022	37	439.3	Wheel with letters at lateral side	
C0023	37	439.3	Wheel with no defect	
C0024	37	439.3	Wheel with defects at the lateral side	Annotations for defects
C0025	37	439.3	Wheel with defects like a regular structure	
C0026	37	439.3	Wheel with large hidden defects	Annotations for defects
C0027	37	439.3	Wheel with no defect	
C0028	37	439.3	Wheel with no defect	
C0029	37	439.3	Wheel with no defect (lateral side)	Annotations for regular structure
C0030	37	439.3	Wheel with defect in its axis	Annotations for defects
C0031	37	439.3	Wheel with several defects	Annotations for defects
C0032	37	439.3	Wheel with several defects	Annotations for defects
C0033	37	439.3	Wheel with several defects	Annotations for defects
C0034	37	439.3	Wheel with defects. No motion	Annotations for defects
C0035	37	439.3	Wheel with large defect	Annotations for defects
C0036	37	439.3	Wheel with letters at lateral side	Annotations for letters

(continued)

Table A.2 (continued)

Series	Images	Kpixels	Description	Additional
C0037	37	439.3	Wheel with large defect on an edge	Annotations for defects
C0038	37	439.3	Wheel with hidden defect	Annotations for defects
C0039	37	439.3	Wheel with hidden defect	Annotations for defects
C0040	37	439.3	Wheel with hidden defect	Annotations for defects
C0041	37	439.3	Wheel with defects. No motion	Annotations for defects
C0042	37	439.3	Wheel with several defects in motion	Annotations for defects
C0043	37	439.3	Wheel with large defect at axis	Annotations for defects
C0044	66	65.5	Wheel with small drilled defects	
C0045	66	65.5	Wheel with small drilled defects	Annotations for defects
C0046	65	65.5	Wheel with small drilled defects	
C0047	72	65.5	Wheel with small drilled defects	Annotations for defects
C0048	71	65.5	Wheel with small drilled defects	
C0049	63	65.5	Wheel with small drilled defects	
C0050	54	65.5	Wheel with small drilled defects	
C0051	77	65.5	Wheel with small drilled defects	Annotations for defects
C0052	17	440.8	Knuckle with small defects in motion	
C0053	31	440.8	Knuckle with small defects in motion	
C0054	31	440.8	Knuckle with low contrast defects in motion	Annotations for defects
C0055	28	440.8	Sink strainer	Annotations for holes
C0056	10	440.8	Sink strainer high speed	
C0057	31	440.8	Knuckle with low contrast defects in motion	Annotations for defects
C0058	56	440.8	Knuckle with small defects in motion	
C0059	43	440.8	Knuckle with small defects in motion	
C0060	14	440.8	Knuckle with small defects in motion	Annotations for defects

(continued)

Table A.2 (continued)

Series	Images	kpixels	Description	Additional
C0061	31	440.8	Knuckle with small defects in motion	
C0062	10	440.8	Knuckle with small defects in motion	Annotations for defects
C0063	11	440.8	Knuckle with small defects in motion	
C0064	56	440.8	Knuckle with small defects in motion	
C0065	10	440.8	Knuckle with small defects in motion	Annotations for defects
C0066	52	440.8	Knuckle with small defects in motion	
C0067	83	440.8	Knuckle with small defects in motion	

Table A.3 Description of group ‘Baggages’ of 𝔾𝔻𝕏ray

Series	Images	kpixels	Description	Additional
B0001	14	5935.1	Pen case with several objects	Annotations for razor blades
B0002	9	1287.0	Pen case with several objects	Annotations for razor blades
B0003	10	1287.0	Pen case with several objects	Annotations for clips
B0004	9	722.5	Pen case with occluded razor blade	Annotations for razor blades
B0005	10	722.5	Pen case with several objects	Annotations for pins
B0006	10	722.5	Pen case with occluded razor blade	Annotations for razor blades
B0007	20	129.6	Razor blade for training purposes	
B0008	361	745.8	Rotation of a knife in 1°	
B0009	4	276.6	Backpack with handgun	Annotations for handguns
B0010	11	276.6	Backpack with handgun	Annotations for handguns
B0011	10	276.6	Backpack with handgun	Annotations for handguns
B0012	4	276.6	Backpack with handgun	Annotations for handguns
B0013	10	276.6	Backpack with handgun and knife	Annotations for knives
B0014	5	276.6	Backpack with handgun and camera	Annotations for handguns
B0015	5	276.6	Backpack with handgun	Annotations for handguns
B0016	4	276.6	Backpack with self-occluded handgun	Annotations for handguns
B0017	5	276.6	Backpack with occluded handgun	Annotations for handguns

(continued)

Table A.3 (continued)

Series	Images	kpixels	Description	Additional
B0018	4	276.6	Backpack with handgun and laptop	Annotations for handguns
B0019	6	276.6	Backpack with handgun and laptop	Annotations for handguns
B0020	4	276.6	Backpack with handgun	Annotations for handguns
B0021	4	276.6	Backpack with handgun	Annotations for handguns
B0022	6	276.6	Backpack with handgun	Annotations for handguns
B0023	6	276.6	Backpack with handgun	Annotations for handguns
B0024	5	276.6	Backpack with handgun	Annotations for handguns
B0025	4	276.6	Backpack with handgun	Annotations for handguns
B0026	5	276.6	Backpack with handgun	Annotations for handguns
B0027	5	276.6	Backpack with handgun	Annotations for handguns
B0028	5	276.6	Backpack with handgun and laptop	Annotations for handguns
B0029	7	276.6	Backpack with handgun and laptop	Annotations for handguns
B0030	7	276.6	Backpack with handgun	Annotations for handguns
B0031	4	276.6	Backpack with handgun and laptop	Annotations for handguns
B0032	4	276.6	Backpack with handgun	Annotations for handguns
B0033	5	276.6	Backpack with occluded handgun	Annotations for handguns
B0034	6	276.6	Backpack with handgun and laptop	Annotations for handguns
B0035	4	276.6	Backpack with handgun and laptop	Annotations for handguns
B0036	11	276.6	Backpack with self-occluded handgun	Annotations for handguns
B0037	11	276.6	Backpack with handgun	Annotations for handguns
B0038	11	276.6	Backpack with handgun	Annotations for handguns
B0039	9	276.6	Backpack with handgun	Annotations for handguns
B0040	12	276.6	Backpack with handgun	Annotations for handguns
B0041	10	276.6	Backpack with handgun	Annotations for handguns
B0042	19	276.6	Backpack with handgun and knives	Annotations for handguns
B0043	9	276.6	Backpack with handgun and camera	Annotations for handguns
B0044	178	5935.1	Backpack with handgun	Calibration parameters
B0045	90	1287.0	Pen case in 90 positions	Annotations for razor blades
B0046	200	5844.0	Backpack with handgun	Annotations for handguns
B0047	200	5896.9	Backpack with shuriken	Annotations for shuriken

(continued)

Table A.3 (continued)

Series	Images	kpixels	Description	Additional
B0048	200	5412.0	Backpack with razor blade	Annotations for razor blade
B0049	200	759.5	Handguns for training purposes	
B0050	100	741.3	Shuriken for training purposes	
B0051	100	165.6	Razor blades for training purposes	
B0052	144	741.3	Shuriken with 8 points for training purposes	
B0053	144	741.3	Shuriken with 7 points for training purposes	
B0054	144	741.3	Shuriken with 6 points for training purposes	
B0055	800	16.9	200 4-image sequences of single objects	Labels
B0056	1200	18.1	200 6-image sequences of single objects	Labels
B0057	1600	18.0	200 8-image sequences of single objects	Labels
B0058	64	196.6	Crops of clips, springs, razor blades and others	Labels. See B0059
B0059	64	196.6	Binary ideal segmentation of images of B0058	Labels. Original images in B0058
B0060	2	5935.1	Images for dual-energy experiments	Annotations for shuriken
B0061	21	3656.8	Razor blade in a can	
B0062	22	3656.8	Razor blade in a wallet	
B0063	19	3656.8	Razor blade in a CD case	Annotations for razor blades
B0064	19	3656.8	Razor blade in a pen case	
B0065	21	3656.8	Razor blade in a pen case	Annotations for razor blades
B0066	22	3656.8	Razor blade in a pen case	
B0067	17	2856.1	Razor blade in a wallet	Annotations for razor blades
B0068	20	2856.1	Razor blade in a large wallet	
B0069	25	2856.1	Razor blade in a pen case	
B0070	21	2856.1	Razor blade in a pen case	Annotations for razor blades
B0071	22	2856.1	Razor blade in a pen case	
B0072	22	2856.1	Razor blade in a small pen case	
B0073	20	2856.1	Razor blade in a can	Annotations for razor blades
B0074	37	2856.1	Rotation of a door key in 10°	
B0075	576	5935.1	Knife in 576 positions	
B0076	576	1581.8	Knife in 576 positions	
B0077	576	1582.6	Knife in 576 positions	

Table A.4 Description of group 'Welds' of 𝔾𝔻𝕏ray

Series	Images	kpixels	Description	Additional
W0001	10	3323.8	Selection of 10 images of W0003	Annotations for defects. See W0002
W0002	10	3323.8	Binary ideal segmentation of images of W0001	
W0003	68	6693.8	Radiographs from a round robin test performed by BAM	Excel file with real-values

Table A.5 Description of group 'Settings' of 𝔾𝔻𝕏ray

Series	Images	kpixels	Description	Additional
S0001	18	5935.1	Checkerboard captured by flat panel	Calibration parameters
S0002	1	427.9	Regular grid captured by image intensifier	Coordinates of calibration points
S0003	36	440.8	Circular pattern in different positions	Manipulator coordinates, 3D coordinates
S0004	23	440.8	Circular pattern in different positions	Manipulator coordinates, 3D coordinates
S0005	27	440.8	Circular pattern in different positions	Manipulator coordinates, 3D coordinates
S0006	17	440.8	Circular pattern in different positions	Manipulator coordinates, 3D coordinates
S0007	29	440.8	Circular pattern in different positions	Coordinates of calibration points (2D and 3D)

Appendix B
𝕏vis Toolbox: Quick Reference

Every function is available on http://web.ing.puc.cl/~dmery/xvis/function_name.m.

In this appendix we show the details of the commands of 𝕏vis Toolbox. The toolbox consists of approximately 150 functions that are divided into 8 groups: input and output (see Table B.1), geometry (see Table B.2), image processing (see Table B.3), feature extraction (see Table B.4), feature analysis and feature selection (see Table B.5), classification (see Table B.6), performance evaluation (see Table B.7), and simulation (see Table B.8). 𝕏vis Toolboxcan be downloaded from our webpage.[2]

For the installation of 𝕏vis Toolbox, please follow these steps:

1. Download zip file from our webpage (see footnote 2).
2. Unzip zip file to the desired location of the toolbox.
3. Start Matlab.
4. Change into the folder used in step 2
5. Execute `Xsetup`

𝕏vis Toolbox does not need any compilation. The installation just add the commands to search path of Matlab.

[2] http://dmery.ing.puc.cl/index.php/book/xvis/.

D. Mery, *Computer Vision for X-Ray Testing*, DOI 10.1007/978-3-319-20747-6

Table B.1 Input and output

Function	Description
Xannotate	GUI for annotation
Xbinview	Display of a binary image superimposed onto a gray scale image
Xcolorimg	Display 10 colored representations of a gray scale image
Xdecisionline	Display a 2D feature space and decision line
Xdrawellipse	Draw an ellipse
Xdualenergy	Display a dual-energy image
Xdyncolor	Dynamic pseudocolor representations of a gray scale image
Xgdxannotate	Interactive environment for annotation of GDXray images
Xgdxbrowse	GUI for browsing GDXray series and images
Xgdxdir	Name of directory of a group and series of GDXray
Xgdxrandom	Display random images of GDXray
Xgdxstats	Count number of X-ray images on GDXray database
Xloaddata	Load data from GDXray
Xloadimg	Load an image from GDXray
Xplotepipolarline	Plot epipolar line
Xplotfeatures	Plot features and feature space
Xplotroc	Plot ROC curve and fit to an exponential curve
Xprintfeatures	Display feature values
Xpseudocolor	Color a gray scale image
XshowGT	Display ground truth of a GDXray series
Xshowconfusion	Display confusion matrix
Xshowseries	Display X-ray images of a GDXray series
Xsincolormap	Sin color map for pseudo coloring

Table B.2 Geometry

Function	Description
Xfundamental	Fundamental matrix
Xh2nh	Conversion homogeneous to non-homogeneous coordinates
Xhyperproj	Hyperbolic projection model
Xmatrixp	Perspective proyection matrix 3D->2D
Xmatrixr2	2D rotation matrix
Xmatrixr3	3D rotation matrix
Xreco3d2	3D reconstruction for two views
Xreco3dn	3D reconstruction for n>1 views
Xreproj3	Reprojection of third view using trifocal tensors
Xtrgui	GUI for tracking algorithm
Xtrifocal	Trifocal tensors

Table B.3 Image processing

Function	Description
Xforceuni	Enhancement ensuring an uniform histogram
Xgradlog	Edge detection using LoG and gradient
Ximaverage	Linear average filtering
Ximgaussian	Gaussian filtering
Ximgrad	Image gradient
Ximmedian	Median filtering
Xlinimg	Linear enhancement
Xregiongrowing	Region segmentation using growing region algorithm
Xresminio	Image restoration using MINIO criterium
Xsaliency	Center surround saliency image
Xsegbimodal	Segmentation in bimodal images
Xseglogfeat	Segmentation of regions using LoG edge detection
Xsegmser	Segmentation using MSER algorithm
Xsegsliwin	Segmentation using sliding windows
Xshading	Shading correction

Table B.4 Feature extraction

Function	Description
Xbasicgeo	Standard geometric features of a binary image
Xbasicint	Basic intensity features
Xbsif	Binarized statistical image features (BSIF)
Xcentroid	Centroid of a region
Xclp	Crossing line profiles (CLP)
Xcontrast	Contrast features
Xdct	2D DCT features
Xfitellipse	Fit ellipse for the boundary of a binary image
Xflusser	Flusser moments
Xfourier	2D Fourier features
Xfourierdes	Fourier descriptors
Xfxall	Pixel values of all pixels
Xfxbuild	Build structure for feature extraction with default values
Xfxgeo	Geometric features
Xfxgui	GUI for feature extraction
Xfxint	Intensity features
Xfxrandompatches	Feature extraction of random patches of an image
Xfxseqpatches	Feature extraction of random patches of a set of images

(continued)

Table B.4 (continued)

Function	Description
Xfxtractor	Feature extraction from a set of images
Xgabor	Gabor features
Xgupta	Gupta moments
Xharalick	Statistical texture features
Xhog	Histogram of oriented gradients
Xhugeo	Hu moments using geometric information
Xhuint	Hu moments using intensity and geometric information
Xlbp	Local binary patterns (LBP)

Table B.5 Feature analysis and feature selection

Function	Description
Xfbb	Feature selection using branch and bound
Xfclean	Feature selection cleaning
Xfexsearch	Feature selection using exhaustive search
XfmRMR	Feature selection using mRMR criterium
Xfnorm	Normalization of features
Xfosmod	Feature selection using FOS-MOD algorithm
Xfrank	Feature selection based on command rank features
Xfsall	Select all features
Xfsbuild	Build structure for feature selection with default values
Xfscore	Performance's score of selected features
Xfsp100	Specificity at sensibility = 100 %
Xjfisher	Fisher objective function
Xlsef	Feature selection using LSE-forward algorithm
Xnofeatures	Elimination of features with specific name
Xnorotation	Elimination of non-rotation invariant features
Xnoscale	Elimination of non-scale invariant features
Xnotranslation	Elimination of non-translation invariant features
Xpca	Principal component analysis
Xplsr	Feature transformation using partial least squares regression
Xsfs	Sequential forward selection
Xuninorm	Normalization of rows of features

Table B.6 Classification

Function	Description
Xadaboost	AdaBoost.M2 classifier
Xbayes	Bayes classifier
Xboosting	Boosting classifier
Xclassify	Classification using Xvis classifier(s)
Xclbuild	Build structure for classifiers with default values
Xclgui	GUI for classification selection
Xclsearch	Feature and classifier selection tool
Xdmin	Classifier using Euclidean minimal distance
Xglm	Neural network using a generalized linear model
Xknn	KNN (k-nearest neighbors) classifier
Xlda	LDA (linear discriminant analysis) classifier
Xmaha	Classifier using Mahalanobis minimal distance
Xmlp	Multi layer perceptron
Xparzen2	Mutual information using Parzen windows for two variables
Xpnn	Probabilistic neural network
Xqda	QDA (quadratic discriminant analysis) classifier
Xsparsecl	Classifier using sparse representations
Xsvm	Support vector machine (for two classes)
Xsvmplus	Support vector machine (for two and more classes)
Xtree	Classifier using a tree algorithm

Table B.7 Performance evaluation

Function	Description
Xaccuracy	Accuracy by comparing ideal with real classification
Xconfusion	Confusion matrix
Xcrossval	Evaluation of a classifier using cross-validation
Xdetectionstats	Evaluation of a segmentation algorithm on a set of images
Xgaussgen	Simulation of data with Gaussian distributions
Xholdout	Evaluation of a classifier using holdout
Xleaveoneout	Evaluation of a classifier using leave-one-out
Xnostratify	Data sampling without stratification
Xpascalstats	Pascal statistics
Xstratify	Data stratification

Table B.8 Simulation

Function	Description
Xobjvoxels	Generate voxels of an object
Xsimdefect	Simulation of defects using ellipsoids
Xsimmask	Mask superimposition
Xsimsphere	Simulation of defects using spheres
Xsimvoxels	Simulation of an X-ray image using voxels (or STL files)

Index

D. Mery, *Computer Vision for X-Ray Testing*, DOI 10.1007/978-3-319-20747-6