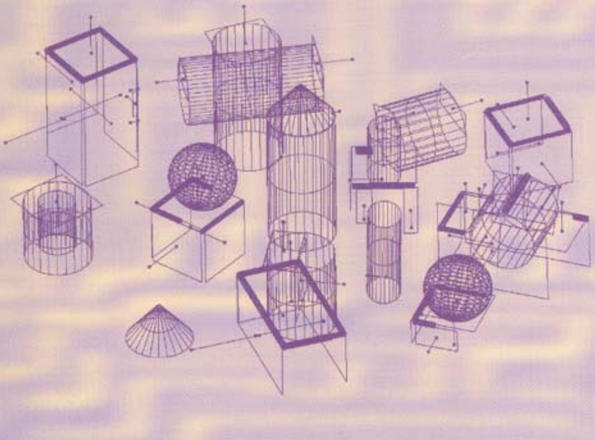
Sung Joon Ahn

LNCS 3151

Least Squares Orthogonal Distance Fitting of Curves and Surfaces in Space



Springer

This page intentionally left blank

Sung Joon Ahn

Least Squares Orthogonal Distance Fitting of Curves and Surfaces in Space

Springer

eBook ISBN: 3-540-28627-6
Print ISBN: 3-540-23966-9

Foreword

Due to the continuing progress of sensor technology, the availability of 3-D cameras is already foreseeable. These cameras are capable of generating a large set of measurement points within a very short time. There is a variety of 3-D camera applications in the fields of robotics, rapid product development and digital factories. In order to not only visualize the point cloud but also to recognize 3-D object models from the point cloud and then further process them in CAD systems, efficient and stable algorithms for 3-D information processing are required. For the automatic segmentation and recognition of such geometric primitives as plane, sphere, cylinder, cone and torus in a 3-D point cloud, efficient software has recently been developed at the Fraunhofer IPA by Sung Joon Ahn. This book describes in detail the complete set of 'best-fit' algorithms for general curves and surfaces in space which are employed in the Fraunhofer software.

Best-fit algorithms estimate curve/surface parameters by minimizing the shortest distances between the curve/surface and the measurement points. In many tasks of industrial image processing and optical 3-D measuring techniques, the fitting of curves and surfaces to a set of points plays an important role. Not only within the range of coordinate metrology and quality control but also for applications of automatic real object reconstruction, the fast and accurate fitting of geometric primitives will be of great benefit. By employing the best-fit algorithms in the Fraunhofer software, it is possible to reconstruct the geometric primitives in a way which is both CAD- and construction-friendly. The geometric primitives are obtained with all the necessary model parameters defining the form and orientation of the geometric primitives in space.

In principle, the software developed at the Fraunhofer IPA can be used everywhere where geometric primitives need to be quickly and automatically recognized from a given point cloud. Thus robotics represents a broad area of application of the Fraunhofer software, including the navigation of autonomous vehicles or the grasping of workpieces. A further application lies in factory digitization; this is concerned with the construction of a virtual 3-D production environment from an existing factory building which can be used, for example, as the starting point for simulating or planning the factory. Here, immense measurement data is produced and it is extremely time-consuming and therefore cost-intensive to process them manually. The automatic recognition of geometric primitives enables efficient factory digitization because a factory building can be composed by extensively using such simple geometric elements as piping (torus, cylinder) and planes.

The Fraunhofer software is able to segment a 3-D point cloud automatically. Without further user interaction, the suitable geometric primitives are fitted to the point cloud and the associating subsets of the point cloud are extracted at the same time. In order to accomplish this segmentation, no pre-treatment of the point cloud is required by hand. Furthermore, the point cloud does not have to fulfil special conditions. Without any point order or point arrangement, it is possible to process a large point cloud. The real objects must thereby not be completely covered by the measurement points. Even the incomplete collection of object points, e.g. due to shadings caused by the limited accessibility of the measuring devices to the object surface – which occurs frequently with optical sensors – does not cause any difficulties. Also, if a point cloud has been contaminated by severe measurement errors, it is still possible to obtain very good results. With these characteristics, the Fraunhofer software is applicable to a wide range of engineering areas where fast and accurate processing of 3-D point clouds is carried out.

Sung Joon Ahn has been working as a research scientist for the Fraunhofer IPA in Stuttgart, Germany, since April 1990. For his doctoral thesis, Dr. Ahn was awarded his research doctorate with the highest academic distinction 'summa cum laude' from the University of Stuttgart on 15th January 2004. This book is a replica of his doctoral thesis. We look forward to continuing our cooperation with Dr. Ahn after his return to Korea as a professor of the Sungkyunkwan University, especially in the fields of computer vision, intelligent robots, digital factories and reverse engineering where we particularly need his professional knowledge.

September 2004

Univ.-Prof. Dr.-Ing. Engelbert Westkämper
Institute director
Fraunhofer IPA

Preface

Least-squares is a term which implies that the model parameters are determined by *minimizing the square sum* of some predefined error measures. If a data set is a set of measurement points in 2-D/3-D space, the most natural error measure is the shortest distance between the model feature and the measurement point. This is because the measurement point is a probable observation of the *nearest* object point to the measurement point. The least-squares model fitting with this error measure is called *orthogonal distance fitting* (ODF). The square sum of the error distances can be described in two ways: the distance-based cost function and the coordinate-based cost function. The numerical methods minimizing these two cost functions are respectively called the *distance-based algorithm* and the *coordinate-based algorithm.*

The ODF task has two sub-problems: not only do the model parameters have to be found but also the shortest distance points on a model feature from each given point. In general, the shortest distance point is by nature nonlinear to the model parameters. Thus, the ODF problem is inherently a *nonlinear minimization problem* solved through iteration. There are two minimization strategies for solving the ODF problems through iteration. The *total method* simultaneously finds both the model parameters and the shortest distance points, whilst the *variable-separation method* finds them alternately using a nested iteration scheme. Four combinations of the two numerical methods and the two minimization strategies can be applied. The combination of the distance-based algorithm with the total method results in an underdetermined linear equation system for iteration and is thus impractical for solving ODF problems. As a result, three algorithmic approaches for solving ODF problems are realistic:

- Algorithm I: combination of the coordinate-based algorithm with the total method
- Algorithm II: combination of the distance-based algorithm with the variable-separation method
- Algorithm III: combination of the coordinate-based algorithm with the variable-separation method.

This book presents the ODF Algorithms I–III for implicit and parametric curves/surfaces in 2-D/3-D space possessing the following algorithmic features:

- Estimation of the model parameters minimizing the square sum of the shortest distances between a model feature and the given points in 2-D/3-D space
- General application to parametric and implicit curves/surfaces

- Estimation of the model parameters in terms of form, position, and rotation
- General implementation of the distance-based and coordinate-based algorithms
- General implementation of the total and variable-separation methods
- Solving ODF problems with additional constraints
- Provision of parameter-testing possibilities
- Robust and fast convergence
- Low computing costs and memory space usage
- Low implementation costs for a new model feature
- Automatic determination of the initial parameter values for iteration.

Chapter 1 first reviews the mathematical descriptions of curves and surfaces in space. The ordinary least-squares fitting algorithm is compared with the ODF algorithm in general and the current ODF algorithms are reviewed in detail. Chapter 2 begins with the ODF of lines/planes which can be solved in closed form. The two numerical methods, the *distance-based algorithm* and the *coordinate-based algorithm,* are then presented in a very general manner without any assumption of the type of model feature to be fitted at this stage. Chapter 3 deals with Algorithms II and III for *implicit* curves/surfaces. To find the shortest distance point on an implicit feature from a given point, the generalized Newton method and the method of Lagrangian multipliers are described. Chapter 4 deals with Algorithms I–III for *parametric* curves/surfaces. To locate the shortest distance point on a parametric feature from a given point, the Newton method and the Levenberg-Marquardt algorithm are described. Chapter 5 gives two application examples of the ODF algorithms for *object reconstruction.* A semi-automatic procedure for segmentation, outlier elimination and model fitting in point clouds is proposed and examined. Chapter 6 summarizes the contents of this book. Appendices A–C provide practical and helpful information for applying the general ODF algorithms to fit specific model features.

This book could not have been written without support from many individuals. First and foremost, I am indebted to my wife Kyung Mi and daughters Eun Bi and Eun Sol for their patience. I am deeply grateful for their love and support. I would like to thank Prof. Dr.-Ing. Engelbert Westkämper for the advice and financial support he has given to my research work at the Fraunhofer IPA. I would also like to thank Prof. Dr.-Ing. Hans-Jürgen Warnecke, the former president of the Fraunhofer Society, who met me in Korea and motivated me to work here. I owe special thanks to Prof. Dr.-Ing. Rolf Dieter Schraft and Prof. Dr. sc. nat. Wolfgang Osten who helped me to complete my research works. I thank Mrs. Schließer for making many improvements to the manuscript. I also thank the many past and present colleagues for their constructive discussions and practical support. In particular, sincere thanks go to Dr.-Ing. Kai-Udo Modrich and Dr.-Ing. Wolfgang Rauh, Heads of the Department of Technical Information Processing at the Fraunhofer IPA, who supported my work.

September 2004 Sung Joon Ahn

Table of Contents

List of Figures

List of Tables

This page intentionally left blank

List of Symbols

Acronyms

CAD : Computer Aided Design.
CAGD : Computer Aided Geometric Design.
CAM : Computer Aided Manufacturing.
CCD : Charge Coupled Device.
CLA : Coded Light Approach.
CMM : Coordinate Measuring Machine.
CT: Computer Tomography.
DIN : Deutsches Institut für Normung e.V., Germany.
ETH : Eidgenössische Technische Hochschule, Switzerland.
FhG: Fraunhofer-Gesellschaft, Germany.
ICP : Iterative Closest Point.
ISO : International Organization for Standardization.
LCD : Liquid Crystal Display.
LSM : Least-Squares Method.
NIST : National Institute of Standards and Technology, USA.
NPL : National Physical Laboratory, UK.
ODF : Orthogonal Distance Fitting.
PC : Personal Computer.
PTB : Physikalisch-Technische Bundesanstalt, Germany.
rms : root mean square.
SVD : Singular Value Decomposition.

General symbols

$\{\ldots\}$: set.
$\triangleq$: equality by definition.
$\forall$: for all or for any.
$\in$: element of a set.
$\emptyset$: empty set.
∞ : infinity.
$|\cdot|$: absolute value of a real number.
$\|\cdot\|$: norm of a vector/matrix.
$\leftarrow$: substitute.
$\leftrightarrow$: exchange.

Δ : difference, update.
$^{\mathrm{T}}$: transpose a vector/matrix.
$f(\cdot), F(\cdot), G(\cdot)$: functions.
$O(\cdot)$: computing complexity or storage space usage.
$\mathrm{sign}(\cdot)$: sign of a number.
$\mathrm{tr}(\cdot)$: trace of a matrix.
$\mathbb{R}$: real number field.
$\mathbb{R}^n$: n-dimensional real number space.
S : set of points.
xyz : moving (model) coordinate system.
XYZ : reference (machine) coordinate system.

Variables and parameters

l : number of form parameters.
m : number of data points.
n : dimensional degree of data points.
o : order of polynomials.
p : number of model parameters, $p = l + n + s$.
Also, index of power norm, e.g. L_p-norm.
q : number of constraints.
s : number of rotation parameters.
$a_1, \ldots, a_l$: form parameters of a model feature.
X_o, Y_o, Z_o : position (translation) parameters of a model feature.
ω, φ, κ : rotation parameters (Euler angles) of a model feature.
u, v : point location parameters for parametric features.
u_i', v_i' : location parameters of the minimum distance point $\mathbf{x}_i'$
on a parametric feature from the given point $\mathbf{x}_i$.
$\mathrm{Cor}(\hat{a}_i, \hat{a}_j)$: correlation coefficient of the two estimated parameters $\hat{a}_i$ and $\hat{a}_j$.
$\mathrm{Cov}(\hat{a}_i, \hat{a}_j)$: covariance between the two estimated parameters $\hat{a}_i$ and $\hat{a}_j$.
d_i : distance between the two points $\mathbf{X}_i$ and $\mathbf{X}_i'$, $d_i = \|\mathbf{X}_i - \mathbf{X}_i'\|$.
t : safety margin for segmentation of point cloud. Also, computing time.
w, w_c : (constraint) weighting values.
$w_1, \ldots, w_p$: singular values of a matrix extracted by the SVD,
$\mathbf{W} = [\mathrm{diag}(w_1, \ldots, w_p)]$.
α : step size factor for parameter update, e.g. $\mathbf{a}_{k+1} = \mathbf{a}_k + \alpha\Delta\mathbf{a}$.
λ : Adding factor with the Levenberg-Marquardt algorithm.
Also, Lagrangian multiplier.
σ_0^2 : square sum of (weighted) distance errors.

$$\sigma_0^2 \triangleq \sum_{i=1}^{m} d_i^2 \text{ or } \sigma_0^2 \triangleq \|\mathbf{Pd}\|^2 \text{ or } \sigma_0^2 \triangleq \|\mathbf{P}(\mathbf{X} - \mathbf{X}')\|^2 .$$

σ_{rms} : rms distance error, $\sigma_{\mathrm{rms}} = \sqrt{\sigma_0^2/m}$.

Vectors and vector functions

∇ : gradient operator,

$$\nabla \triangleq (\partial/\partial \mathbf{x})^{\mathrm{T}} = (\nabla_x, \nabla_y, \nabla_z)^{\mathrm{T}} = (\partial/\partial x, \partial/\partial y, \partial/\partial z)^{\mathrm{T}} .$$

$\mathbf{a}$: model parameter vector, $\mathbf{a}^{\mathrm{T}} \triangleq (\mathbf{a}_{\mathrm{g}}^{\mathrm{T}}, \mathbf{a}_{\mathrm{p}}^{\mathrm{T}}, \mathbf{a}_{\mathrm{r}}^{\mathrm{T}}) = (a_1, \ldots, a_p)$.

$\hat{\mathbf{a}}$: estimated model parameter vector.

$\mathbf{a}_{\mathrm{g}}$: form parameter vector, $\mathbf{a}_{\mathrm{g}}^{\mathrm{T}} \triangleq (a_1, \ldots, a_l)$.

$\mathbf{a}_{\mathrm{p}}$: position parameter vector, $\mathbf{a}_{\mathrm{p}}^{\mathrm{T}} \triangleq \mathbf{X}_{\mathrm{o}}^{\mathrm{T}} = (X_{\mathrm{o}}, Y_{\mathrm{o}}, Z_{\mathrm{o}})$.

$\mathbf{a}_{\mathrm{r}}$: rotation parameter vector, $\mathbf{a}_{\mathrm{r}}^{\mathrm{T}} \triangleq (\omega, \varphi, \kappa)$.

$\mathbf{b}$: special parameter vector for parametric features, $\mathbf{b}^{\mathrm{T}} \triangleq (\mathbf{a}^{\mathrm{T}}, \mathbf{u}_1^{\mathrm{T}}, \ldots, \mathbf{u}_m^{\mathrm{T}})$.
Also, algebraic parameter vector for algebraic features,
$\mathbf{b}^{\mathrm{T}} \triangleq (b_1, \ldots, b_p)$ with p = number of algebraic parameters.

$\mathbf{d}$: distance column vector, $\mathbf{d} \triangleq (d_1, \ldots, d_m)^{\mathrm{T}}$ with $d_i = \|\mathbf{X}_i - \mathbf{X}_i'\|$.

$\mathbf{f}(\cdot)$: function vector.

$\mathbf{n}$: normal vector.

$\mathbf{r}_{\mathrm{x}}, \mathbf{r}_{\mathrm{y}}, \mathbf{r}_{\mathrm{z}}$: direction cosine vector of the x-, y- and z-axis, respectively.

$\mathbf{u}$: location parameter vector for parametric features, $\mathbf{u} = (u, v)^{\mathrm{T}}$.

$\mathbf{u}_i'$: location parameter vector of the minimum distance point $\mathbf{x}_i'$
on a parametric feature from the given point $\mathbf{x}_i$, $\mathbf{u}_i' = (u_i', v_i')^{\mathrm{T}}$.

$\mathbf{x}_u, \mathbf{x}_v, \mathbf{x}_{uu}, \mathbf{x}_{vv}, \mathbf{x}_{uv}, \mathbf{x}_{vu}$: partial derivatives of $\mathbf{x}(\mathbf{a}_{\mathrm{g}}, \mathbf{u})$.
$\mathbf{x}_u = \partial \mathbf{x}/\partial u$, $\mathbf{x}_{uu} = \partial^2 \mathbf{x}/\partial u \partial u$, $\mathbf{x}_{uv} = \partial^2 \mathbf{x}/\partial u \partial v$, etc.

$\mathbf{x}, \mathbf{X}$: general coordinate vector for a point in frame xyz and XYZ, respectively.
$\mathbf{x} = (x, y, z)^{\mathrm{T}}$, $\mathbf{X} = (X, Y, Z)^{\mathrm{T}}$, $\mathbf{x} = \mathbf{R}(\mathbf{X} - \mathbf{X}_{\mathrm{o}})$.
Also, coordinate column vector, $\mathbf{X}^{\mathrm{T}} \triangleq (\mathbf{X}_1^{\mathrm{T}}, \ldots, \mathbf{X}_m^{\mathrm{T}})$.

$\mathbf{X}'$: coordinate column vector, $\mathbf{X}'^{\mathrm{T}} \triangleq (\mathbf{X}_1'^{\mathrm{T}}, \ldots, \mathbf{X}_m'^{\mathrm{T}})$.

$\mathbf{x}_i, \mathbf{X}_i$: coordinate vector of a given point in frame xyz and XYZ, respectively.
$\mathbf{x}_i = (x_i, y_i, z_i)^{\mathrm{T}}$, $\mathbf{X}_i = (X_i, Y_i, Z_i)^{\mathrm{T}}$.

$\mathbf{x}_i', \mathbf{X}_i'$: coordinate vector of the minimum distance point on a model
feature from the given point $\mathbf{x}_i$ and $\mathbf{X}_i$, respectively.
$\mathbf{x}_i' = (x_i', y_i', z_i')^{\mathrm{T}}$, $\mathbf{X}_i' = (X_i', Y_i', Z_i')^{\mathrm{T}}$.

$\mathbf{X}_{\mathrm{o}}$: origin of the model coordinate frame xyz referenced to
the machine coordinate frame XYZ, $\mathbf{X}_{\mathrm{o}} = (X_{\mathrm{o}}, Y_{\mathrm{o}}, Z_{\mathrm{o}})^{\mathrm{T}}$.

$\bar{\mathbf{X}}$: coordinate vector of the mass center of $\{\mathbf{X}_i\}_{i=1}^m$, $\bar{\mathbf{X}} = \frac{1}{m} \sum_{i=1}^m \mathbf{X}_i$.

Matrices

Cor($\hat{\mathbf{a}}$) : correlation matrix of the estimated model parameters $\hat{\mathbf{a}}$.

Cov($\hat{\mathbf{a}}$) : covariance matrix of the estimated model parameters $\hat{\mathbf{a}}$.

FHG matrix : Vector and matrix group of ∇f , $\mathbf{H} = \partial^2 f/\partial \mathbf{x} \partial \mathbf{x}$ and $\mathbf{G}$
derived from implicit feature equation $f(\mathbf{a}_{\mathrm{g}}, \mathbf{x}) = 0$.

XHG matrix : Vector and matrix group of $\partial \mathbf{x}/\partial \mathbf{u}$, $\mathbf{H} = \partial^2 \mathbf{x}/\partial \mathbf{u} \partial \mathbf{u}$ and $\mathbf{G}$
derived from parametric feature description $\mathbf{x}(\mathbf{a}_{\mathrm{g}}, \mathbf{u})$.

$\mathbf{G}$: special matrix defined in Sect. 3.2.2 for implicit features or
in Sect. 4.2.3 for parametric features.

$\mathbf{H}$: Hessian matrix. $\mathbf{H} = \partial^2 f/\partial \mathbf{x} \partial \mathbf{x}$ or $\mathbf{H} = \partial^2 \mathbf{x}/\partial \mathbf{u} \partial \mathbf{u}$.
Also, inertia tensor of centroidal moment of a point set.

$\mathbf{I}$: identity matrix.
$\mathbf{J}$: Jacobian matrix. $\mathbf{J} = \partial\mathbf{d}/\partial\mathbf{a}$ for the distance-based algorithm or
$\mathbf{J} = \partial\mathbf{X}'/\partial\mathbf{a}$ for the coordinate-based algorithm.
$\mathbf{J}_{d_i,\mathbf{a}}$: *i*-th row of the Jacobian matrix $\partial\mathbf{d}/\partial\mathbf{a}$, $\mathbf{J}_{d_i,\mathbf{a}} = \partial d_i/\partial\mathbf{a}$.
$\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}}$: *i*-th row group of the Jacobian matrix $\partial\mathbf{X}'/\partial\mathbf{a}$, $\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}} = \partial\mathbf{X}'_i/\partial\mathbf{a}$.
$\mathbf{M}$: central moment tensor of a point set.
$\mathbf{P}$: nonsingular symmetric matrix.
$\mathbf{P}^{\mathrm{T}}\mathbf{P}$: weighting or error covariance matrix (positive definite).
$\mathbf{R}$: rotation matrix. $\mathbf{R} = \mathbf{R}_{\omega,\varphi,\kappa} = (\mathbf{r}_{\mathrm{x}}\ \ \mathbf{r}_{\mathrm{y}}\ \ \mathbf{r}_{\mathrm{z}})^{\mathrm{T}}$, $\mathbf{R}^{-1} = \mathbf{R}^{\mathrm{T}}$.
U, V, $\mathbf{W}$: decomposed matrices of a matrix $\mathbf{J}$ by the SVD.
$\mathbf{J} = \mathbf{UWV}^{\mathrm{T}}$ with $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$ and
$\mathbf{W} = [\mathrm{diag}(w_1, \ldots, w_p)]$.
$\mathbf{W}_{\mathrm{c}}$: weighting matrix for parameter constraints.

1. Introduction

The *parametric model description* of real objects is a subject relevant to coordinate metrology, reverse engineering, and computer/machine vision, with which objects are represented by their model parameters estimated from the measurement data (*object reconstruction,* Fig. 1.1). The parametric model description has a wide range of application fields, e.g. manufacturing engineering, particle physics, and the entertainment industry. The alternative to parametric model description is *facet model description* that describes objects by using the polygonal mesh generated from a point cloud. Although the facet model description is adequate for object visualization and laser lithography, it does not provide applications with such object model information as shape, size, position, and rotation of the object. The facet model description has a limited accuracy in representing an object and demands considerable data memory space. For a faithful representation of objects using a polygonal mesh, a dense point cloud should be available as input data.

On the other hand, the parametric model description provides applications with highly compact and effective information about the object. The minimum number of the data points required for estimating model parameters is usually about the same as the number of model parameters. The data reduction ratio from a dense point cloud to the parametric model description is usually to the order of one thousandth. The parametric model description is preferred by most applications because of its usability and flexibility. For certain applications, such as in coordinate metrology and motion analysis, only the parametric model description is of real interest. If a real object can once be represented by its model parameters, the parametric model description can then be converted or exported to other formats including the facet model description (product data exchange [59]). The disadvantage of the parametric model description is the analytical and computational difficulties encountered in estimating the model parameters of real objects from their measurement data (point cloud).

This work aims at using the parametric model description for reconstructing real objects from their measurement points. The overall procedures for object reconstruction can be divided into two elements, 3-D measurement and 3-D data processing (Fig. 1.1). In recent years, noticeable progress has been made in optical 3-D measurement techniques through interdisciplinary cooperation between optoelectronics, image processing, and close-range photogrammetry. Among the diverse optical 3-D measuring devices, the laser radar is capable of measuring millions of

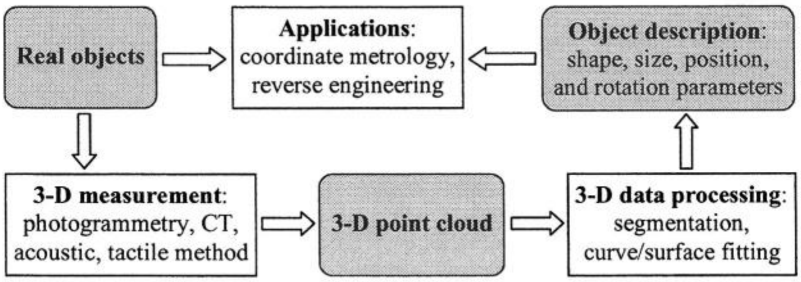

Real objects
Applications: coordinate metrology, reverse engineering
Object description: shape, size, position, and rotation parameters
3-D measurement: photogrammetry, CT, acoustic, tactile method
3-D point cloud
3-D data processing: segmentation, curve/surface fitting

Fig. 1.1. Parametric model recovery of real objects

dense 3-D points in a few seconds under various measuring conditions such as object surface material, measuring range, and ambient light [43], [56], [78]. An example of the application of laser radar is the digital documentation of an engineering plant that mainly consists of simple geometric elements such as plane, sphere, cylinder, cone, and torus. Demands for 3-D measurement techniques and the 3-D data volume to be processed are rapidly increasing with continuing scientific and technical development.

In contrast to the recent progress made in 3-D measurement techniques, that of software tools for 3-D data processing has virtually stagnated. The software tools that are currently available demand an intensive user action for documenting a complex real world scene. In practice, segmentation and outlier elimination in a point cloud is the major bottleneck of the overall procedure and is usually assisted by a human operator. It is accepted that no general software tool exists for the fully automated 3-D data processing of a real world scene [43], [56], [78]. Thus, considering the remarkable 3-D data throughput of advanced 3-D measuring devices, the development of a flexible and efficient software tool for 3-D data processing with a high degree of accuracy and automation is urgently required for applications of advanced optical 3-D measurement techniques in many engineering fields.

With the software tools for 3-D data processing, model fitting algorithms play a fundamental role in estimating the object model parameters from a point cloud. These are usually based on the *least-squares method* (LSM) [36] determining model parameters by minimizing the square sum of a predefined error-of-fit. Among the various error measures, the *shortest distance* (also known as geometric distance, orthogonal distance, or Euclidean distance in literature) between the model feature and the given point is of primary concern where dimensional data is being processed [1], [29], [61]. For a long time however, the analytical and computational difficulties encountered in computing and minimizing the geometric distance errors blocked the development of appropriate fitting (*orthogonal distance fitting,* ODF) algorithms [20], [21]. To bypass the difficulties associated with ODF, most data processing software tools use the approximating measures of the geometric error [22], [26], [32], [48], [51], [53], [67], [73], [76]. Nevertheless, when a highly accurate and reliable estimation of model parameters is essential to applications, the only possibility available is to use the geometric distance as the error measure to be minimized [29], despite the high computing costs and the difficulties involved in developing the ODF algorithms. Actually, the use of the geometric error measure has been prescribed by a recently ratified international standard for testing the data processing softwares for coordinate metrology [60].

Another performance factor of a software tool for 3-D data processing is the degree of automation of the object recognition procedure. Object recognition, including object detection and identification, is a highly relevant subject to reverse engineering and computer/machine vision. Object recognition is usually a sophisticated decision procedure analyzing all the available information about the objects, such as point cloud, object surface texture and color. Even human intelligence sometimes has difficulties in recognizing objects in a point cloud if no additional information is available. The availability of a fully automatic technique for object recognition is not expected in the near future. In the interim, a semi-automatic solution for object recognition in a point cloud would be welcomed which would minimize the user involvement in segmentation and outlier elimination.

Under the circumstances discussed above, this work deals with the development of ODF algorithms and their application for object recognition in a point cloud. Five ODF algorithms (two new for implicit features $F(\mathbf{a}, \mathbf{X}) = 0$, another two new for parametric features $\mathbf{X}(\mathbf{a}, \mathbf{u})$, and a known one for parametric features) are described below and compared in a well-organized and easily understandable manner. As any explicit feature $Z = F(\mathbf{a}, X, Y)$ can be simply converted to both implicit and parametric features, any analytic (i.e., explicit, implicit, or parametric) curve and surface in space can be fitted using these five ODF algorithms. As a practical application of the ODF algorithms, in this work an efficient object recognition procedure 'Click & Clear' is presented, which is an interactive procedure between segmentation, outlier detection, and model fitting utilizing the advantageous algorithmic features of the new ODF algorithms.

1.1 Curves and Surfaces in Space

1.1.1 Mathematical Description

Analytic curves/surfaces in 2-D/3-D space can be mathematically described in explicit, implicit, or parametric form as shown in Table 1.1. The explicit features are the subset of the implicit and parametric features. For example, an explicit surface

$$Z = F(\mathbf{a}, X, Y)$$

can be simply converted not only to an implicit surface

$$G(\mathbf{a}, X, Y, Z) \triangleq Z - F(\mathbf{a}, X, Y) = 0$$

but also to a parametric surface

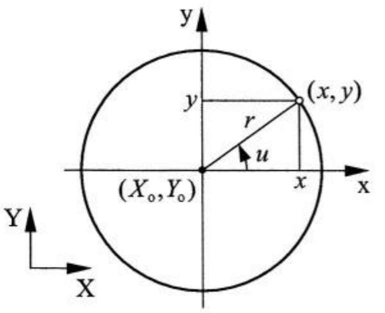
$$\begin{pmatrix} X(\mathbf{a}, u, v) \\ Y(\mathbf{a}, u, v) \\ Z(\mathbf{a}, u, v) \end{pmatrix} \triangleq \begin{pmatrix} u \\ v \\ F(\mathbf{a}, u, v) \end{pmatrix} .$$

Table 1.1. Curves and surfaces in a rectangular coordinate system

Description	Plane curve	Space curve	Surface
Explicit	$Y = F(\mathbf{a}, X)$	$\begin{pmatrix} Y \\ Z \end{pmatrix} = \begin{pmatrix} F(\mathbf{a}, X) \\ G(\mathbf{b}, X) \end{pmatrix}$	$Z = F(\mathbf{a}, X, Y)$
Implicit	$F(\mathbf{a}, X, Y) = 0$	$\begin{pmatrix} F(\mathbf{a}, X, Y, Z) \\ G(\mathbf{b}, X, Y, Z) \end{pmatrix} = \mathbf{0}$	$F(\mathbf{a}, X, Y, Z) = 0$
Parametric	$\begin{pmatrix} X(\mathbf{a}, u) \\ Y(\mathbf{a}, u) \end{pmatrix}$	$\begin{pmatrix} X(\mathbf{a}, u) \\ Y(\mathbf{a}, u) \\ Z(\mathbf{a}, u) \end{pmatrix}$	$\begin{pmatrix} X(\mathbf{a}, u, v) \\ Y(\mathbf{a}, u, v) \\ Z(\mathbf{a}, u, v) \end{pmatrix}$

In comparison with implicit or parametric features, the use of explicit features for applications is limited due to the fact that explicit features are axis-dependent and single-valued, e.g. a full circle/sphere cannot be described in explicit form (Fig. 1.2). With diverse applications handling dimensional models or objects, the usual descriptive form of a curve/surface is the implicit or parametric form [3], [30], [62]. Hence the main emphasis associated with the aim of this work, i.e. dimensional model fitting, is concentrated on the use of implicit and parametric features.

The distinctive model feature in Table 1.1 is the space curve (3-D curve). 3-D curves can be found in many disciplines of science and engineering such as computer vision, motion analysis, coordinate metrology, and CAGD. Although a 3-D curve can be created by intersecting two implicit surfaces, the question as to how to properly select, if they exist, the two implicit surfaces creating the target 3-D curve is not clear to answer. For example, there are a number of ways of creating a 3-D circle by intersecting two quadric surfaces. Even if the apparently best pair of implicit surfaces have been selected (e.g. a plane and a cylinder for creating a 3-D circle), the use of two surfaces implies an unnecessarily large set of model parameters (overparameterization). Consequently, with model fitting, the proper constraints have to be found and applied between model parameters, which remove the excessive freedom in parameter space. Therefore, the use of two implicit surfaces for describing a 3-D



Explicit: $y = \sqrt{r^2 - x^2}$ (upper semi-circle).

Implicit: $x^2 + y^2 - r^2 = 0$.

Parametric: $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r\cos u \\ r\sin u \end{pmatrix}, \quad 0 \le u < 2\pi$.

Rigid body motion: $\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} X_o \\ Y_o \end{pmatrix}$.

Fig. 1.2. Mathematical description of a 2-D circle

curve should be avoided with applications. In contrast, many useful 3-D curves such as a helix can only be efficiently described in parametric form. For these reasons the parametric form is preferable to the implicit form for describing a 3-D curve. A more detailed comparison of and discussion about the mathematical description of curves and surfaces in implicit and parametric form can be found in literature [3], [23], [30], [62], [72].

1.1.2 Rigid Body Motion of Model Features in Space

Generally, a model feature is arbitrarily located and oriented in space. With the applications of computer vision, motion analysis, robotics, and visualization, the rigid body motion of model features is of great interest. Thus, a model fitting algorithm would be highly advantageous for applications if model parameters $\mathbf{a}$ are grouped and estimated in terms of form $\mathbf{a}_\mathrm{g}$, position $\mathbf{a}_\mathrm{p}$, and rotation parameters $\mathbf{a}_\mathrm{r}$

$$\begin{aligned}\mathbf{a}^\mathrm{T} &\triangleq (\mathbf{a}_\mathrm{g}^\mathrm{T},\ \mathbf{a}_\mathrm{p}^\mathrm{T},\ \mathbf{a}_\mathrm{r}^\mathrm{T}) \\ &= (\underbrace{a_1,\ldots,a_l}_{l},\underbrace{X_\mathrm{o},Y_\mathrm{o},Z_\mathrm{o}}_{n},\underbrace{\omega,\varphi,\kappa}_{s}) \\ &= (\underbrace{a_1,\ldots,a_p}_{p})\ .\end{aligned}$$

The *form parameters* $\mathbf{a}_\mathrm{g}$ (e.g. radius r of a circle/sphere/cylinder, axis-lengths a, b, c of an ellipsoid) represent the shape and size of the standard (canonical) model feature defined in a model coordinate system xyz

$$f(\mathbf{a}_\mathrm{g},\mathbf{x}) = 0 \qquad \text{with} \qquad \mathbf{a}_\mathrm{g} = (a_1,\ldots,a_l)^\mathrm{T} \tag{1.1}$$

for implicit features (Fig. 3.1), and,

$$\mathbf{x}(\mathbf{a}_\mathrm{g},\mathbf{u}) \qquad \text{with} \qquad \mathbf{a}_\mathrm{g} = (a_1,\ldots,a_l)^\mathrm{T} \tag{1.2}$$

for parametric features (Fig. 4.1). The form parameters are invariant to the rigid body motion of model feature. The *position parameters* $\mathbf{a}_\mathrm{p}$ and *rotation parameters* $\mathbf{a}_\mathrm{r}$ determine the rigid body motion of model features in a machine coordinate system XYZ (see Sect. B.1.3)

$$\begin{aligned}&\mathbf{X} = \mathbf{R}^{-1}\mathbf{x} + \mathbf{X}_\mathrm{o} \qquad \text{or} \qquad \mathbf{x} = \mathbf{R}(\mathbf{X} - \mathbf{X}_\mathrm{o})\ , \qquad \text{where} \\ &\mathbf{R} = \mathbf{R}_{\omega,\varphi,\kappa} = (\mathbf{r}_\mathrm{x}\ \ \mathbf{r}_\mathrm{y}\ \ \mathbf{r}_\mathrm{z})^\mathrm{T}\ , \qquad \mathbf{R}^{-1} = \mathbf{R}^\mathrm{T}\ , \\ &\mathbf{a}_\mathrm{p} = \mathbf{X}_\mathrm{o} = (X_\mathrm{o}, Y_\mathrm{o}, Z_\mathrm{o})^\mathrm{T}\ , \qquad \text{and} \qquad \mathbf{a}_\mathrm{r} = (\omega,\varphi,\kappa)^\mathrm{T}\ .\end{aligned} \tag{1.3}$$

The form parameters are characterized as *intrinsic parameters* and the position/rotation parameters as *extrinsic parameters* of a model feature according to their context. A model feature is then identified and classified according to its intrinsic parameters, independent of the varying extrinsic parameters, i.e. even after a rigid body motion. The functional interpretation and treatment of the extrinsic parameters (1.3) are basically the same for all model features in 2-D/3-D space. By combining (1.1)–(1.3) a model feature can be defined in a machine coordinate system XYZ

$$\begin{aligned} F(\mathbf{a}_g, \mathbf{a}_p, \mathbf{a}_r, \mathbf{X}) &\triangleq f(\mathbf{a}_g, \mathbf{x}(\mathbf{a}_p, \mathbf{a}_r, \mathbf{X})) \\ &= f(\mathbf{a}_g, \mathbf{R}(\mathbf{X} - \mathbf{X}_o)) = 0 \end{aligned}$$

for implicit features, and,

$$\mathbf{X}(\mathbf{a}_g, \mathbf{a}_p, \mathbf{a}_r, \mathbf{u}) \triangleq \mathbf{R}^{-1}\mathbf{x}(\mathbf{a}_g, \mathbf{u}) + \mathbf{X}_o$$

for parametric features.

1.1.3 Model Hierarchy

By utilizing the parameter grouping in terms of form, position, and rotation parameters as proposed in Sect. 1.1.2, the hierarchical interface between model features can be constructed (Fig. 1.3). For example, a sphere, the generalization of a point in 3-D space, is a special instance of an ellipsoid that is again a special instance of a superellipsoid (Fig. 1.3d). The hierarchical model interface permits efficient and advantageous parameter initialization and model selection with model fitting.

The initial parameter values for the iterative model fitting can be obtained by fitting a low-level model feature to the same set of data points (see Sect. 3.3 and [9]). For example, the initial parameters for the superellipsoid fitting can be successively obtained from the sphere and the ellipsoid fitting. For the initial parameters for the sphere fitting, the mass center and the rms central distance of the set of points are used. By fitting a lower level model feature to the same set of data points, the parameter initialization preconditions a stable and robust convergence of the iterative model fitting because the model parameter values experience no abrupt changes with model transition. In other words, the two model features before and after model transition look the same (e.g. the case of transition from a sphere to an ellipsoid) or similar (e.g. the case of transition from a 3-D circle to a torus).

It is to be noted that ODF problems associated with the most primitive features in the model hierarchy can be solved in closed form (line/plane fitting) or their initial parameter values can be determined in closed form (circle/sphere fitting). This means that the overall procedure for iteratively fitting many model features can be carried out automatically by successively using the internally-supplied initial parameter values. The strategy for obtaining the initial parameter values for iterative model fitting is summarized below:

- Usage of the resultant parameter values from a low-level model fitting as the initial parameter values for a high-level model fitting (Fig. 1.3)
- Usage of the 3-D line parameter values and the rms centroidal error distance as the initial parameter values for the rod-cylinder fitting (Fig. 1.3a)
- Usage of the mass center and the rms central distance as the initial parameter values for the circle/sphere fitting (Fig. 1.3d)

- Usage of the plane parameter values and parameter values of the projected 2-D circle on the plane as the initial parameter values for the 3-D circle fitting (Fig. 1.3b and Fig. 1.3c)

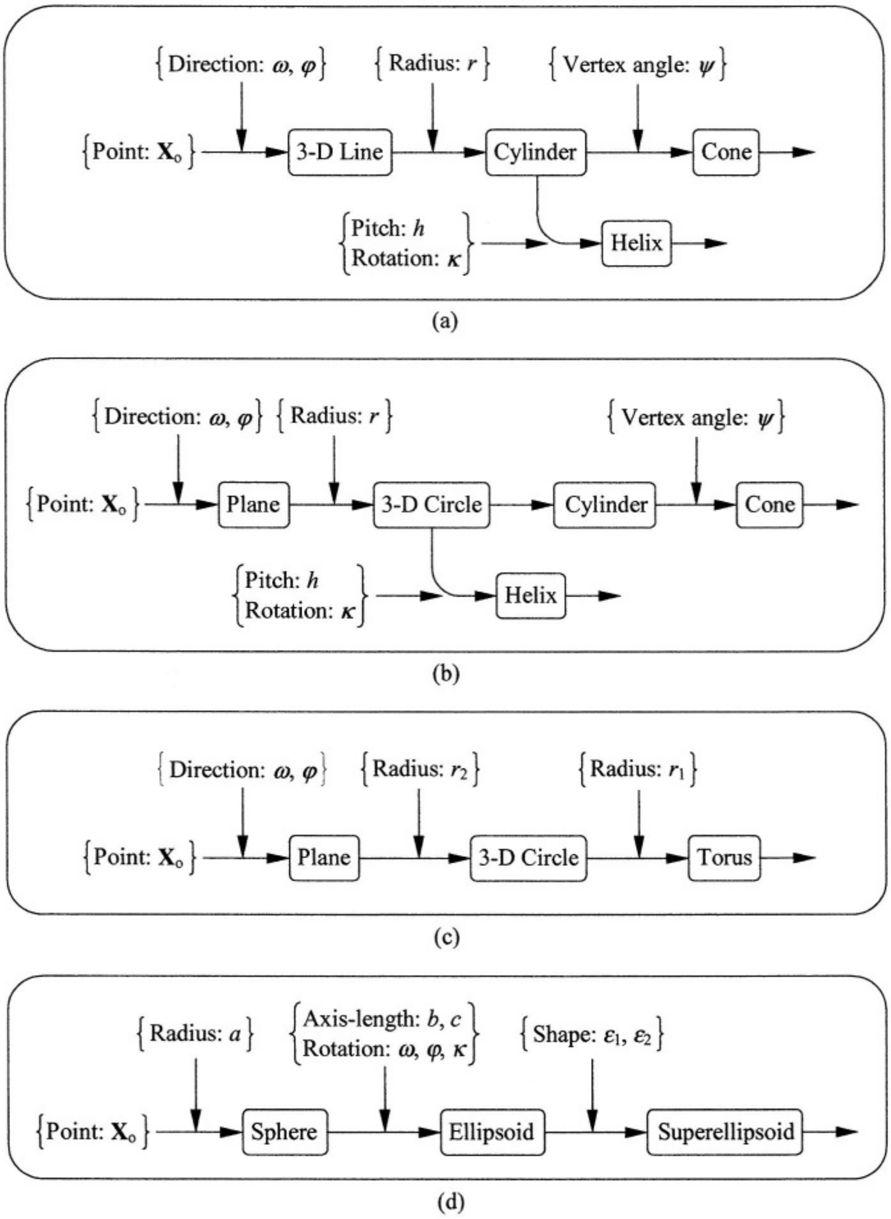


Fig. 1.3. Evolution of model features: (a) Cone and helix via rod-cylinder; (b) Cone via disc-cylinder and helix via 3-D circle; (c) Torus via 3-D circle; (d) Superellipsoid via ellipsoid

- If necessary, allow jumping or crossing in the model hierarchy (e.g., use the mass center and the rms central distance as the initial parameter values for the ellipsoid or the superellipsoid fitting (Fig. 1.3d)).

Selection of the proper model feature such as the decision between sphere and ellipsoid (Fig. 1.3d, see also Sect. 5.3.1) for a set of given points is a relevant issue to many applications. To explain the strategy for model selection, the following general procedure for least-squares model fitting has been assumed:

1. Observation (measurement) of the real object points
2. Proposition of a mathematical model description for the real object with unknown model parameters
3. Estimation of the model parameters by minimizing the square sum of the predefined error-of-fit between the object model and the object observation
4. Testing the reliability of the estimation results
5. If necessary, changing of the model and repetition of the procedure from the third step onwards. If not required, the procedure can then be terminated.

The ODF algorithms described in this work provide helpful information about the quality of the estimation results. This includes the square sum σ_0^2 (see (2.20)–(2.21)) of the Euclidean error distances and the parameter covariance matrix $\mathrm{Cov}(\hat{\mathbf{a}})$ (see (2.35)–(2.37)). The index σ_0^2 indicates the overall performance of the model selection and model fitting. The covariance matrix also makes a qualitative insight into the estimated parameters possible, such as the standard deviations and correlation coefficients of the estimated parameters. The general criteria for selecting the correct model feature for a set of given points are briefly noted below:

- A better model feature has a smaller performance index σ_0^2
- A simple model feature, i.e. a low-level model feature having a small number of parameters, is preferred to a complex one
- A reliable model feature has low variances and weak correlations of the estimated model parameters
- The *a posteriori* error distance (residual) $\sqrt{\sigma_0^2/(m+q-p)}$ resulting from the correct model selection and model fitting should be about the same as the *apriori* error level of the measuring device used.

The use of the above criteria is explained by comparing the sphere fitting with the ellipsoid fitting. The resulting performance index σ_0^2 of an ellipsoid fitting is generally smaller than that of a sphere fitting for the same set of points. Hence an ellipsoid could be regarded as a correct model feature, even though an ellipsoid is more complex than a sphere. However, if the set of points has a sphere-like distribution or represents only a small portion of a sphere/ellipsoid, the resulting σ_0^2 shows no significant difference between the ellipsoid fitting and the sphere fitting. Furthermore, the covariance matrix of the ellipsoid parameters suggests an overparameterization in terms of relatively large standard deviations and strong correlations between the estimated ellipsoid parameters. In this case, a sphere is thus selected as the correct (reliable) model feature for the given set of points. For further information, please refer to [28] for the statistical methods for model selection (hypothesis test).

1.2 Curve and Surface Fitting

1.2.1 Applications of Curve and Surface Fitting

The fitting of curves/surfaces (model) to a set of measurement data is the most frequently performed elementary task in many disciplines of science and engineering. In fact, almost every sector of science and engineering is concerned with curve or surface fitting. The fields of application of curve/surface fitting are very vast and practically unlimited. In this section, a short application list is given of model fitting to a set of given data points in 2-D/3-D space (point cloud, coordinate data set).

Historically, the least-squares method (LSM), the *de facto* mathematical tool for model fitting, was developed in 1795 by the German mathematician C.F. Gauss [36] and applied by him to locate the newly-discovered and then lost asteroid Ceres in 1801. Another example of application of model fitting in astronomy is the ellipse fitting to the image contour of a galaxy which is classified by its shape in telescopic image.

At the other end of the dimensional scale, model fitting finds its applications in computer-aided drug design and computational chemistry/biology. The molecular surface of a drug (ligand) requires matching with the binding site on the receptor (protein) surface in order to precondition energetic interaction (molecular docking) between the two molecules. In particle physics, curve fitting to the flight trajectory of a particle split from an atom by an accelerator is a task which is essential in order to determine particle properties.

With pattern recognition and computer/machine vision, model fitting plays an important role in extracting geometric information from objects taken in 2-D image or 3-D range image containing the curve/surface structures of the objects. The model parameters obtained are used for scene interpretation and object reconstruction. In the case of motion analysis - which is a substantial task carried out in robotics, ergonomics, sport science, and entertainment industry techniques - the motion parameters of a moving object are determined by fitting curves to the sampled points of the object trajectory.

Model fitting to the measurement points of real objects is a central task in coordinate metrology, reverse engineering, and CAD/CAM, where the workpieces require a mathematical description with their model parameters (parametric model recovery, Fig. 1.1). Model fitting is of particular interest in coordinate metrology, where the ultimate goal is the accurate and reliable determination of the geometric parameters (shape, size, position, and rotation parameters) of measurement objects. One of the methods possible with CAGD for converting a curve/surface to another one (geometric data exchange) is to fit the second curve/surface to the sampled points of the first one.

1.2.2 Algebraic Fitting Vs. Geometric Fitting

There are two categories of least-squares curve/surface fitting in literature, i.e. algebraic fitting and geometric fitting. These are differentiated by their respective definition of the error measure to be minimized [22], [34], [71], [75]. Although the terminology originated with the fitting problems of implicit curves/surfaces, it reflects the research efforts made in past decades to develop fitting algorithms for general curves/surfaces.

A highly representative class of curves/surfaces is the algebraic curve/surface described by an implicit polynomial equation with the algebraic parameters **b** (polynomial coefficients)

$$F(\mathbf{b},\mathbf{X}) \triangleq \sum_{j=1}^{p} b_j X^{k_j} Y^{l_j} = 0 \quad \text{with} \quad 0 \le k_j + l_j \le o \quad \forall j$$

for algebraic plane curves, and,

$$F(\mathbf{b},\mathbf{X}) \triangleq \sum_{j=1}^{p} b_j X^{k_j} Y^{l_j} Z^{m_j} = 0 \quad \text{with} \quad 0 \le k_j + l_j + m_j \le o \quad \forall j \tag{1.4}$$

for algebraic surfaces. The positive integer o is the order of polynomials and is equal to the maximum number of intersections between the algebraic curve/surface and a straight line. Assuming the use of arbitrary order polynomials, many kinds of curves/surfaces can be represented by algebraic curves/surfaces.

If $F(\mathbf{b},\mathbf{X}_i) \neq 0$, the given point $\mathbf{X}_i$ does not lie on the model feature, i.e. there is some error-of-fit. The error measure $F(\mathbf{b},\mathbf{X}_i)$ is termed *algebraic distance* in literature. A quick and straightforward method of fitting an algebraic curve/surface to a set of given points $\{\mathbf{X}_i\}_{i=1}^{m}$ is to determine the algebraic parameters $\{b_j\}_{j=1}^{p}$ which minimize the square sum of the algebraic distances at each given point (*algebraic fitting*) [22]

$$\min_{\mathbf{b}} \sum_{i=1}^{m} F^2(\mathbf{b},\mathbf{X}_i) \; . \tag{1.5}$$

Because the algebraic distance $F(\mathbf{b},\mathbf{X}_i)$ is expressed in linear terms of the algebraic parameters $\{b_j\}_{j=1}^{p}$, the algebraic fitting problem can be solved in closed form at low implementation and computing cost.

However, algebraic fitting has drawbacks in accuracy. Also, the physical interpretation of the algebraic distance $F(\mathbf{b},\mathbf{X}_i)$ and the parameters $\{b_j\}_{j=1}^{p}$ is generally not clear with regard to real applications. In exceptional cases, e.g. algebraic line/plane fitting (also known as linear regression in statistics)

$$\min_{a,b,c} \sum_{i=1}^{m} (Z_i - aX_i - bY_i - c)^2 \; , \tag{1.6}$$

the algebraic distance is the vertical distance of the given point $\mathbf{X}_i$ to the fitted line/plane. The algebraic parameters of a line/plane can be interpreted as the normal direction of the line/plane and as the distance (although not yet normalized through the length of the normal vector) between the line/plane and the origin of coordinate frame. Unfortunately, when the given points are sampled from a vertical object line/plane, algebraic line/plane fitting (1.6) gives a clearly unacceptable result, that of a slanted (biased) line/plane, due to the fact that the algebraic distance of any off-the-line/plane point becomes infinite with a vertical line/plane. Also, with algebraic circle/sphere fitting ([49], [65])

$$\min_{r,X_o,Y_o,Z_o} \sum_{i=1}^{m} [(X_i - X_o)^2 + (Y_i - Y_o)^2 + (Z_i - Z_o)^2 - r^2]^2 , \qquad (1.7)$$

the algebraic distance can be interpreted as the annulus area between the fitted circle (or the middle section of the fitted sphere) and the concentric circle passing through the given point $\mathbf{X}_i$. Especially if the data points are sampled from a small portion of an object circle/sphere, the algebraic circle/sphere fitting (1.7) gives a smaller (biased) circle/sphere than the original one. This is because the square sum (1.7) is minimized much more with a smaller annulus radius than with a smaller annulus width (a variation of the annulus radius results in a quadratic variation of the annulus area, while the annulus width results in a linear variation).

In the case of fitting general implicit curves/surfaces, it is highly difficult, if not impossible, to find a physical interpretation of the algebraic distance. Also, with the exception of algebraic features up to the second order (i.e. conic section and quadric surface), there is no efficient method for extracting the physical information (such as shape, size, location, and orientation) of the model feature from the estimated algebraic parameters. In other words, it is not known what has been minimized and then obtained in terms of physical quantities. The known disadvantages of algebraic fitting are listed as follows [8], [58], [60], [66], [67], [70]:

- Error definition does not comply with the measurement guidelines
- Conversion of the algebraic parameters to the physical parameters (shape, size, position, and rotation parameters) is highly difficult
- Fitting errors are weighted
- The estimated model parameters are biased
- It is very difficult to test the reliability of the estimated model parameters (particularly in terms of the physical parameters)
- The fitting procedure sometimes ends with an unintended model feature (e.g. a hyperbola instead of an ellipse)
- The model parameters are not invariant to coordinate transformation (e.g. a parallel shift of the set of given points causes changes not only in position but also in the form and rotation of the estimated model feature).

To enhance the algebraic distance $F(\mathbf{b}, \mathbf{X}_i)$, the first order approximation of the shortest distance of the given point $\mathbf{X}_i$ to the algebraic feature is recommended as the better error measure (normalized algebraic distance) [71], [76].

$$\min_{\mathbf{b}} \sum_{i=1}^{m} \left(\frac{F(\mathbf{b}, \mathbf{X}_i)}{\|\nabla F(\mathbf{b}, \mathbf{X}_i)\|} \right)^2 \quad \text{with} \quad \nabla \triangleq (\partial/\partial X, \partial/\partial Y, \partial/\partial Z)^{\mathrm{T}} . \qquad (1.8)$$

Although the use of normalized algebraic distance improves the results of algebraic fitting, the drawbacks listed above remain unchanged. As an unwanted side-effect, algebraic fitting with normalized algebraic distances cannot be solved in closed form. However, because of the advantages regarding implementation and computing costs, the algebraic distance $F(\mathbf{b}, \mathbf{X}_i)$ (1.5) or normalized algebraic distance $F(\mathbf{b}, \mathbf{X}_i)/\|\nabla F(\mathbf{b}, \mathbf{X}_i)\|$ (1.8) has been selected as the error measure for many applications.

In *geometric fitting,* also known as orthogonal distance fitting or best fitting, error distance is defined as the shortest distance *(geometric distance)* from the given point to the model feature

$$\min_{\mathbf{b}, \{\mathbf{X}'_i\}_{i=1}^m \in F} \sum_{i=1}^{m} \|\mathbf{X}_i - \mathbf{X}'_i\|^2 , \tag{1.9}$$

where $\{\mathbf{X}'_i\}_{i=1}^m$ are the nearest points on the model feature F to each given point $\{\mathbf{X}_i\}_{i=1}^m$. With applications of model fitting in 2-D/3-D space, the natural and best choice of error measure is the geometric distance $\|\mathbf{X}_i - \mathbf{X}'_i\|$ [1], [29], because a measurement point is the probable observation of the nearest object point to the measurement point [1]. The use of geometric distance as the error measure to be minimized by model fitting is also prescribed in coordinate metrology guidelines [58], [60]. Because the geometric distance is invariant to coordinate transformation (translation and rotation), it avoids some of the drawbacks of algebraic fitting listed on p. 11.

Unfortunately, contrary to its clear and preferable definition, the geometric distance is generally nonlinear to the model parameters (compared with the algebraic distance $F(\mathbf{b}, \mathbf{X}_i)$ which is linear to the algebraic parameters b (1.4)). Taking this fact alone into consideration, the tasks involved in computing and minimizing the square sum of the geometric distances (1.9) for general features are highly complex, with the exception of a few model features (line/plane and circle/sphere). The problem of geometric line/plane fitting can be solved in closed form [61] because the geometric distance is linear to the line/plane parameters. For circles/spheres which have no rotation parameters (highly nonlinear by nature) and permit the geometric distance to be described in closed form, a stable geometric fitting algorithm exists [87].

Due to the analytical and computational difficulties involved, a geometric fitting algorithm for general model features was reported in literature first in 1987 [20]. In order to bypass the difficulties, diverse approximating measures (including the normalized algebraic distance (1.8)) of the geometric distance have been used particularly for applications in computer/machine vision [26], [32], [48], [51], [53], [67], [73]. Due to the fact that the performance gap between algebraic fitting and geometric fitting is wide and unable to be bridged using the approximating measures of the geometric distance, many research efforts have been made to develop geometric fitting algorithms (see the next section for a review [20], [25], [34], [44], [45], [61], [74], [75], [87]).

Besides choosing the geometric distance as the error measure to be minimized, additional algorithmic features are also required for a geometric fitting algorithm in order to completely overcome the drawbacks of algebraic fitting as listed on p. 11.

Taking into account the rigid body motion (see Sect. 1.1.2) and the difficulty in converting the algebraic parameters **b** to the physical parameters **a**, a geometric fitting algorithm would be highly advantageous for many applications if model parameters **a** can be estimated in terms of form $\mathbf{a}_\mathrm{g}$, position $\mathbf{a}_\mathrm{p}$, and rotation parameters $\mathbf{a}_\mathrm{r}$

$$\min_{\mathbf{a}_\mathrm{g},\mathbf{a}_\mathrm{p},\mathbf{a}_\mathrm{r},\{\mathbf{X}'_i\}_{i=1}^{m}\in F} \sum_{i=1}^{m} \|\mathbf{X}_i - \mathbf{X}'_i\|^2 .$$

Moreover, a geometric fitting algorithm would be helpful for applications, particularly in coordinate metrology, if the reliability (variance) of the estimated parameters could be tested. In summary, as far as data points in 2-D/3-D space are concerned, the objective is to develop generally applicable and computationally efficient algorithms for the geometric fitting (orthogonal distance fitting) of curves and surfaces, which estimate model parameters in terms of form, position, and rotation parameters, and provide parameter testing possibilities.

1.2.3 State-of-the-Art Orthogonal Distance Fitting

A century ago, K. Pearson [61] elegantly solved the ODF problem of line/plane in space in closed form by using the moment method (see Sect. 2.1). The ODF problem of any other curve/surface than a line/plane is a *nonlinear least-squares* problem which should thus be solved using iteration. Although there are a few dedicated ODF algorithms for relatively simple model features such as circles/spheres and ellipses [34], [87], in this section the general ODF algorithms for explicit [20], implicit [44], [75], and parametric [25], [45], [74] curves and surfaces are reviewed (Table 1.1 and Table 1.2).

Boggs et al. [20] developed a general ODF algorithm for *explicit* features $Z = F(\mathbf{b}, X, Y)$ and laid open the program source code [21]. Since then, their-algorithm has been chosen as the standard ODF algorithm for many applications, despite the limited usability of the explicit features as mentioned in Sect. 1.1.1. Their algorithm simultaneously determines the model parameters $\{b_i\}_{i=1}^{p}$ and the

Table 1.2. Orthogonal distance fitting algorithms for curves and surfaces

Algorithm	Curve/surface	Determination of $\mathbf{a}$, $\mathbf{b}$, $\{\mathbf{X}'_i\}_{i=1}^{m}$	Parameter grouping
Boggs [20]	$Z = F(\mathbf{b}, X, Y)$, explicit	Simultaneous	None
Helfrich [44], Sullivan [75]	$F(\mathbf{b}, \mathbf{X}) = 0$, implicit	Alternate	None
Butler [25], Helfrich [45]	$\mathbf{X}(\mathbf{b}, \mathbf{u})$, parametric	Alternate	None
Sourlier [74]	$\mathbf{X}(\mathbf{a}, \mathbf{u})$, parametric	Simultaneous	$\mathbf{a}_\mathrm{g}$, $\mathbf{a}_\mathrm{p}$, $\mathbf{a}_\mathrm{r}$

nearest points $\{\mathbf{X}_i'\}_{i=1}^m$ on model feature from each given point $\{\mathbf{X}_i\}_{i=1}^m$. With the realistic assumption of $m \gg p$, memory space usages and computing costs increase to the order of $O(m^2)$ and $O(m^3)$, respectively, unless a sparse matrix algorithm [38] is used additionally. The model parameters $\{b_i\}_{i=1}^p$ are not estimated in terms of form, position, and rotation parameters.

Helfrich et al. [44] presented a general ODF algorithm for *implicit* features $F(\mathbf{b}, \mathbf{X}) = 0$ without the grouping of the parameters of form, position, and rotation. Because their algorithm alternately determines the model parameters $\{b_i\}_{i=1}^p$ and the shortest distance points $\{\mathbf{X}_i'\}_{i=1}^m$ in a nested iteration scheme, memory space usages and computing costs are proportional to the number of data points, i.e. to the order of $O(m)$. A similar algorithm for algebraic features (1.4) is presented in [75].

Butler et al. [25] formulated a general ODF algorithm for *parametric* features $\mathbf{X}(\mathbf{b}, \mathbf{u})$ without the grouping of the parameters of form, position, and rotation. Their algorithm alternately determines the model parameters $\{b_i\}_{i=1}^p$ and the shortest distance points $\{\mathbf{X}_i'\}_{i=1}^m$. A drawback of this algorithm is the poor convergence performance on 3-D curve fitting [77]. A similar algorithm is presented in [45].

Sourlier [74] developed a new general ODF algorithm for *parametric* features $\mathbf{X}(\mathbf{a}, \mathbf{u})$ *with* the grouping of the parameters of form, position, and rotation. A further advantage of this algorithm is the low implementation cost for a new model feature, because only the first derivatives of $\mathbf{X}(\mathbf{a}, \mathbf{u})$ are required. The algorithm simultaneously determines the model parameters $\{a_i\}_{i=1}^p$ and the shortest distance points $\{\mathbf{X}_i'\}_{i=1}^m$, thus requiring considerable memory space and high computing cost if a sparse matrix algorithm is not used additionally. Another drawback of this algorithm is the poor convergence performance if the initial parameter values are not close enough to the final estimation values.

Reviewing the current ODF algorithms in literature (Table 1.2) and considering the usability of implicit and parametric features compared to that of explicit features (Sect. 1.1.1), the current ODF algorithms do not possess all of the following algorithmic advantages desired:

- Fitting of general implicit or parametric features defined in 2-D/3-D space
- Estimation of the model parameters in terms of form, position, and rotation parameters
- Robust and rapid convergence
- Low computing cost and memory space usage
- Low implementation cost for a new model feature.

If an ODF algorithm could be developed for use with all the above-mentioned advantages, it would be highly promising for 2-D/3-D data processing applications in many disciplines of science and engineering as listed in Sect. 1.2.1.

1.2.4 ISO 10360-6 and Industrial Requirements of CMM Software Tools

In the early 1980's, when all the hardware (mechanical and electronic) technologies essential to the construction of a highly-accurate coordinate measuring machine

(CMM) were being developed and came into use [83], [84], coordinate metrology researchers examined the software tools integrated in CMM for fitting model curves/surfaces to measurement points. Surprisingly, they found there were significant inconsistencies in the CMM software tools provided by different CMM manufacturers [85], [86], [82]. The CMM as a system consisting of hardware and software was then suddenly suspected as being unreliable, thus triggering a software crisis in the circle of CMM users and manufacturers.

As the applied algorithm techniques were kept confidential by each CMM manufacturer (this situation remains unchanged even now), it was practically impossible to eliminate the software inconsistencies by comparing the algorithm techniques used and, if necessary, by duplicating the best one. Consequently, a series of international research projects was launched in order to establish an objective method for testing CMM software tools without requiring CMM manufacturers to disclose their software techniques (black-box approach). The research results have now been crystallized in the recently ratified ISO 10360-6 [60]:

- **Testing method:** evaluation of the accuracy of a CMM software tool by comparing the parameter values of the test model features estimated by the software under test with the reference values prepared by the reference software [63], [29], [24]
- **Feature parameterization:** definition of the effective and competent set of parameters representing the test model feature [14] (Table B.1)
- **Parameter estimation criterion:** minimization of the square sum of predefined error measures (least-squares method [36])
- **Error measure definition:** geometric distance between the test model feature and the given point [29], [58]
- **Set of data points:** rectangular coordinate values of a 2-D/3-D set of points emulating the real measurement points of the test model feature arbitrarily located and oriented in the predefined measurement volume [42], [81].

The testing procedure is carried out for a maximum of nine test model features (Table B. 1). For each test model feature, multiple sets of a few data points are prepared. A set of points represents only a small portion of a test model feature usually having an extreme shape such as the quarter face of a long-rod cylinder or of a thin-disk cylinder (as an example, see the set of points used by the cone fitting in Sect. 3.3.2).

The point measurement accuracy of the high-end products on the current CMM market is at the level of 1–2 μm for a measurement volume of 1 m^3. To retain the overall accuracy of CMM's at a high level, the CMM software tool used for processing the measurement data must be at least 5–10 times more accurate than the point measurement accuracy of the host CMM. ISO 10360-6 gives full details of the accuracy evaluation and testing method for CMM software tools, without giving any indication of how to realize a highly-accurate CMM software tool. With ISO 10360-6 now coming into force, CMM manufacturers are required to present their own certificate of software accuracy to their client.

The CMM software tool based on the new ODF algorithms described in this work has successfully passed the testing procedures carried out by the German federal authority PTB and has obtained the supreme grade of software accuracy (higher than $0.1\ \mu\mathrm{m}$ for length unit and higher than $0.1\ \mu\mathrm{rad}$ for angle unit for all parameters of all test model features with all sets of data points). In the following three chapters, a detailed description is given of four new ODF algorithms and one known algorithm in a well-organized and easily understandable manner. In Chap. 2, after an introductory review of the moment method for line/plane fitting, two computational frameworks for solving the general ODF problem are proposed. The applications of the frameworks for the ODF problems of implicit and parametric curves/surfaces are described in Chap. 3 and Chap. 4, respectively. Appendix B gives the key formulas necessary for implementing the new algorithms for the nine test model features defined in ISO 10360-6.

2. Least-Squares Orthogonal Distance Fitting

The orthogonal distance fitting (ODF) of general curves and surfaces is by nature an analytically and computationally difficult problem. In order to start with a soft example, this chapter begins by reviewing the moment method (linear ODF) for line/plane fitting which K. Pearson [61] solved in closed form a century ago. The general aim of ODF is then defined and classified into three categories of ODF problems encountered in applications, namely, point-to-point matching, template matching, and curve/surface fitting. For solving general nonlinear ODF problems in a unified manner, two computational frameworks are proposed. These are the *distance-based algorithm* and the *coordinate-based algorithm;* their implementation for general implicit and parametric curves/surfaces is explained in Chap. 3 and Chap. 4, respectively. In this chapter, to give simple application examples of the proposed frameworks, the circle/sphere fitting problem is solved in a straight manner due to the fact that a circle/sphere has no rotation parameters and permits the geometric error distance to be described in closed form.

2.1 Moment Method for Line and Plane Fitting

The orthogonal distance fitting of line/plane is a linear problem and can thus be solved in closed form at a very low computing cost. Although it is the easiest ODF problem and was solved long ago, the significance of line/plane fitting should not be overlooked because most man-made objects can be represented by lines/planes. Moreover, because of its closed-form solution and low computing cost, the resulting parameter values from the line/plane fitting should be used wherever possible as the initial parameter values for the iterative fitting of other model features (see Fig. 1.3). For formulating the problem and solution of line/plane fitting in a comprehensive manner, the concept of point mechanism [17] and the modern matrix computational tool of singular value decomposition (SVD) has been used [40], [64].

2.1.1 Line Fitting

The line containing the point $\mathbf{X}_o$ and parallel to the unit vector $\mathbf{r}$ in n-dimensional space $(n \geq 2)$ can be described in parametric form below:

$$\mathbf{X}_o + u\mathbf{r} \qquad \text{with} \qquad \|\mathbf{r}\| = 1 \qquad \text{and} \qquad -\infty \leq u \leq \infty \, . \tag{2.1}$$

The square sum of the orthogonal distances from each given point $\{\mathbf{X}_i\}_{i=1}^m$ to the line (2.1) is

$$\sigma_0^2 \triangleq \sum_{i=1}^m \|(\mathbf{X}_i - \mathbf{X}_o) \times \mathbf{r}\|^2 . \tag{2.2}$$

Given $\mathbf{r}$, $\mathbf{X}_o$ is looked for which minimizes (2.2), e.g. for $n = 3$,

$$\frac{\partial \sigma_0^2}{\partial \mathbf{X}_o} = -2\left(\sum_{i=1}^m (X_i - X_o),\ \sum_{i=1}^m (Y_i - Y_o),\ \sum_{i=1}^m (Z_i - Z_o)\right)(\mathbf{I} - \mathbf{r}\mathbf{r}^\mathrm{T}) = \mathbf{0}^\mathrm{T} . \tag{2.3}$$

With $\mathbf{r}$ of arbitrary direction not parallel to any axis of the coordinate frame XYZ, there is only the trivial solution to (2.3)

$$\sum_{i=1}^m (X_i - X_o) = \sum_{i=1}^m (Y_i - Y_o) = \sum_{i=1}^m (Z_i - Z_o) = 0 . \tag{2.4}$$

From (2.4), we obtain (*parallel-axis theorem* [17])

$$X_o = \bar{X} = \frac{1}{m}\sum_{i=1}^m X_i\ , \quad Y_o = \bar{Y} = \frac{1}{m}\sum_{i=1}^m Y_i\ , \quad Z_o = \bar{Z} = \frac{1}{m}\sum_{i=1}^m Z_i\ . \tag{2.5}$$

If $\mathbf{r}$ is parallel to one of the coordinate frame axes, the coordinate value of $\mathbf{X}_o$ may be set along this axis in an arbitrary fashion and the trivial solution (2.5) still applies.

If

$$\begin{aligned} x_i &= X_i - X_o\ , & y_i &= Y_i - Y_o\ , & z_i &= Z_i - Z_o\ , \\ M_{xx} &= \sum_{i=1}^m x_i^2\ , & M_{yy} &= \sum_{i=1}^m y_i^2\ , & M_{zz} &= \sum_{i=1}^m z_i^2\ , \\ M_{xy} &= \sum_{i=1}^m x_i y_i\ , & M_{yz} &= \sum_{i=1}^m y_i z_i\ , & M_{zx} &= \sum_{i=1}^m z_i x_i \end{aligned}$$

are set and the matrix $\mathbf{H}$ (*inertia tensor of centroidal moment* [17]) defined

$$\mathbf{H} \triangleq \begin{pmatrix} M_{yy} + M_{zz} & -M_{xy} & -M_{zx} \\ -M_{xy} & M_{zz} + M_{xx} & -M_{yz} \\ -M_{zx} & -M_{yz} & M_{xx} + M_{yy} \end{pmatrix} , \tag{2.6}$$

the square sum of the orthogonal distances (2.2) becomes

$$\sigma_0^2 = \mathbf{r}^\mathrm{T}\mathbf{H}\mathbf{r} \tag{2.7}$$

with the line

$$\bar{\mathbf{X}} + u\mathbf{r} .$$

The symmetric square matrix $\mathbf{H}$ (2.6), extended for n-dimensional space ($n \geq 2$) without loss of generality, is decomposed by the SVD

$$\mathbf{H} = \mathbf{V_H}\mathbf{W_H}\mathbf{V_H^T} \,, \tag{2.8}$$

with

$$\begin{aligned} \mathbf{W_H} &= [\mathrm{diag}(w_{\mathbf{H}1}, \ldots, w_{\mathbf{H}n})] \,, \\ \mathbf{V_H} &= (\mathbf{v}_{\mathbf{H}1}, \ldots, \mathbf{v}_{\mathbf{H}n}) \,, \quad \text{where} \quad \mathbf{V_H}^{\mathrm{T}}\mathbf{V_H} = \mathbf{I} \,. \end{aligned} \tag{2.9}$$

The singular values $w_{\mathbf{H}1}, \ldots, w_{\mathbf{H}n}$ and the orthogonal vectors $\mathbf{v}_{\mathbf{H}1}, \ldots, \mathbf{v}_{\mathbf{H}n}$ are the *principal centroidal moments* and the *principal axes* of inertia, respectively. If $\mathbf{r}$ is parallel to one of the principal axes $\mathbf{v}_{\mathbf{H}j}$, from (2.7)–(2.9), the following is obtained:

$$\begin{aligned} \sigma_0^2 &= \mathbf{r}^{\mathrm{T}}\mathbf{H}\mathbf{r} \\ &= \mathbf{v}_{\mathbf{H}j}^{\mathrm{T}}\mathbf{V_H}\mathbf{W_H}\mathbf{V_H^T}\mathbf{v}_{\mathbf{H}j} \\ &= w_{\mathbf{H}j} \,. \end{aligned} \tag{2.10}$$

From this fact, when choosing the index j of the smallest $w_{\mathbf{H}j}$ (2.9), the ODF problem of line is solved. The resultant ODF line (2.1) is

$$\bar{\mathbf{X}} + u\mathbf{v}_{\mathbf{H}j} \,.$$

In summary, the ODF line for the m given points $\{\mathbf{X}_i\}_{i=1}^m$ in n-dimensional space ($n \geq 2$) passes through the mass center (2.5) and is parallel to one of the principal axes of inertia associating with the smallest principal centroidal moment [2], [37], [61].

For an experimental example, the $m = 13$ coordinate pairs ($n = 2$) in Table 2.1 were taken and the following results obtained:

$$\mathbf{W_H} = \begin{pmatrix} 5.05624 & 0.00000 \\ 0.00000 & 0.06296 \end{pmatrix} \,, \quad \mathbf{X}_o = \bar{\mathbf{X}} = \begin{pmatrix} 0.13108 \\ 0.00000 \end{pmatrix} \,, \quad \text{and}$$
$$\mathbf{V_H} = \begin{pmatrix} -0.99989 & 0.01504 \\ -0.01504 & -0.99989 \end{pmatrix} \,.$$

The smaller principal centroidal moment is $w_{\mathbf{H}2} = 0.06296$ and the resultant ODF line (2.1) is

Table 2.1. 13 coordinate pairs ($n = 2$) distributed about a vertical line [12]

X	0.237	0.191	0.056	0.000	0.179	0.127	0.089
Y	−1.000	−0.833	−0.667	−0.500	−0.333	−0.167	0.000
X	0.136	0.202	0.085	0.208	0.156	0.038	
Y	0.167	0.333	0.500	0.667	0.833	1.000	

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} 0.13108 \\ 0.00000 \end{pmatrix} + u \begin{pmatrix} 0.01504 \\ -0.99989 \end{pmatrix} ,$$

or, in implicit form,

$$0.99989X + 0.01504Y - 0.13108 = 0 ,$$

with $\sigma_0^2 = 0.06296$.

2.1.2 Plane Fitting

Analogously to the case of line fitting, the ODF problem of plane can also be solved. The plane containing the point $\mathbf{X}_o$ and normal to the unit vector $\mathbf{n}$ in n-dimensional space ($n \geq 2$) can be described in implicit form below:

$$(\mathbf{X} - \mathbf{X}_o)^{\mathrm{T}}\mathbf{n} = 0 \qquad \text{with} \qquad \|\mathbf{n}\| = 1 . \tag{2.11}$$

The square sum of the orthogonal distances from each given point $\{\mathbf{X}_i\}_{i=1}^{m}$ to the plane (2.11) is

$$\begin{aligned} \sigma_0^2 &\triangleq \sum_{i=1}^{m} \left[(\mathbf{X}_i - \mathbf{X}_o)^{\mathrm{T}}\mathbf{n}\right]^2 \\ &= \mathbf{n}^{\mathrm{T}} \left[\sum_{i=1}^{m}(\mathbf{X}_i - \mathbf{X}_o)(\mathbf{X}_i - \mathbf{X}_o)^{\mathrm{T}}\right] \mathbf{n} . \end{aligned} \tag{2.12}$$

Given $\mathbf{n}$, $\mathbf{X}_o$ is looked for which minimizes (2.12), e.g. for $n = 3$,

$$\frac{\partial \sigma_0^2}{\partial \mathbf{X}_o} = -2 \left(\sum_{i=1}^{m}(X_i - X_o), \sum_{i=1}^{m}(Y_i - Y_o), \sum_{i=1}^{m}(Z_i - Z_o) \right) \mathbf{n}\mathbf{n}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}} . \tag{2.13}$$

The same trivial solution (2.5) is obtained for $\mathbf{X}_o$ as in the case of line fitting and the square sum of the orthogonal distances (2.12) becomes

$$\sigma_0^2 = \mathbf{n}^{\mathrm{T}}\mathbf{M}\mathbf{n} \tag{2.14}$$

with

$$\mathbf{M} \triangleq \begin{pmatrix} M_{xx} & M_{xy} & M_{zx} \\ M_{xy} & M_{yy} & M_{yz} \\ M_{zx} & M_{yz} & M_{zz} \end{pmatrix} . \tag{2.15}$$

The symmetric square matrix $\mathbf{M}$ (*central moments tensor* [17]), extended for n-dimensional space ($n \geq 2$) without loss of generality, is decomposed by the SVD

$$\mathbf{M} = \mathbf{V}_{\mathbf{M}}\mathbf{W}_{\mathbf{M}}\mathbf{V}_{\mathbf{M}}^{\mathrm{T}} , \tag{2.16}$$

with

$$\begin{aligned} \mathbf{W}_{\mathbf{M}} &= [\mathrm{diag}(w_{\mathbf{M}1}, \ldots, w_{\mathbf{M}n})] , \\ \mathbf{V}_{\mathbf{M}} &= (\mathbf{v}_{\mathbf{M}1}, \ldots, \mathbf{v}_{\mathbf{M}n}) , \quad \text{where} \quad {\mathbf{V}_{\mathbf{M}}}^{\mathrm{T}}\mathbf{V}_{\mathbf{M}} = \mathbf{I} . \end{aligned} \tag{2.17}$$

Table 2.2. Four coordinate triples ($n = 3$) [61]

X	2	2	4	4
Y	16	26	16	26
Z	219	261	127	231

When choosing $\mathbf{n}$ parallel to the vector $\mathbf{v}_{\mathbf{M}j}$ having the smallest $w_{\mathbf{M}j}$, the ODF problem of plane is solved. The resultant ODF plane (2.11) is

$$(\mathbf{X} - \bar{\mathbf{X}})^{\mathrm{T}} \mathbf{v}_{\mathbf{M}j} = 0 .$$

In summary, the ODF plane for the m given points $\{\mathbf{X}_i\}_{i=1}^{m}$ in n-dimensional space ($n \geq 2$) contains the mass center (2.5) and is normal to one of the principal axes of central moments (2.15) associating with the smallest principal central moment [37], [61].

For an experimental example, the $m = 4$ coordinate triples ($n = 3$) in Table 2.2 were taken and the following results obtained:

$$\mathbf{W}_{\mathbf{M}} = \begin{pmatrix} 10066.0 & & \mathbf{0} \\ & 0.79134 & \\ \mathbf{0} & & 48.25736 \end{pmatrix}, \quad \mathbf{X}_{\mathrm{o}} = \bar{\mathbf{X}} = \begin{pmatrix} 3.0 \\ 21.0 \\ 209.5 \end{pmatrix}, \quad \text{and}$$

$$\mathbf{V}_{\mathbf{M}} = \begin{pmatrix} -0.01209 & 0.98146 & -0.19130 \\ 0.07305 & -0.18994 & -0.97908 \\ 0.99726 & 0.02581 & 0.06940 \end{pmatrix} .$$

The smallest principal central moment is $w_{\mathbf{M}2} = 0.79134$ and the resultant ODF plane (2.11) becomes

$$\begin{pmatrix} X - 3.0 \\ Y - 21.0 \\ Z - 209.5 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} 0.98146 \\ -0.18994 \\ 0.02581 \end{pmatrix} = 0 ,$$

with $\sigma_0^2 = 0.79134$.

2.1.3 Relationship Between Line and Plane Fitting

If the inertia tensor $\mathbf{H}$ (2.6) is compared with the central moments tensor $\mathbf{M}$ (2.15), from (2.8) and (2.16)–(2.17),

$$\begin{aligned} \mathbf{H} &= \mathrm{tr}(\mathbf{M})\mathbf{I} - \mathbf{M} \\ &= \mathrm{tr}(\mathbf{M})\mathbf{I} - \mathbf{V}_{\mathbf{M}}\mathbf{W}_{\mathbf{M}}\mathbf{V}_{\mathbf{M}}^{\mathrm{T}} \\ &= \mathbf{V}_{\mathbf{M}}\left[\mathrm{tr}(\mathbf{M})\mathbf{I} - \mathbf{W}_{\mathbf{M}}\right]\mathbf{V}_{\mathbf{M}}^{\mathrm{T}} \end{aligned} \tag{2.18}$$

is obtained, where $\mathrm{tr}(\mathbf{M})$ is the trace of $\mathbf{M}$ (trace of tensor is invariant to the rotation of coordinate frame [23])

$$\begin{aligned}\mathrm{tr}(\mathbf{M}) &\triangleq \sum_{j=1}^{n} M_{jj} \\ &= \sum_{j=1}^{n} w_{\mathbf{M}j} \,.\end{aligned} \tag{2.19}$$

From (2.8) and (2.18)–(2.19), the following can be concluded

$$\mathbf{W_H} = \sum_{j=1}^{n} w_{\mathbf{M}j}\mathbf{I} - \mathbf{W_M} \qquad \text{and} \qquad \mathbf{V_H} = \mathbf{V_M} \,.$$

This means that ([37], [61]):

- The pose of the principal axes is the same to both line and plane fitting
- The axis chosen for the line fitting is the worst one for the plane fitting, and vice versa
- Two axes chosen for line and plane fitting, respectively, are orthogonal
- Both line and plane fitting can be carried out using either of the two tensor matrices, the inertia tensor **H** (2.6) or the central moment tensor **M** (2.15).

2.2 Generalized Orthogonal Distance Fitting

2.2.1 Problem Definition

The ultimate goal of the orthogonal distance fitting of a model feature to a set of given points in space is the determination of the model parameters which *minimize* the square sum of the *minimum* distances between the given points and the model feature. To be exact, two sub-problems need to be solved, i.e. not only the square sum but also every single distance between the given points and the model feature should be minimized.

In order to handle these ODF problems in a general and easily understandable manner using vectors and matrices, two performance indices (cost functions) are introduced which represent in two different ways the square sum of the weighted distances between the given points and the model feature:

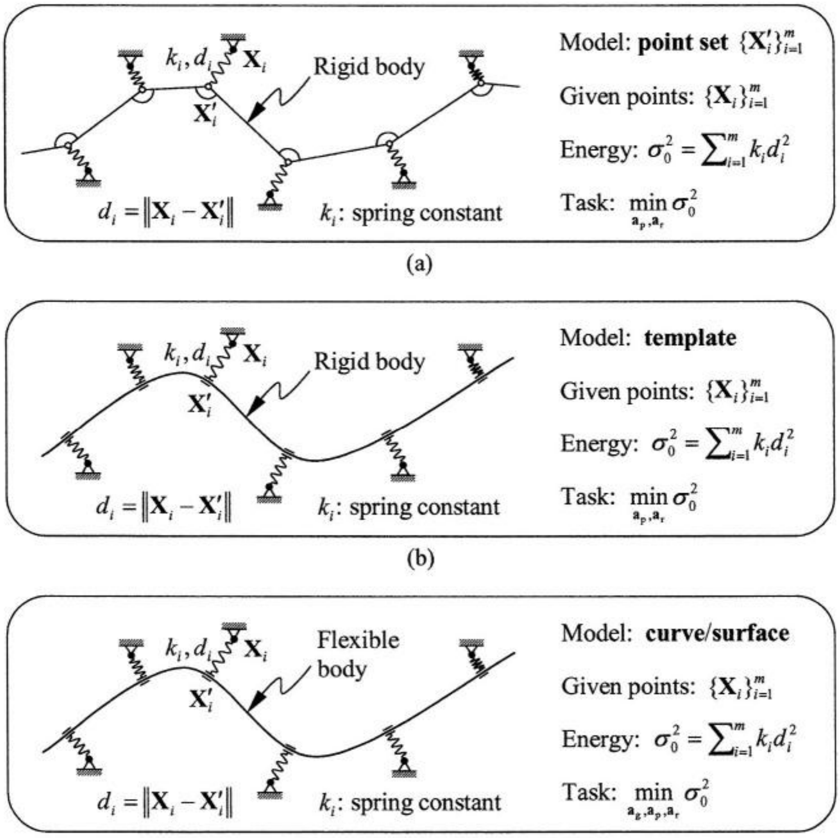
$$\begin{aligned}\sigma_0^2 &\triangleq \|\mathbf{Pd}\|^2 \\ &= \mathbf{d}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}\mathbf{d}\end{aligned} \tag{2.20}$$

and

$$\begin{aligned}\sigma_0^2 &\triangleq \|\mathbf{P}(\mathbf{X} - \mathbf{X}')\|^2 \\ &= (\mathbf{X} - \mathbf{X}')^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}(\mathbf{X} - \mathbf{X}') \,,\end{aligned} \tag{2.21}$$

where

- $\mathbf{X}$: coordinate column vector of the m given points $\{\mathbf{X}_i\}_{i=1}^m$, $\mathbf{X}^{\mathrm{T}} \triangleq (\mathbf{X}_1^{\mathrm{T}}, \ldots, \mathbf{X}_m^{\mathrm{T}})$, $\mathbf{X}_i^{\mathrm{T}} = (X_i, Y_i, Z_i)$
- $\mathbf{X}'$: coordinate column vector of the m corresponding points $\{\mathbf{X}_i'\}_{i=1}^m$ on the model feature, $\mathbf{X}'^{\mathrm{T}} \triangleq (\mathbf{X}_1'^{\mathrm{T}}, \ldots, \mathbf{X}_m'^{\mathrm{T}})$, $\mathbf{X}_i'^{\mathrm{T}} = (X_i', Y_i', Z_i')$
- $\mathbf{d}$: distance column vector, $\mathbf{d} \triangleq (d_1, \ldots, d_m)^{\mathrm{T}}$ with
  $d_i = \|\mathbf{X}_i - \mathbf{X}_i'\| = \sqrt{(\mathbf{X}_i - \mathbf{X}_i')^{\mathrm{T}}(\mathbf{X}_i - \mathbf{X}_i')}$
- $\mathbf{P}^{\mathrm{T}}\mathbf{P}$: weighting matrix or error covariance matrix (positive definite)
- $\mathbf{P}$: nonsingular symmetric matrix [28].



k_i, d_i
$\mathbf{X}_i$
Rigid body
$\mathbf{X}_i'$
$d_i = \|\mathbf{X}_i - \mathbf{X}_i'\|$
k_i: spring constant
Model: **point set** $\{\mathbf{X}_i'\}_{i=1}^m$
Given points: $\{\mathbf{X}_i\}_{i=1}^m$
Energy: $\sigma_0^2 = \sum_{i=1}^m k_i d_i^2$
Task: $\min_{\mathbf{a}_p, \mathbf{a}_r} \sigma_0^2$
(a)
k_i, d_i
$\mathbf{X}_i$
Rigid body
$\mathbf{X}_i'$
$d_i = \|\mathbf{X}_i - \mathbf{X}_i'\|$
k_i: spring constant
Model: **template**
Given points: $\{\mathbf{X}_i\}_{i=1}^m$
Energy: $\sigma_0^2 = \sum_{i=1}^m k_i d_i^2$
Task: $\min_{\mathbf{a}_p, \mathbf{a}_r} \sigma_0^2$
(b)
k_i, d_i
$\mathbf{X}_i$
Flexible body
$\mathbf{X}_i'$
$d_i = \|\mathbf{X}_i - \mathbf{X}_i'\|$
k_i: spring constant
Model: **curve/surface**
Given points: $\{\mathbf{X}_i\}_{i=1}^m$
Energy: $\sigma_0^2 = \sum_{i=1}^m k_i d_i^2$
Task: $\min_{\mathbf{a}_g, \mathbf{a}_p, \mathbf{a}_r} \sigma_0^2$
(c)

Fig. 2.1. Variety of orthogonal distance fitting tasks as energy minimization problem. The spring constants $\{k_i\}_{i=1}^m$ correspond to the diagonal elements of the weighting matrix $\mathbf{P}^{\mathrm{T}}\mathbf{P}$ in (2.20)–(2.21). With most ODF tasks, putting $\mathbf{P}^{\mathrm{T}}\mathbf{P} = \mathbf{I}$ is allowed: (a) Point-to-point matching; (b) Template matching; (c) Curve/surface fitting

If the independent identically-distributed (i.i.d.) measurement errors are assumed, the two performance indices (2.20) and (2.21) have the same value up to the *apriori* standard deviation of the measurement errors (Pythagorean theorem).

The definition of the performance indices (2.20) and (2.21) does not suppose the type of model feature nor how to determine the corresponding points $\{\mathbf{X}'_i\}_{i=1}^m$ on the model feature to the given points $\{\mathbf{X}_i\}_{i=1}^m$. Thus, paradoxically, a variety of generally applicable algorithms for ODF problems can be derived from (2.20)–(2.21). Any probable ODF algorithm which minimizes the performance indices (2.20) or (2.21) is characterized as *distance-based algorithm* or *coordinate-based algorithm,* respectively (see Sect. 2.3). On the other hand, according to the type of the model feature to be handled by the ODF algorithms, a classification of ODF problems arising with real applications into three categories can be made, i.e. point-to-point matching, curve/surface fitting and template matching (Fig. 2.1).

2.2.2 Point-to-Point Matching

If the set of corresponding points $\{\mathbf{X}'_i\}_{i=1}^m$ is given as another set of points (in this case, the model feature is a point template without the form parameters $\mathbf{a}_g$), the ODF problem becomes a *point-to-point matching* problem determining the rigid body motion (the position parameters $\mathbf{a}_p$ and the rotation parameters $\mathbf{a}_r$) of the set of corresponding points (Fig. 2.1 a). Once the point correspondences betwen the two sets of points have been given, the point-to-point matching problem can be solved in closed form, for which several solutions are available [15], [31], [46], [69]. The resultant σ_0^2 represents the quality-of-match. When the rigidity-of-match information (variance of the position and rotation parameters) is necessary for applications, it can be derived from the parameter covariance matrix Cov($\hat{\mathbf{a}}$) (2.37). The time-consuming element of the problem, particularly where each set of points is composed of a large number of points (such as obtained by laser radar) is the establishment of the point correspondences between the two sets of points (searching for the minimum distance points), a task of $O(m^2)$ cost. As a further development of the point-to-point matching algorithm, the iterative closest point (ICP [19]) algorithm alternately repeats the determination of the rigid body motion of the second set of points and the reestablishment of the point correspondences between the two sets of points. The point-to-point matching problem is not dealt with in this work.

2.2.3 Curve and Surface Fitting

With curve/surface fitting, the corresponding points $\{\mathbf{X}'_i\}_{i=1}^m$ are constrained to being membership points of a curve/surface in space (Fig. 2.1c). As the model feature S, the following general implicit/parametric curves and surfaces are handled in this work (see Table 1.1):

- Implicit 2-D curve ($n = 2$)

$$S = \{\mathbf{X} \in \mathbb{R}^2 \,|\, F(\mathbf{a}, \mathbf{X}) = 0 \quad \text{with} \quad \mathbf{a} \in \mathbb{R}^p\}$$

- Implicit surface ($n = 3$)

$$S = \{\mathbf{X} \in \mathbb{R}^3 \,|\, F(\mathbf{a}, \mathbf{X}) = 0 \quad \text{with} \quad \mathbf{a} \in \mathbb{R}^p\}$$

- Parametric 2-D curve ($n = 2$)

$$S = \{\mathbf{X} \in \mathbb{R}^2 \,|\, \mathbf{X}(\mathbf{a}, u) \quad \text{with} \quad \mathbf{a} \in \mathbb{R}^p \quad \forall u \in \mathbb{R}\}$$

- Parametric 3-D curve ($n = 3$)

$$S = \{\mathbf{X} \in \mathbb{R}^3 \,|\, \mathbf{X}(\mathbf{a}, u) \quad \text{with} \quad \mathbf{a} \in \mathbb{R}^p \quad \forall u \in \mathbb{R}\}$$

- Parametric surface ($n = 3$)

$$S = \{\mathbf{X} \in \mathbb{R}^3 \,|\, \mathbf{X}(\mathbf{a}, \mathbf{u}) \quad \text{with} \quad \mathbf{a} \in \mathbb{R}^p \quad \forall \mathbf{u} \in \mathbb{R}^2\} .$$

In order to minimize the performance indices (2.20)–(2.21) with curve and surface fitting, not only do the model parameters **a** have to be determined but also the minimum distance points $\{\mathbf{X}'_i\}_{i=1}^m$ on the model feature S from each given point $\{\mathbf{X}_i\}_{i=1}^m$. With the *total method* [74], the model parameters **a** and the minimum distance points $\{\mathbf{X}'_i\}_{i=1}^m$ can be determined simultaneously

$$\min_{\mathbf{a}\in\mathbb{R}^p, \{\mathbf{X}'_i\}_{i=1}^m \in S} \sigma_0^2 \left(\{\mathbf{X}'_i(\mathbf{a})\}_{i=1}^m\right) . \tag{2.22}$$

They are determined alternately using the *variable-separation method* [44], [25] in a nested iteration scheme

$$\min_{\mathbf{a}\in\mathbb{R}^p} \min_{\{\mathbf{X}'_i\}_{i=1}^m \in S} \sigma_0^2 \left(\{\mathbf{X}'_i(\mathbf{a})\}_{i=1}^m\right) . \tag{2.23}$$

Four combinations exist between the performance indices (2.20)–(2.21) and the iteration schemes (2.22)–(2.23) as shown in Table 2.3. One of the four combinations results in a clearly underdetermined linear equation system for iteration with both implicit and parametric features. The Algorithms II and III for implicit features are described in detail in Chap. 3 and the Algorithms I–III for parametric features described in Chap. 4.

Although Algorithm I could be realized for the implicit features, it is not handled in this work because of the large memory space usage and the high computing

Table 2.3. Algorithms for the orthogonal distance fitting of curves and surfaces

Approach	Distance-based algorithm	Coordinate-based algorithm
Total method	Underdetermined, $\min_{\mathbf{a}\in\mathbb{R}^p, \{\mathbf{X}'_i\}_{i=1}^m \in S} \Vert\mathbf{P}\mathbf{d}\Vert^2$	Algorithm I, $\min_{\mathbf{a}\in\mathbb{R}^p, \{\mathbf{X}'_i\}_{i=1}^m \in S} \Vert\mathbf{P}(\mathbf{X}-\mathbf{X}')\Vert^2$
Variable-separation method	Algorithm II, $\min_{\mathbf{a}\in\mathbb{R}^p} \min_{\{\mathbf{X}'_i\}_{i=1}^m \in S} \Vert\mathbf{P}\mathbf{d}\Vert^2$	Algorithm III, $\min_{\mathbf{a}\in\mathbb{R}^p} \min_{\{\mathbf{X}'_i\}_{i=1}^m \in S} \Vert\mathbf{P}(\mathbf{X}-\mathbf{X}')\Vert^2$

Table 2.4. Conditions of Algorithm I for orthogonal distance fitting of surfaces ($n = 3$)

Conditions of Algorithm I	Parametric surface, $\mathbf{X}(\mathbf{a}, \mathbf{u})$	Implicit surface, $F(\mathbf{a}, \mathbf{X}) = 0$
No. of unknowns	$p + 2m \quad (\mathbf{a}, \{\mathbf{u}_i\}_{i=1}^m)$	$p + 3m \quad (\mathbf{a}, \{\mathbf{X}'_i\}_{i=1}^m)$
No. of equations	$3m$	$3m$
No. of constraints	—	$m \quad (\{F(\mathbf{a}, \mathbf{X}'_i) = 0\}_{i=1}^m)$
Solvability	$3m \geq p + 2m$	$4m \geq p + 3m$
Storage space	$O(3m(p+2m)) \approx O(6m^2)$	$O(4m(p+3m)) \approx O(12m^2)$
Computing cost	$O(3m(p+2m)^2) \approx O(12m^3)$	$O(4m(p+3m)^2) \approx O(36m^3)$

cost involved. In addition, a poor convergence performance of Algorithm I on the implicit features is deduced from our experience in the known Algorithm I for the parametric features [74] (see Sect. 4.2.1). The convergence performance of Algorithm I for the parametric features is poor if the initial model parameter values are not close enough to the final estimation values. Table 2.4 compares the conditions for solving the ODF problems of implicit/parametric surfaces using Algorithm I. In order to force the corresponding points $\{\mathbf{X}'_i\}_{i=1}^m$ to the given points $\{\mathbf{X}_i\}_{i=1}^m$ to be membership points of the implicit feature $F(\mathbf{a}, \mathbf{X}) = 0$, the additional constraints of $\{F(\mathbf{a}, \mathbf{X}'_i) = 0\}_{i=1}^m$ are applied. On the other hand, the corresponding points $\{\mathbf{X}(\mathbf{a}, \mathbf{u}_i)\}_{i=1}^m$ are spontaneously membership points of the parametric feature $\mathbf{X}(\mathbf{a}, \mathbf{u})$ without being forced by any additional constraint. From this fact, the convergence performance of Algorithm I for the implicit features would be worse than that for the parametric features.

2.2.4 Template Matching

Matching a template of known shape and size to a set of given points in space (*template matching,* Fig. 2.1b) is a frequent task of pattern recognition and computer/machine vision applications. Template matching can be regarded as a special case of model fitting, which determines the rigid body motion (the position parameters $\mathbf{a}_p$ and the rotation parameters $\mathbf{a}_r$) of the model feature with given (fixed) form parameters $\mathbf{a}_g$.

There are two ways of solving template matching problems using the model fitting algorithms. The straight one is to simply omit all columns of the Jacobian matrix concerning the form parameters $\mathbf{a}_g$ in the linear equation system for updating the model parameters (Sect. 2.3.1 and Sect. 2.3.2). Here, the form parameters are ignored and thus not updated whilst the position and the rotation parameters are updated. As the result, the form parameters remain unchanged at the given initial values. The second way is to apply the additional constraints of $\{a_i - a_{i,\text{given}} = 0\}_{i=1}^l$ with very large values of constraint weighting to the model fitting. The additional constraints force the form parameters $\{a_i\}_{i=1}^{l}$ to hold the given values $\{a_{i,\text{given}}\}_{i=1}^{l}$ (model fitting with parameter constraints, Sect. 2.3.3). This work gives some examples of model fitting with parameter constraints (see Fig. 2.3, Sect. 3.3.2, and Fig. 5.4).

2.3 Orthogonal Distance Fitting Algorithms

As demonstrated in the previous sections, ODF problems have a variety of combinations of model features, error definitions, algorithmic approaches and constraints. Although it appears to be a highly complicated task, this work intends to describe the generally applicable algorithms for such ODF problems in a way which is full of variety. In order to tackle the task in a well-organized and easily understandable manner, the very basic aspects of ODF are handled first, namely, the performance indices (2.20) and (2.21). Their minimization is the ultimate goal of ODF. In this section, two computational frameworks are proposed, the distance-based algorithm and the coordinate-based algorithm, for solving general ODF problems by minimizing the performance indices (2.20) and (2.21), respectively. The general implementation of the two algorithms in combination with the iteration schemes (2.22)–(2.23) (see Table 2.3) for general implicit and parametric curves/surfaces in space is given in Chap. 3 (Algorithms II and III for implicit features) and Chap. 4 (Algorithms I–III for parametric features).

2.3.1 Distance-Based Algorithm

The first order necessary condition for a minimum of the performance index (2.20) as a function of the parameters **a** is

$$\left(\frac{\partial}{\partial \mathbf{a}}\sigma_0^2\right)^{\mathrm{T}} = 2\mathbf{J}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}\mathbf{d} = \mathbf{0}\,, \qquad \text{where} \qquad \mathbf{J} = \frac{\partial \mathbf{d}}{\partial \mathbf{a}}\,. \tag{2.24}$$

The equation (2.24) can be iteratively solved for **a** using the Newton method

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}\mathbf{J} + \mathbf{H}\mathbf{P}^{\mathrm{T}}\mathbf{P}\mathbf{d}\right)\Delta\mathbf{a} = -\mathbf{J}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}\mathbf{d}\,, \quad \text{where} \quad \mathbf{H} = \frac{\partial^2 \mathbf{d}}{\partial \mathbf{a}^2}\,, \tag{2.25}$$

or, more conveniently, using the Gauss-Newton method ignoring the second derivatives term in (2.25)

$$\mathbf{J}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}\mathbf{J}\big|_k\,\Delta\mathbf{a} = -\mathbf{J}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}\mathbf{d}\big|_k\,, \qquad \mathbf{a}_{k+1} = \mathbf{a}_k + \alpha\Delta\mathbf{a}\,. \tag{2.26}$$

In this work, the parameters **a** are iteratively estimated using the direct form of the Gauss normal equation (2.26)

$$\mathbf{P}\mathbf{J}|_k\,\Delta\mathbf{a} = -\mathbf{P}\mathbf{d}|_k\,, \qquad \mathbf{a}_{k+1} = \mathbf{a}_k + \alpha\Delta\mathbf{a}\,, \tag{2.27}$$

with break conditions for the iteration (2.27)

$$\left\|\mathbf{J}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}\mathbf{d}\right\| \approx 0 \qquad \text{or} \qquad \|\Delta\mathbf{a}\| \approx 0 \qquad \text{or} \qquad \sigma_0^2\big|_k - \sigma_0^2\big|_{k+1} \approx 0\,.$$

The first break condition is equivalent to the condition (2.24) and denotes that the error distance vector and its gradient vectors in the linear equation system (2.27) should be orthogonal. If this condition is satisfied, practically no more parameter updates take place ($\|\Delta\mathbf{a}\| \approx 0$) and no improvement is made on the performance index σ_0^2.

The following unfolded form of the linear equation system (2.27) is helpful when implementing the algorithm:

$$\mathbf{P}\underbrace{\begin{pmatrix}\mathbf{J}_{d_1,\mathbf{a}}\\ \vdots \\ \mathbf{J}_{d_m,\mathbf{a}}\end{pmatrix}}_{m\times p}\Delta\mathbf{a} = -\mathbf{P}\underbrace{\begin{pmatrix}d_1\\ \vdots \\ d_m\end{pmatrix}}_{m\times 1} \qquad \text{with} \qquad \mathbf{J}_{d_i,\mathbf{a}} = \frac{\partial d_i}{\partial \mathbf{a}} \,. \tag{2.28}$$

2.3.2 Coordinate-Based Algorithm

The first order necessary condition for a minimum of the performance index (2.21) as a function of the parameters $\mathbf{a}$ is

$$\left(\frac{\partial}{\partial \mathbf{a}}\sigma_0^2\right)^{\mathrm{T}} = -2\mathbf{J}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}(\mathbf{X}-\mathbf{X}') = \mathbf{0}\,, \quad \text{where} \quad \mathbf{J} = \frac{\partial \mathbf{X}'}{\partial \mathbf{a}}\,. \tag{2.29}$$

Whilst in the case of Algorithm III the parameter vector $\mathbf{a}$ contains only the model parameters, it also contains the location parameters $\{\mathbf{u}_i\}_{i=1}^m$ of the corresponding points $\{\mathbf{X}_i'\}_{i=1}^m$ with Algorithm I for the parametric features. In the later case, the notation of the parameter vector is changed from $\mathbf{a}$ to $\mathbf{b}$ (see Sect. 4.2.1).

The equation system (2.29) is iteratively solved for a using the Newton method

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}\mathbf{J} - \mathbf{H}\mathbf{P}^{\mathrm{T}}\mathbf{P}(\mathbf{X}-\mathbf{X}')\right)\Delta\mathbf{a} = \mathbf{J}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}(\mathbf{X}-\mathbf{X}')\,, \quad \mathbf{H} = \frac{\partial^2\mathbf{X}'}{\partial \mathbf{a}^2}\,, \tag{2.30}$$

or using the Gauss-Newton method ignoring the second derivatives term in (2.30)

$$\mathbf{P}\mathbf{J}|_k\,\Delta\mathbf{a} = \mathbf{P}(\mathbf{X}-\mathbf{X}')|_k\,, \qquad \mathbf{a}_{k+1} = \mathbf{a}_k + \alpha\Delta\mathbf{a}\,, \tag{2.31}$$

with break conditions

$$\|\mathbf{J}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}(\mathbf{X}-\mathbf{X}')\| \approx 0 \quad \text{or} \quad \|\Delta\mathbf{a}\| \approx 0 \quad \text{or} \quad \sigma_0^2\big|_k - \sigma_0^2\big|_{k+1} \approx 0\,.$$

The unfolded form of the linear equation system (2.31) appears as

$$\mathbf{P}\underbrace{\begin{pmatrix}\mathbf{J}_{\mathbf{X}_1',\mathbf{a}}\\ \vdots \\ \mathbf{J}_{\mathbf{X}_m',\mathbf{a}}\end{pmatrix}}_{nm\times p}\Delta\mathbf{a} = \mathbf{P}\underbrace{\begin{pmatrix}\mathbf{X}_1-\mathbf{X}_1'\\ \vdots \\ \mathbf{X}_m-\mathbf{X}_m'\end{pmatrix}}_{nm\times 1} \qquad \text{with} \qquad \mathbf{J}_{\mathbf{X}_i',\mathbf{a}} = \frac{\partial \mathbf{X}_i'}{\partial \mathbf{a}}\,. \tag{2.32}$$

2.3.3 Model Fitting with Parameter Constraints

If any parameter constraint (e.g. on shape, size, area, volume, position, orientation of the model feature)

$$f_{cj}(\mathbf{a}) - \mathrm{const}_j = 0\,, \qquad j = 1, \ldots, q\,, \tag{2.33}$$

or, in vector form,

$$\mathbf{f}_\mathrm{c}(\mathbf{a}) - \mathbf{const} = \mathbf{0}$$

is to be applied additionally, the following equation systems (2.34) with the large weighting values w_c is appended to the linear equation systems (2.27) or (2.31):

$$w_{cj}\left(\frac{\partial f_{cj}}{\partial \mathbf{a}}\right)\Delta\mathbf{a} = -w_{cj}\left(f_{cj}(\mathbf{a}) - \mathrm{const}_j\right), \qquad j = 1, \ldots, q\,. \tag{2.34}$$

Considering the parameter constraints (2.33), the performance indices (2.20) and (2.21) and the linear equation systems (2.27) and (2.31) can be rewritten as follows:

$$\begin{gathered}\sigma_0^2 = \begin{pmatrix}\mathbf{Pd}\\ \mathbf{W}_\mathrm{c}(\mathbf{f}_\mathrm{c}(\mathbf{a}) - \mathbf{const})\end{pmatrix}^\mathrm{T}\begin{pmatrix}\mathbf{Pd}\\ \mathbf{W}_\mathrm{c}(\mathbf{f}_\mathrm{c}(\mathbf{a}) - \mathbf{const})\end{pmatrix},\\ \underbrace{\begin{pmatrix}\mathbf{PJ}\\ \mathbf{W}_\mathrm{c}\mathbf{J}_\mathrm{c}\end{pmatrix}}_{(m+q)\times p}\Delta\mathbf{a} = -\underbrace{\begin{pmatrix}\mathbf{Pd}\\ \mathbf{W}_\mathrm{c}(\mathbf{f}_\mathrm{c}(\mathbf{a}) - \mathbf{const})\end{pmatrix}}_{(m+q)\times 1}, \quad \mathbf{J} = \frac{\partial \mathbf{d}}{\partial \mathbf{a}}, \quad \mathbf{J}_\mathrm{c} = \frac{\partial \mathbf{f}_\mathrm{c}}{\partial \mathbf{a}}\end{gathered} \tag{2.35}$$

for the distance-based algorithm, and,

$$\begin{gathered}\sigma_0^2 = \begin{pmatrix}\mathbf{P}(\mathbf{X} - \mathbf{X}')\\ \mathbf{W}_\mathrm{c}(\mathbf{f}_\mathrm{c}(\mathbf{a}) - \mathbf{const})\end{pmatrix}^\mathrm{T}\begin{pmatrix}\mathbf{P}(\mathbf{X} - \mathbf{X}')\\ \mathbf{W}_\mathrm{c}(\mathbf{f}_\mathrm{c}(\mathbf{a}) - \mathbf{const})\end{pmatrix},\\ \underbrace{\begin{pmatrix}\mathbf{PJ}\\ \mathbf{W}_\mathrm{c}\mathbf{J}_\mathrm{c}\end{pmatrix}}_{(nm+q)\times p}\Delta\mathbf{a} = \underbrace{\begin{pmatrix}\mathbf{P}(\mathbf{X} - \mathbf{X}')\\ -\mathbf{W}_\mathrm{c}(\mathbf{f}_\mathrm{c}(\mathbf{a}) - \mathbf{const})\end{pmatrix}}_{(nm+q)\times 1}, \quad \mathbf{J} = \frac{\partial \mathbf{X}'}{\partial \mathbf{a}}, \quad \mathbf{J}_\mathrm{c} = \frac{\partial \mathbf{f}_\mathrm{c}}{\partial \mathbf{a}}\end{gathered} \tag{2.36}$$

for the coordinate-based algorithm.

The model fitting with parameter constraints described above can be applied in a number of ways. One practical application of the constrained parameter estimation is template matching (Sect. 2.2.4), i.e. estimation of the position $\mathbf{a}_\mathrm{p}$ and the rotation parameters $\mathbf{a}_\mathrm{r}$ of a model feature with the given (fixed) form parameters $\mathbf{a}_\mathrm{g}$ (see Sect. 2.3.5 for an example of circle matching and [8] for fitting an ellipse with a known area). Of course, any model parameter can be fixed at the given value (see Sect. 5.3.1 for fitting a horizontally-lying ellipsoid with the constraints of $\omega = 0$ and $\varphi = 0$). Particularly with cylinder/cone fitting, the constrained parameter estimation plays an important role in determining the position of the cylinder/cone on its axis (see Sect. B.1.2, Sect. B.2.7, and Sect. B.2.8).

The most promising application of the constrained parameter estimation is the reconstruction of multiple objects from their measurement points, where certain geometric constraints (interrelations) between the object models need to be satisfied by the multiple model fitting. For example, with the reconstruction of a sphere and a plane *in contact,* the radius of the sphere and the distance from the center of the sphere to the plane must keep the same value

$$f_c(\mathbf{a}_{\text{plane}}, \mathbf{a}_{\text{sphere}}) \triangleq \left\|(\mathbf{X}_{\text{o,sphere}} - \mathbf{X}_{\text{o,plane}})^{\mathrm{T}}\mathbf{n}_{\text{plane}}\right\| - r_{\text{sphere}} = 0 \ .$$

The linear equation system (2.35) for the simultaneous fitting of multiple object models with parameter constraints appears as

$$\begin{pmatrix} (\mathbf{PJ})_1 & \mathbf{0} \\ \mathbf{0} & (\mathbf{PJ})_2 \\ \mathbf{W}_c\mathbf{J}_{c,1} & \mathbf{W}_c\mathbf{J}_{c,2} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{a}_1 \\ \Delta\mathbf{a}_2 \end{pmatrix} = - \begin{pmatrix} (\mathbf{Pd})_1 \\ (\mathbf{Pd})_2 \\ \mathbf{W}_c(\mathbf{f}_c(\mathbf{a}_1, \mathbf{a}_2) - \mathbf{const}) \end{pmatrix} ,$$

or, in a mixed form of (2.35) and (2.36)

$$\begin{pmatrix} (\mathbf{PJ})_1 & \mathbf{0} \\ \mathbf{0} & (\mathbf{PJ})_2 \\ \mathbf{W}_c\mathbf{J}_{c,1} & \mathbf{W}_c\mathbf{J}_{c,2} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{a}_1 \\ \Delta\mathbf{a}_2 \end{pmatrix} = \begin{pmatrix} -(\mathbf{Pd})_1 \\ (\mathbf{P}(\mathbf{X} - \mathbf{X}'))_2 \\ -\mathbf{W}_c(\mathbf{f}_c(\mathbf{a}_1, \mathbf{a}_2) - \mathbf{const}) \end{pmatrix} .$$

2.3.4 Parameter Test

The Jacobian matrix in (2.35) and (2.36) is decomposed by the SVD

$$\begin{pmatrix} \mathbf{PJ} \\ \mathbf{W}_c\mathbf{J}_c \end{pmatrix} = \mathbf{UWV}^{\mathrm{T}}$$

with

$$\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I} \qquad \text{and} \qquad \mathbf{W} = [\operatorname{diag}(w_1, \ldots, w_p)] \ .$$

The linear equation systems (2.35) and (2.36) can then be solved for $\Delta\mathbf{a}$. After successfully terminating the iterations (2.35) or (2.36), the Jacobian matrix together with the performance index σ_0^2 provides useful information about the quality of parameter estimations as follows:

- Covariance matrix

$$\begin{aligned} \operatorname{Cov}(\hat{\mathbf{a}}) &= \left[\begin{pmatrix} \mathbf{PJ} \\ \mathbf{W}_c\mathbf{J}_c \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \mathbf{PJ} \\ \mathbf{W}_c\mathbf{J}_c \end{pmatrix}\right]^{-1} \\ &= \mathbf{VW}^{-2}\mathbf{V}^{\mathrm{T}} \end{aligned}$$

- Parameter covariance

$$\operatorname{Cov}(\hat{a}_j, \hat{a}_k) = \sum_{i=1}^{p} \left(\frac{V_{ji}V_{ki}}{w_i^2}\right) , \quad j, k = 1, \ldots, p \tag{2.37}$$

- Variance of parameters

$$\sigma^2(\hat{a}_j) = \frac{\sigma_0^2}{m+q-p}\mathrm{Cov}(\hat{a}_j,\hat{a}_j)\,, \quad j=1,\ldots,p$$

- Correlation coefficients

$$\mathrm{Cor}(\hat{a}_j,\hat{a}_k) = \frac{\mathrm{Cov}(\hat{a}_j,\hat{a}_k)}{\sqrt{\mathrm{Cov}(\hat{a}_j,\hat{a}_j)\mathrm{Cov}(\hat{a}_k,\hat{a}_k)}}\,, \quad j,k=1,\ldots,p\,.$$

By using the above information, the reliability of the estimated parameters $\hat{\mathbf{a}}$ and the propriety of the model selection (see Sect. 1.1.3) can be tested.

The parameter covariance (2.37) with the distance-based algorithm is generally a little larger than that with the coordinate-based algorithm. If σ_0^2 is small enough, justifying the ignoring of the second derivatives term in (2.27) and (2.31), there is practically no parameter covariance difference (2.37) between the two algorithms.

2.3.5 Application to Circle and Sphere Fitting

To fit a model feature to a set of given points $\{\mathbf{X}_i\}_{i=1}^m$ using the distance-based algorithm (2.28), the linear equation system (2.28) must be provided with not only the minimum distances $\{d_i\}_{i=1}^m$ between the model feature and each given point but also the Jacobian matrices $\{\mathbf{J}_{d_i,\mathbf{a}}\}_{i=1}^m$ of each d_i. For model fitting using the coordinate-based algorithm (2.32), the minimum distance points $\{\mathbf{X}'_i\}_{i=1}^m$ and their Jacobian matrices $\{\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}}\}_{i=1}^m$ are required. The next two chapters deal with the general implementation of the two algorithms (2.28) and (2.32) for implicit (Chap. 3) and parametric (Chap. 4) curves/surfaces. Using these, the information $\{d_i\}_{i=1}^m$, $\{\mathbf{X}'_i\}_{i=1}^m$, $\{\mathbf{J}_{d_i,\mathbf{a}}\}_{i=1}^m$, and $\{\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}}\}_{i=1}^m$ necessary for the two algorithms is obtained in a highly general manner for implicit/parametric curves/surfaces in 2-D/3-D space.

In this section, before going into the general implementation, the specific implementation of the algorithms (2.28) and (2.32) to circle/sphere fitting is shown as an introductory example. Because a circle/sphere has no rotation parameter and permits the minimum distance d_i and the minimum distance point $\mathbf{X}'_i$ to be described in closed form, the implementation of the two algorithms to circle/sphere fitting is relatively straightforward. A circle/sphere with radius r and center at $\mathbf{X}_o$ in n-dimensional space ($n \geq 2$) can be described in implicit form (Fig. 2.2)

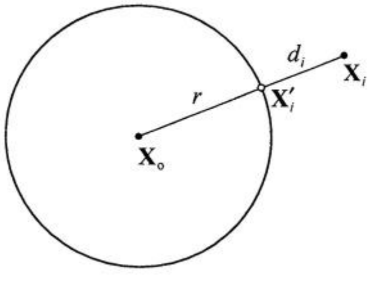


Fig. 2.2. A circle/sphere with radius r and center at $\mathbf{X}_o$ in n-dimensional space

$$F(\mathbf{a},\mathbf{X}) \triangleq \|\mathbf{X}-\mathbf{X}_o\|^2 - r^2 = 0 \qquad \text{with} \qquad \mathbf{a}^T \triangleq (r, \mathbf{X}_o^T) \,. \tag{2.38}$$

The minimum distance point $\mathbf{X}_i'$ on (2.38) from a given point $\mathbf{X}_i$ $(\neq \mathbf{X}_o)$ is

$$\mathbf{X}_i' = \mathbf{X}_o + r\frac{\mathbf{X}_i - \mathbf{X}_o}{\|\mathbf{X}_i - \mathbf{X}_o\|} \,, \qquad i = 1, \ldots, m \,. \tag{2.39}$$

And, the (signed) distance d_i between the two points $\mathbf{X}_i$ and $\mathbf{X}_i'$ is

$$d_i = \|\mathbf{X}_i - \mathbf{X}_o\| - r \,. \tag{2.40}$$

The Jacobian matrix $\mathbf{J}_{d_i,\mathbf{a}}$ of d_i can be directly derived from (2.40)

$$\begin{aligned}\mathbf{J}_{d_i,\mathbf{a}} &= \frac{\partial d_i}{\partial \mathbf{a}} \\ &= \frac{\partial}{\partial \mathbf{a}}\left[\|\mathbf{X}_i - \mathbf{X}_o\| - r\right] \\ &= -\frac{(\mathbf{X}_i - \mathbf{X}_o)^T}{\|\mathbf{X}_i - \mathbf{X}_o\|}\frac{\partial \mathbf{X}_o}{\partial \mathbf{a}} - \frac{\partial r}{\partial \mathbf{a}} \,,\end{aligned}$$

where

$$\frac{\partial r}{\partial \mathbf{a}} = \begin{pmatrix} 1 & | & \mathbf{0}^T \end{pmatrix} \qquad \text{and} \qquad \frac{\partial \mathbf{X}_o}{\partial \mathbf{a}} = \begin{pmatrix} \mathbf{0} & | & \mathbf{I} \end{pmatrix} \,.$$

Then, with sphere fitting ($n = 3$), the linear equation system (2.28) appears as

$$\underbrace{\mathbf{P}\begin{pmatrix} \mathbf{J}_{d_1,r} & \mathbf{J}_{d_1,X_o} & \mathbf{J}_{d_1,Y_o} & \mathbf{J}_{d_1,Z_o} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{J}_{d_m,r} & \mathbf{J}_{d_m,X_o} & \mathbf{J}_{d_m,Y_o} & \mathbf{J}_{d_m,Z_o} \end{pmatrix}}_{m\times 4} \begin{pmatrix} \Delta r \\ \Delta X_o \\ \Delta Y_o \\ \Delta Z_o \end{pmatrix} = -\mathbf{P}\underbrace{\begin{pmatrix} d_1 \\ \vdots \\ d_m \end{pmatrix}}_{m\times 1} \,. \tag{2.41}$$

In a similar way, the Jacobian matrix $\mathbf{J}_{\mathbf{X}_i',\mathbf{a}}$ of the minimum distance point $\mathbf{X}_i'$ can be directly derived from (2.39)

$$\begin{aligned}\mathbf{J}_{\mathbf{X}_i',\mathbf{a}} &= \frac{\partial \mathbf{X}_i'}{\partial \mathbf{a}} \\ &= \frac{\partial}{\partial \mathbf{a}}\left[\mathbf{X}_o + r\frac{\mathbf{X}_i - \mathbf{X}_o}{\|\mathbf{X}_i - \mathbf{X}_o\|}\right] \\ &= \frac{\partial \mathbf{X}_o}{\partial \mathbf{a}} + \frac{\mathbf{X}_i - \mathbf{X}_o}{\|\mathbf{X}_i - \mathbf{X}_o\|}\frac{\partial r}{\partial \mathbf{a}} - \frac{r}{\|\mathbf{X}_i - \mathbf{X}_o\|}\mathbf{H}_i\frac{\partial \mathbf{X}_o}{\partial \mathbf{a}} \,,\end{aligned}$$

where

$$\mathbf{H}_i = \mathbf{I} - \frac{(\mathbf{X}_i - \mathbf{X}_o)(\mathbf{X}_i - \mathbf{X}_o)^T}{\|\mathbf{X}_i - \mathbf{X}_o\|^2} \,.$$

With sphere fitting ($n = 3$), the linear equation system (2.32) appears as

$$\mathbf{P}\underbrace{\begin{pmatrix} \mathbf{J}_{X'_1,r} & \mathbf{J}_{X'_1,X_o} & \mathbf{J}_{X'_1,Y_o} & \mathbf{J}_{X'_1,Z_o} \\ \mathbf{J}_{Y'_1,r} & \mathbf{J}_{Y'_1,X_o} & \mathbf{J}_{Y'_1,Y_o} & \mathbf{J}_{Y'_1,Z_o} \\ \mathbf{J}_{Z'_1,r} & \mathbf{J}_{Z'_1,X_o} & \mathbf{J}_{Z'_1,Y_o} & \mathbf{J}_{Z'_1,Z_o} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{J}_{X'_m,r} & \mathbf{J}_{X'_m,X_o} & \mathbf{J}_{X'_m,Y_o} & \mathbf{J}_{X'_m,Z_o} \\ \mathbf{J}_{Y'_m,r} & \mathbf{J}_{Y'_m,X_o} & \mathbf{J}_{Y'_m,Y_o} & \mathbf{J}_{Y'_m,Z_o} \\ \mathbf{J}_{Z'_m,r} & \mathbf{J}_{Z'_m,X_o} & \mathbf{J}_{Z'_m,Y_o} & \mathbf{J}_{Z'_m,Z_o} \end{pmatrix}}_{3m\times 4} \begin{pmatrix} \Delta r \\ \Delta X_o \\ \Delta Y_o \\ \Delta Z_o \end{pmatrix} = \mathbf{P}\underbrace{\begin{pmatrix} X_1 - X'_1 \\ Y_1 - Y'_1 \\ Z_1 - Z'_1 \\ \vdots \\ X_m - X'_m \\ Y_m - Y'_m \\ Z_m - Z'_m \end{pmatrix}}_{3m\times 1} . \tag{2.42}$$

To give an experimental example, a circle is fitted to the six points in Table 2.5. As the initial parameter values starting the Gauss-Newton iteration (2.41) and (2.42), the mass center (2.5) and the rms central distance of the set of given points are used

$$\mathbf{X}_{o,0} = \bar{\mathbf{X}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{X}_i \qquad \text{and} \qquad r_0 = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\|\mathbf{X}_i - \bar{\mathbf{X}}\|^2} .$$

Table 2.5. Six coordinate pairs ($n = 2$) [34]

X	1	2	5	7	9	3
Y	7	6	8	7	5	7

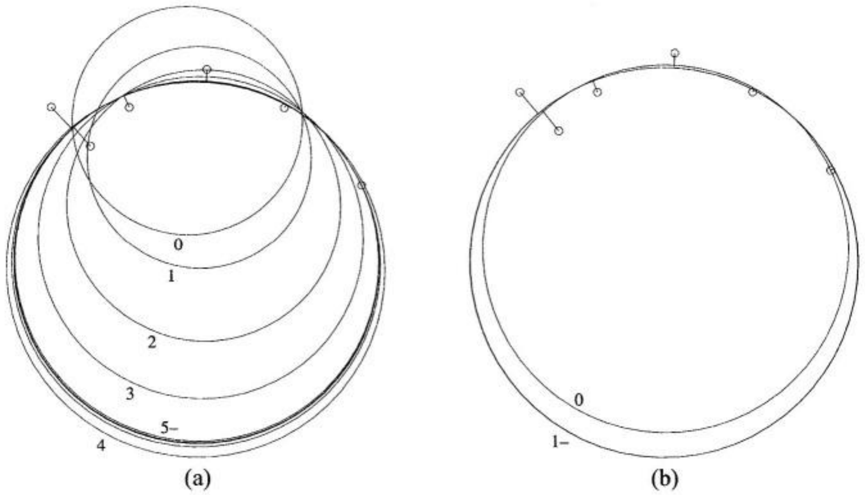


Fig. 2.3. Orthogonal distance fitting of a circle to the set of points in Table 2.5: (a) Without any constraint (model fitting); (b) With the constraint of $r - 5 = 0$ (template matching)

The distance-based algorithm (2.41) terminates the circle fitting after ten iteration cycles for $\|\Delta\mathbf{a}\| = 3.2 \cdot 10^{-6}$ and the coordinate-based algorithm (2.42) terminates after 13 iteration cycles for $\|\Delta\mathbf{a}\| = 9.4 \cdot 10^{-7}$ (Table 2.6). A strong anti-correlation (≈ -1.0) between the radius r and the center coordinate Y_o of the resultant circle can be seen (Table 2.7) which is caused by the distribution of the set of given points. In other words, given the set of points distributed over the upper side of the resultant circle, the center coordinate Y_o increases/decreases as the radius r differentially decreases/increases (Fig. 2.3a).

In order to give an example of template matching, a circle with the radius of $r = 5$ is fitted to the same set of points by applying the additional constraint of $r - 5 = 0$ with the constraint weighting value of $w_c = 1.0 \cdot 10^6$ (see Sect. 2.3.3). The distance-based algorithm and the coordinate-based algorithm terminate after a further four and five iteration cycles respectively for $\|\Delta\mathbf{a}\| = 1.8 \cdot 10^{-6}$ and $\|\Delta\mathbf{a}\| = 6.9 \cdot 10^{-7}$ (Fig. 2.3b). As expected, the variance of the radius r and its correlations with other parameters disappear (Table 2.8 and Table 2.9).

Table 2.6. Results of orthogonal distance fitting to the points in Table 2.5

$\sigma_0 = 1.1080$	r	X_o	Y_o
Parameter $\hat{\mathbf{a}}$	4.7142	4.7398	2.9835
$\sigma(\hat{\mathbf{a}})$ II	1.2243	0.4776	1.5429
III	1.1422	0.4628	1.4331

II: distance-based algorithm, III: coordinate-based algorithm.

Table 2.7. Correlation coefficients of the circle parameters in Table 2.6

Distance-based algorithm				Coordinate-based algorithm			
Cor($\hat{\mathbf{a}}$)	r	X_o	Y_o	Cor($\hat{\mathbf{a}}$)	r	X_o	Y_o
r	1.00			r	1.00		
X_o	−0.37	1.00		X_o	−0.31	1.00	
Y_o	−0.98	0.39	1.00	Y_o	−0.97	0.34	1.00

Table 2.8. Results of orthogonal distance fitting with the constraint of $r - 5 = 0$ to the set of points in Table 2.5

$\sigma_0 = 1.1151$	r	X_o	Y_o
Parameter $\hat{\mathbf{a}}$	5.0000	4.6917	2.6320
$\sigma(\hat{\mathbf{a}})$ II	0.0000	0.4064	0.2795
III	0.0000	0.4023	0.2785

II: distance-based algorithm, III: coordinate-based algorithm.

Table 2.9. Correlation coefficients of the circle parameters in Table 2.8

Distance-based algorithm				Coordinate-based algorithm			
Cor($\hat{\mathbf{a}}$)	r	X_o	Y_o	Cor($\hat{\mathbf{a}}$)	r	X_o	Y_o
r	1.00			r	1.00		
X_o	−0.00	1.00		X_o	−0.00	1.00	
Y_o	−0.00	0.15	1.00	Y_o	−0.00	0.14	1.00

3. Orthogonal Distance Fitting of Implicit Curves and Surfaces

This chapter describes in detail the general implementation of the two ODF Algorithms II and III (*variable-separation method,* see Table 2.3 and Sect. 2.3) on implicit curves and surfaces $F(\mathbf{a}, \mathbf{X}) = 0$ in space. With the implementations, compared with the known Algorithm II [44], [75] (Table 1.2), the model parameters $\mathbf{a}$ are grouped and simultaneously estimated in terms of form $\mathbf{a}_g$, position $\mathbf{a}_p$, and rotation parameters $\mathbf{a}_r$, which is a highly desirable algorithmic feature for applications (Sect. 1.1.2, Fig. 3.1). With the variable-separation method, the model parameters $\mathbf{a}$ and the minimum distance points $\{\mathbf{X}'_i\}_{i=1}^m$ on the model feature $F(\mathbf{a}, \mathbf{X}) = 0$ from each given point $\{\mathbf{X}_i\}_{i=1}^m$ are alternately determined in a nested iteration scheme

$$\min_{\mathbf{a}\in\mathbb{R}^p} \min_{\{\mathbf{X}'_i\}_{i=1}^m \in S} \sigma_0^2\left(\{\mathbf{X}'_i(\mathbf{a})\}_{i=1}^m\right) . \tag{3.1}$$

The inner iteration, carried out at each outer iteration cycle, finds the minimum distance points on the current model feature (Sect. 3.1). The outer iteration cycle updates the model parameters (Sect. 3.2). Once appropriate break conditions are satisfied, e.g. when no more significant improvement on the performance indices (2.20) and (2.21) can be gained through parameter updating, the outer iteration terminates.

Besides the parameter grouping of $\mathbf{a}^T = (\mathbf{a}_g^T, \mathbf{a}_p^T, \mathbf{a}_r^T)$, the novelty of the implementation of the two ODF Algorithms II and III on implicit features is that the necessary information for the Gauss-Newton iteration (2.28) and (2.32) is obtained in a very general manner. The minimum distance information $\{d_i\}_{i=1}^m$, $\{\mathbf{X}'_i\}_{i=1}^m$, and their Jacobian matrices $\{\mathbf{J}_{d_i,\mathbf{a}}\}_{i=1}^m$, $\{\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}}\}_{i=1}^m$ are obtained from the standard feature equation $f(\mathbf{a}_g, \mathbf{x}) = 0$ (1.1) defined in a model coordinate frame xyz and from the coordinate transformation equation $\mathbf{X} = \mathbf{R}^{-1}\mathbf{x} + \mathbf{X}_o$ (1.3). In order to apply the algorithms to a new implicit feature, merely the provision of the first and the second derivatives of the standard feature equation (1.1) is required which usually has only a few form parameters. Functional interpretations and treatment of the position/rotation parameters concerning the coordinate transformation (1.3) are the same for all implicit features.

Section 3.1 describes two generally applicable algorithms for finding the minimum distance point on an implicit feature from a given point. Section 3.2 describes the general implementation of the Gauss-Newton iteration (2.28) and (2.32) for implicit features. Various fitting examples are given in Sect. 3.3.

3.1 Minimum Distance Point

The time-consuming part of the variable-separation method (3.1) is the inner iteration which finds the minimum distance points $\{\mathbf{X}'_i\}_{i=1}^m$ on a general model feature from each given point $\{\mathbf{X}_i\}_{i=1}^m$. For a given point $\mathbf{x}_i$ in xyz frame (Fig. 3.1)

$$\mathbf{x}_i = \mathbf{R}(\mathbf{X}_i - \mathbf{X}_o) \ , \tag{3.2}$$

the minimum distance point $\mathbf{x}'_i$ is determined on the standard model feature (1.1) defined in xyz frame. Then the minimum distance point $\mathbf{X}'_i$ in XYZ frame from the given point $\mathbf{X}_i$ can be obtained through the backward transformation of $\mathbf{x}'_i$ into XYZ frame as below

$$\mathbf{X}'_i = \mathbf{R}^{-1}\mathbf{x}'_i + \mathbf{X}_o \ .$$

For some implicit features such as a circle, sphere, cylinder, cone and torus, the minimum distance point $\mathbf{x}'_i$ can be determined in closed form at low computing cost. However, in general, $\mathbf{x}'_i$ must be found iteratively to satisfy appropriate minimum distance conditions. Two generally applicable algorithms are described for finding the minimum distance point $\mathbf{x}'_i$ in xyz frame. The first algorithm is derived from the general properties of the minimum distance point. The second algorithm is based on a constrained minimization method, the method of Lagrangian multipliers [33]. The minimum distance point is marked in the following subsections with prime, as $\mathbf{X}'_i$ in XYZ frame and as $\mathbf{x}'_i$ in xyz frame, once it has been determined and is ready for use. Otherwise, it is notated with the general coordinate vector $\mathbf{X}$ or $\mathbf{x}$.

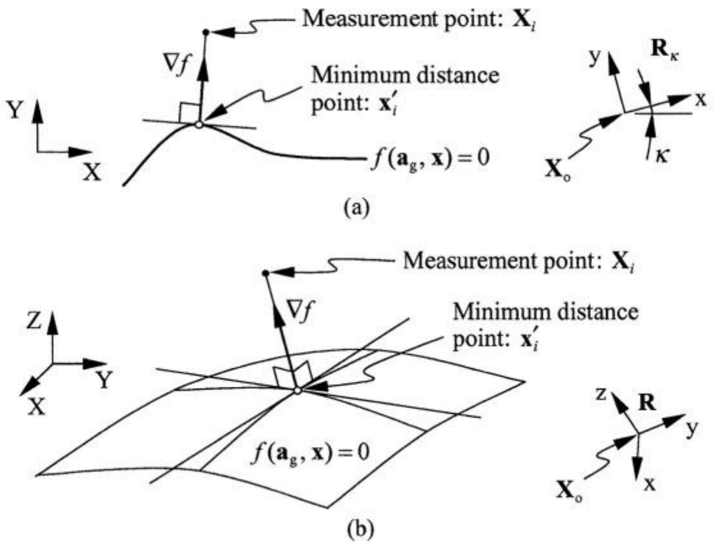

Measurement point: $\mathbf{X}_i$
∇f
Minimum distance point: $\mathbf{x}'_i$
$f(\mathbf{a}_g, \mathbf{x}) = 0$
Y
X
y
x
$\mathbf{R}_\kappa$
κ
$\mathbf{X}_o$
(a)
Measurement point: $\mathbf{X}_i$
∇f
Minimum distance point: $\mathbf{x}'_i$
$f(\mathbf{a}_g, \mathbf{x}) = 0$
Z
Y
X
z
y
x
$\mathbf{R}$
$\mathbf{X}_o$
(b)

Fig. 3.1. Implicit features and the minimum distance point $\mathbf{x}'_i$ in xyz frame from the given point $\mathbf{X}_i$ in XYZ frame: (a) Plane curve; (b) Surface

3.1.1 Generalized Newton Method

A necessary condition for the minimum distance point **x** on the implicit feature (1.1) from the given point $\mathbf{x}_i$ is that the connecting line of $\mathbf{x}_i$ with **x** should be parallel to the feature normal ∇f at **x** (Fig. 3.1)

$$\nabla f \times (\mathbf{x}_i - \mathbf{x}) = \mathbf{0}\ , \qquad \text{where} \qquad \nabla \triangleq (\partial/\partial x,\ \partial/\partial y,\ \partial/\partial z)^{\mathrm{T}}\ . \tag{3.3}$$

Equation (3.3) is equivalent to

$$\nabla_x f/(x_i - x) = \nabla_y f/(y_i - y) = \nabla_z f/(z_i - z)\ ,$$

and only two of three equation rows of (3.3) are independent. Equation (3.3) describes a space curve as the intersection of two implicit surfaces defined in xyz frame by two independent equation rows of (3.3). It is found that (3.3) generally satisfies the parallel condition of the vector $(\mathbf{x}_i - \mathbf{x})$ to the feature normal ∇f of the iso-feature of the standard model feature $f(\mathbf{a}_{\mathrm{g}}, \mathbf{x}) = 0$ (1.1)

$$f(\mathbf{a}_{\mathrm{g}}, \mathbf{x}) - \text{const} = 0\ . \tag{3.4}$$

As the constant in (3.4) varies continuously, the trajectories of the points lying on (3.4) and satisfying (3.3) draw a curve described by (3.3). This curve (3.3) is called the *centrortho-curve* about the point $\mathbf{x}_i$ to the iso-features (3.4). One aim of the work was to find the intersection of the centrortho-curve (3.3) with the implicit feature (1.1). In other words, the objective is to find the point **x** that simultaneously satisfies (1.1) and (3.3) (minimum distance condition, see Fig. 3.2 for the case of a plane curve, an ellipse)

$$\mathbf{f}(\mathbf{a}_{\mathrm{g}}, \mathbf{x}_i, \mathbf{x}) \triangleq \begin{pmatrix} f \\ \nabla f \times (\mathbf{x}_i - \mathbf{x}) \end{pmatrix} = \mathbf{0} \quad \text{with} \quad \mathbf{x}_i = \mathbf{R}(\mathbf{X}_i - \mathbf{X}_{\mathrm{o}})\ . \tag{3.5}$$

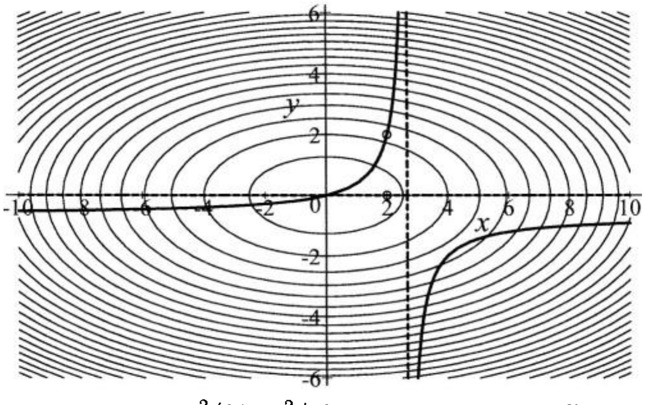


Fig. 3.2. Isocurves $x^2/64 + y^2/16 - \text{const} = 0$ (const > 0) and the trajectories (centrortho curves) $(3x - 8)(3y + 2) + 16 = 0$ (thick line) and $(3x - 8)y = 0$ (broken line) of the minimum distance points on the isocurves from the point (2, 2) and (2, 0), respectively

The minimum distance condition (3.5) governs the variational behavior of the minimum distance point in xyz frame relative to the differential change of the model parameters $\mathbf{a}$ and plays an important role in the coordinate-based algorithm described in Sect. 3.2.2.

Equation (3.5) is solved for $\mathbf{x}$ by using the generalized Newton method starting from the initial point of $\mathbf{x}_0 = \mathbf{x}_i$ (Fig. 3.3) (how to compute the matrix $\partial\mathbf{f}/\partial\mathbf{x}$ is shown in Sect. 3.2.2).

$$\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_k \Delta\mathbf{x} = -\mathbf{f}(\mathbf{x})|_k \,, \qquad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\Delta\mathbf{x} \,. \tag{3.6}$$

If necessary, especially in order to prevent the point update $\Delta\mathbf{x}$ from heading for the divergence zone where an implicit feature has a very high local curvature, the on-the-feature condition in (3.5) is underweighted as shown below:

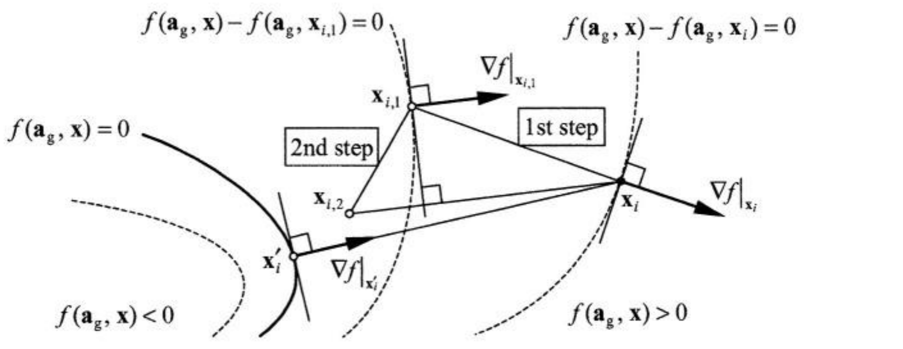
$$\mathbf{f}(\mathbf{a}_g, \mathbf{x}_i, \mathbf{x}) \triangleq \begin{pmatrix} wf \\ \nabla f \times (\mathbf{x}_i - \mathbf{x}) \end{pmatrix} = \mathbf{0} \quad \text{with} \quad 0 < w \leq 1 \,.$$

In the first iteration cycle with the starting point of $\mathbf{x}_0 = \mathbf{x}_i$, the linear equation system (3.6) and its solution (i.e., the first moving step) are equivalent to

$$\left.\begin{pmatrix} \nabla f \cdot \Delta\mathbf{x} \\ \nabla f \times \Delta\mathbf{x} \end{pmatrix}\right|_{\mathbf{x}=\mathbf{x}_i} = \left.\begin{pmatrix} -f \\ \mathbf{0} \end{pmatrix}\right|_{\mathbf{x}=\mathbf{x}_i} \quad \text{and} \quad \Delta\mathbf{x} = \left.\frac{-f}{\|\nabla f\|^2}\nabla f\right|_{\mathbf{x}=\mathbf{x}_i} \,.$$

It should be noted that the first moving step is the negative of the normalized algebraic error vector shown in (1.8) which has been described in literature [71], [76] as a good approximation of the minimum distance error (geometric error) vector.

Also to be noted is the singular point of an implicit feature, such as the vertex of an elliptic cone, where the feature normal vanishes as $\nabla f = \mathbf{0}$. Because a singular point on an implicit feature (1.1) satisfies the minimum distance condition (3.5),



$f(\mathbf{a}_g, \mathbf{x}) - f(\mathbf{a}_g, \mathbf{x}_{i,1}) = 0$
$f(\mathbf{a}_g, \mathbf{x}) - f(\mathbf{a}_g, \mathbf{x}_i) = 0$
$\nabla f|_{\mathbf{x}_{i,1}}$
$\mathbf{x}_{i,1}$
$f(\mathbf{a}_g, \mathbf{x}) = 0$
1st step
2nd step
$\mathbf{x}_{i,2}$
$\mathbf{x}_i$
$\nabla f|_{\mathbf{x}_i}$
$\mathbf{x}_i'$
$\nabla f|_{\mathbf{x}_i'}$
$f(\mathbf{a}_g, \mathbf{x}) < 0$
$f(\mathbf{a}_g, \mathbf{x}) > 0$

Fig. 3.3. Iterative search (generalized Newton method) for the minimum distance point $\mathbf{x}_i'$ on $f(\mathbf{a}_g, \mathbf{x}) = 0$ from the given point $\mathbf{x}_i$. The points $\mathbf{x}_{i,1}$ and $\mathbf{x}_{i,2}$ are the first and the second approximation of $\mathbf{x}_i'$, respectively

even though the singular point might not be a minimum distance point, the linear equation system (3.6) at a singular point results in $\Delta\mathbf{x} = \mathbf{0}$. In other words, the iteration (3.6) is caught by a singular point. Thus, when the feature normal ∇f vanishes with the termination of the iteration (3.6), the resultant point will be perturbed by adding small artifact errors and the iteration (3.6) restarted (random walking technique), in order to give a chance for the iteration (3.6) to escape from the singular point. Nevertheless, if the iteration (3.6) repeatedly falls into a singular point, this may be accepted as a minimum distance point.

For plane curves on the xy-plane of $z = 0$, the second and the third row of (3.5), which have coordinate z terms in the cross product and have been automatically satisfied, are simply ignored. The fourth row of (3.5) without coordinate z terms in the cross product is nothing but the orthogonality condition of the error vector $(\mathbf{x}_i - \mathbf{x})$ to the isocurve tangent in xy-plane [6], [7]. From this fact and for the sake of a common program module for curves and surfaces, the second and fourth row of (3.5) are interchanged. Then, for the case of plane curves, only the first two rows of the modified (3.5) are activated in the program module (see Sect. A.1 and Fig. 3.2 for the case of an ellipse).

3.1.2 Method of Lagrangian Multipliers

As an alternative to the generalized Newton method, the minimum distance point $\mathbf{x}$ can also be found using a constrained minimization method. The aim was to find the nearest point $\mathbf{x}$ on the model feature $f(\mathbf{a}_g, \mathbf{x}) = 0$ from the given point $\mathbf{x}_i$ and is formulated as shown below:

$$\min_{\mathbf{x}} (\mathbf{x}_i - \mathbf{x})^{\mathrm{T}}(\mathbf{x}_i - \mathbf{x})$$

subject to

$$f(\mathbf{x}) = 0 \; .$$

This problem is solved by minimizing the Lagrangian function below (method of Lagrangian multipliers [33]):

$$L(\lambda, \mathbf{x}) \triangleq (\mathbf{x}_i - \mathbf{x})^{\mathrm{T}}(\mathbf{x}_i - \mathbf{x}) + \lambda f \; . \tag{3.7}$$

The first order necessary condition for a minimum of (3.7) is

$$\begin{pmatrix} \nabla L \\ \frac{\partial L}{\partial \lambda} \end{pmatrix} = \begin{pmatrix} -2(\mathbf{x}_i - \mathbf{x}) + \lambda \nabla f \\ f \end{pmatrix} = \mathbf{0} \; . \tag{3.8}$$

We iteratively solve (3.8) for $\mathbf{x}$ and λ [44], [75]

$$\begin{pmatrix} 2\mathbf{I} + \lambda\mathbf{H} & \nabla f \\ \nabla^{\mathrm{T}} f & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\lambda \end{pmatrix} = \begin{pmatrix} 2(\mathbf{x}_i - \mathbf{x}) - \lambda\nabla f \\ -f \end{pmatrix} , \quad \text{where} \quad \mathbf{H} = \frac{\partial}{\partial \mathbf{x}} \nabla f \; . \tag{3.9}$$

In practice, once the second term of the Lagrangian function (3.7) becomes negative (the first term is always non-negative), we have observed that it is occasionally prone to becoming more negative (i.e. diverging) in the succeeding iteration cycles of (3.9) to minimize (3.7), although the second term is commanded to be zero. Consequently, the convergence of the iteration (3.9) is considerably affected by the selection of the initial values. From this observation and with a hint at determining the first moving step size taken from the generalized Newton method described in Sect. 3.1.1, the following initial values for $\mathbf{x}$ and λ have been used:

$$\lambda_0 = \begin{cases} f/\|\nabla f\|^2\big|_{\mathbf{x}_i} & : \mathbf{x}_0 = \mathbf{x}_i \\ +1 & : \mathbf{x}_0 \neq \mathbf{x}_i \quad \text{and} \quad f(\mathbf{a}_g, \mathbf{x}_0) \geq 0 \\ -1 & : \mathbf{x}_0 \neq \mathbf{x}_i \quad \text{and} \quad f(\mathbf{a}_g, \mathbf{x}_0) < 0 \,. \end{cases} \tag{3.10}$$

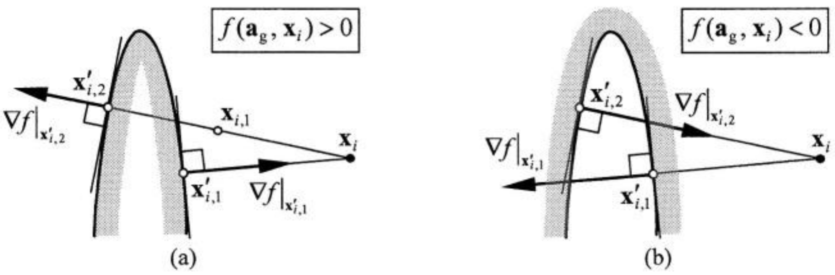
The iteration (3.9) with the initial values in (3.10) converges relatively fast. Nevertheless, if the convergence of the iteration (3.9) fails, it can be obtained using the generalized Newton method, and vice versa.

3.1.3 Verification of the Minimum Distance Point

If no closed form solution of the minimum distance point $\mathbf{x}'_i$ is available, i.e. if iteration has to be used to obtain $\mathbf{x}'_i$, it is preferable to start the inner iteration described in Sect. 3.1.1 and Sect. 3.1.2 from the initial point of $\mathbf{x}_0 = \mathbf{x}_i$ because in this way, no special initial point is prepared. As further advantage, the initial point of $\mathbf{x}_0 = \mathbf{x}_i$ almost always leads the iteration (3.6) and (3.9) to the global minimum distance point that is just the nearest point on model feature from $\mathbf{x}_i$. However, despite this advantageous choice of initial points, the global minimum distance of $\mathbf{x}'_i$ from $\mathbf{x}_i$ cannot be generally guaranteed. The conditions (3.5) and (3.8) merely constrain the connecting line of the point $\mathbf{x}$ on $f(\mathbf{a}_g, \mathbf{x}) = 0$ with the given point $\mathbf{x}_i$ to be parallel to the feature normal ∇f at $\mathbf{x}$. Thus, the minimum distance point $\mathbf{x}'_i$ found by iterations (3.6) or (3.9) is in principle only a local extreme (maximum or minimum) distance point. For example, there are a maximum of four local extreme distance points on an ellipse for a given point (Fig. 3.2). While there is no general *sufficient* condition for the global minimum distance of $\mathbf{x}'_i$ from $\mathbf{x}_i$, we check the correctness of the point $\mathbf{x}'_i$ by testing multiple independent *necessary* conditions.

With many a well-known model feature having axis or plane symmetry (e.g., ellipse, parabola, hyperbola, ellipsoid, paraboloid, etc.), the two points $\mathbf{x}'_i$ and $\mathbf{x}_i$ must lie in the same half-plane/space of the model coordinate frame xyz [7]. From this fact, it is easy to check whether $\mathbf{x}'_i$ is *not* the global minimum distance point from $\mathbf{x}_i$ by comparing the coordinate signs between $\mathbf{x}'_i$ and $\mathbf{x}_i$. If necessary, the coordinate signs of $\mathbf{x}'_i$ are selectively inverted and the iteration (3.6) or (3.9) restarts from the modified $\mathbf{x}'_i$.

The correctness of point $\mathbf{x}'_i$ can also be checked by examining the direction of the feature normal ∇f at $\mathbf{x}'_i$. Once the iteration (3.6) or (3.9) has been successfully terminated, the direction of the feature normal is verified by testing another minimum distance condition (Fig. 3.4) below:



$f(\mathbf{a}_g, \mathbf{x}_i) > 0$
$\mathbf{x}'_{i,2}$
$\nabla f|_{\mathbf{x}'_{i,2}}$
$\mathbf{x}_{i,1}$
$\mathbf{x}_i$
$\mathbf{x}'_{i,1}$
$\nabla f|_{\mathbf{x}'_{i,1}}$
(a)
$f(\mathbf{a}_g, \mathbf{x}_i) < 0$
$\mathbf{x}'_{i,2}$
$\nabla f|_{\mathbf{x}'_{i,2}}$
$\mathbf{x}_i$
$\nabla f|_{\mathbf{x}'_{i,1}}$
$\mathbf{x}'_{i,1}$
(b)

Fig. 3.4. Verification of the minimum distance point. The point $\mathbf{x}'_{i,1}$ satisfies (3.11), while $\mathbf{x}'_{i,2}$ not: (a) $f(\mathbf{a}_g,\mathbf{x}_i) > 0$; (b) $f(\mathbf{a}_g,\mathbf{x}_i) < 0$

$$\left.\frac{\nabla f}{\|\nabla f\|}\right|_{\mathbf{x}'_i} \cdot \frac{\mathbf{x}_i - \mathbf{x}'_i}{\|\mathbf{x}_i - \mathbf{x}'_i\|} - \mathrm{sign}(f(\mathbf{a}_g,\mathbf{x}_i)) = 0 \ . \tag{3.11}$$

Rarely, if (3.11) could not yet be satisfied with a value (probably ±2) other than zero, then the connecting line of the two points $\mathbf{x}'_i$ and $\mathbf{x}_i$ pierces the model feature an odd number of times (i.e., at least, $\mathbf{x}'_i$ is not the global minimum distance point). In this case, the iteration (3.6) or (3.9) is repeated with an adjusted starting point such as $\mathbf{x}_{i,1}$ on the connecting line of the two points $\mathbf{x}'_i$ and $\mathbf{x}_i$ in Fig. 3.4a.

3.1.4 Acceleration of Finding the Minimum Distance Point

The use of the initial point of $\mathbf{x}_0 = \mathbf{x}_i$ for the iterative search for the minimum distance point $\mathbf{x}'_i$ is preferable and very convenient but it demands a relatively high computing cost. Fortunately, the parameter update $\Delta\mathbf{a}$ is generally moderate in the second half-phase of the outer iteration (after the first 5–10 outer iteration cycles with most fitting tasks) and the minimum distance point $\mathbf{X}'_i$ in XYZ frame changes its position incrementally between two consecutive outer iteration cycles. Thus, dramatic savings can be made in the computing cost associated with the inner iteration and, consequently, in the overall computing cost of the ODF, if the current minimum distance point $\mathbf{X}'_i$ is used as a reasonable initial point for the inner iteration inside the next outer iteration cycle

$$\mathbf{x}_{0,\mathbf{a}+\alpha\Delta\mathbf{a}} = \mathbf{R}_{\mathbf{a}+\alpha\Delta\mathbf{a}}(\mathbf{X}'_{i,\mathbf{a}} - \mathbf{X}_{o,\mathbf{a}+\alpha\Delta\mathbf{a}}) \ . \tag{3.12}$$

However, a pitfall of using (3.12) must be mentioned. If the current minimum distance point $\mathbf{X}'_i$ once becomes a local (not global) minimum distance point after an update of the parameters **a,** it can be inherited into the next outer iteration as long as (3.12) is used (Fig. 3.5). In order to interrupt this unintended development, the inner iteration is started periodically (e.g. with every 5–10th outer iteration cycle) from the given point $\mathbf{X}_i$ during the second half-phase of the outer iteration

$$\mathbf{x}_{0,\mathbf{a}+\alpha\Delta\mathbf{a}} = \mathbf{R}_{\mathbf{a}+\alpha\Delta\mathbf{a}}(\mathbf{X}_i - \mathbf{X}_{o,\mathbf{a}+\alpha\Delta\mathbf{a}}) \ . \tag{3.13}$$

The strategy for an efficient and inexpensive way of finding the minimum distance point $\mathbf{x}'_i$ is summarized below:

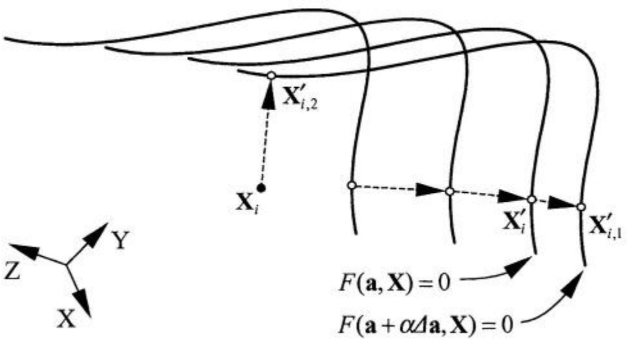

$X'_{i,2}$
X_i
X'_i
$X'_{i,1}$
Y
Z
X
$F(\mathbf{a},\mathbf{X})=0$
$F(\mathbf{a}+\alpha\Delta\mathbf{a},\mathbf{X})=0$

Fig. 3.5. Minimum distance points $\mathbf{X}'_{i,1}$ and $\mathbf{X}'_{i,2}$ on $F(\mathbf{a}+\alpha\Delta\mathbf{a},\mathbf{X})=0$ (updated implicit feature) from the given point $\mathbf{X}_i$. Iterative search for the minimum distance point may converge to $\mathbf{X}'_{i,1}$, if the iterations (3.6) or (3.9) started from the current minimum distance point $\mathbf{X}'_i$ on $F(\mathbf{a},\mathbf{X}) = 0$ (see (3.12)), while starting from the given point $\mathbf{X}_i$ converges to $\mathbf{X}'_{i,2}$ (see (3.13)), being $\|\mathbf{X}_i-\mathbf{X}'_{i,1}\| > \|\mathbf{X}_i-\mathbf{X}'_{i,2}\|$

- The closed form solution of $\mathbf{x}'_i$ is used if available
- The inner iteration is started from $\mathbf{x}_0$ of (3.13) in the first half-phase of the outer iteration, during which the parameter update $\Delta\mathbf{a}$ is generally wild
- In the second half-phase of the outer iteration, the inner iteration is started from $\mathbf{x}_0$ of (3.12)
- Periodically, in the second half-phase of the outer iteration, the inner iteration is started from $\mathbf{x}_0$ of (3.13).

3.2 Orthogonal Distance Fitting

With the algorithms described in the previous sections, the minimum distance points $\{\mathbf{X}'_i\}_{i=1}^m$ on the current model feature from each given point $\{\mathbf{X}_i\}_{i=1}^m$ have been found (inner iteration) and are now available for parameter updating (outer iteration) in the nested iteration scheme (3.1). The inner iteration and parameter updating are to be alternately repeated until appropriate break conditions for the outer iteration are satisfied. In this section, the implementation of two parameter updating algorithms, the distance-based algorithm (2.28) and the coordinate-based algorithm (2.32), is described in detail for general implicit features. With the implementation, the model parameters $\mathbf{a}$ are grouped in terms of form $\mathbf{a}_g$, position $\mathbf{a}_p$, and rotation parameters $\mathbf{a}_r$.

3.2.1 Distance-Based Algorithm

In order to complete the linear equation system (2.28), the Jacobian matrices $\{\mathbf{J}_{d_i,\mathbf{a}}\}_{i=1}^m$ of each distance d_i between the given point $\mathbf{X}_i$ and the *minimum distance point* $\mathbf{X}'_i$ have to be derived.

$$\begin{aligned} d_i &= \|\mathbf{X}_i - \mathbf{X}'_i\| \\ &= \sqrt{(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}(\mathbf{X}_i - \mathbf{X}'_i)}\,. \end{aligned} \tag{3.14}$$

From $\mathbf{X} = \mathbf{R}^{-1}\mathbf{x} + \mathbf{X}_o$ (1.3) and (3.14), the following is obtained

$$\begin{aligned} \mathbf{J}_{d_i,\mathbf{a}} &= \frac{\partial d_i}{\partial \mathbf{a}} \\ &= -\frac{(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}}{\|\mathbf{X}_i - \mathbf{X}'_i\|}\left.\frac{\partial \mathbf{X}}{\partial \mathbf{a}}\right|_{\mathbf{X}=\mathbf{X}'_i} \\ &= -\frac{(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}}{\|\mathbf{X}_i - \mathbf{X}'_i\|}\left.\left(\mathbf{R}^{\mathrm{T}}\frac{\partial \mathbf{x}}{\partial \mathbf{a}} + \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}}[\mathbf{x}] + \frac{\partial \mathbf{X}_o}{\partial \mathbf{a}}\right)\right|_{\mathbf{x}=\mathbf{x}'_i} \\ &= -\frac{(\mathbf{x}_i - \mathbf{x}'_i)^{\mathrm{T}}}{\|\mathbf{x}_i - \mathbf{x}'_i\|}\left.\frac{\partial \mathbf{x}}{\partial \mathbf{a}}\right|_{\mathbf{x}=\mathbf{x}'_i} - \frac{(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}}{\|\mathbf{X}_i - \mathbf{X}'_i\|}\left(\begin{array}{c|c|c} \mathbf{0} & \mathbf{I} & \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}_{\mathrm{r}}}[\mathbf{x}'_i] \end{array}\right), \end{aligned} \tag{3.15}$$

where

$$\mathbf{a}^{\mathrm{T}} \triangleq (\mathbf{a}_{\mathrm{g}}^{\mathrm{T}}, \mathbf{a}_{\mathrm{p}}^{\mathrm{T}}, \mathbf{a}_{\mathrm{r}}^{\mathrm{T}})\,, \quad \frac{\partial \mathbf{R}}{\partial \mathbf{a}_{\mathrm{r}}} \triangleq \left(\frac{\partial \mathbf{R}}{\partial \omega} \quad \frac{\partial \mathbf{R}}{\partial \varphi} \quad \frac{\partial \mathbf{R}}{\partial \kappa}\right), \quad [\mathbf{x}] \triangleq \begin{pmatrix} \mathbf{x} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{x} \end{pmatrix}.$$

Also, from the derivatives of implicit equation $f(\mathbf{a}_{\mathrm{g}}, \mathbf{x}) = 0$ (1.1)

$$\frac{\partial f}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{a}} + \frac{\partial f}{\partial \mathbf{a}} = 0\,,$$

the following is obtained

$$\begin{aligned} \nabla^{\mathrm{T}} f\frac{\partial \mathbf{x}}{\partial \mathbf{a}} &= \frac{\partial f}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{a}} \\ &= -\frac{\partial f}{\partial \mathbf{a}} \\ &= \left(\begin{array}{c|c|c} -\frac{\partial f}{\partial \mathbf{a}_{\mathrm{g}}} & \mathbf{0}^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} \end{array}\right). \end{aligned} \tag{3.16}$$

Then, with (3.16) and from the fact $(\mathbf{x}_i - \mathbf{x}'_i) /\!/ \nabla f|_{\mathbf{x}=\mathbf{x}'_i}$, the Jacobian matrix $\mathbf{J}_{d_i,\mathbf{a}}$ (3.15) eventually becomes

$$\begin{aligned} \mathbf{J}_{d_i,\mathbf{a}} = &\left(\begin{array}{c|c|c} \underbrace{\frac{\operatorname{sign}\left((\mathbf{x}_i - \mathbf{x}'_i)^{\mathrm{T}}\nabla f\right)}{\|\nabla f\|}\left.\frac{\partial f}{\partial \mathbf{a}_{\mathrm{g}}}\right|_{\mathbf{x}=\mathbf{x}'_i}}_{1\times l} & \underbrace{\mathbf{0}^{\mathrm{T}}}_{1\times n} & \underbrace{\mathbf{0}^{\mathrm{T}}}_{1\times s} \end{array}\right) \\ &- \underbrace{\frac{(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}}{\|\mathbf{X}_i - \mathbf{X}'_i\|}}_{1\times n}\left(\begin{array}{c|c|c} \underbrace{\mathbf{0}}_{n\times l} & \underbrace{\mathbf{I}}_{n\times n} & \underbrace{\frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}_{\mathrm{r}}}[\mathbf{x}'_i]}_{n\times s} \end{array}\right). \end{aligned} \tag{3.17}$$

In comparison with the Sullivan et al. algorithm [75], this new algorithm consisting of (2.28) and (3.17) simultaneously estimates the parameters **a** in terms of form, position, and rotation parameters. Also, with using the Sullivan et al. algorithm, unnecessary costs are required to obtain the Jacobian matrix from the minimum distance condition defined in the machine coordinate frame XYZ

$$\begin{pmatrix} F \\ \nabla F \times (\mathbf{X}_i - \mathbf{X}) \end{pmatrix} = \mathbf{0} \quad \text{with} \quad \nabla \triangleq (\partial/\partial X,\ \partial/\partial Y,\ \partial/\partial Z)^{\mathrm{T}} \ . \tag{3.18}$$

If a similar procedure of (3.15)–(3.17) is applied to $F(\mathbf{b}, \mathbf{X}) = 0$ with algebraic parameters **b** defined in the machine coordinate frame XYZ, the Jacobian matrix for the Sullivan et al. algorithm can be obtained as below at a lower implementation cost without using (3.18):

$$\mathbf{J}_{d_i,\mathbf{b}} = \left. \frac{\operatorname{sign}\left((\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}\nabla F\right)}{\|\nabla F\|} \frac{\partial F}{\partial \mathbf{b}} \right|_{\mathbf{X}=\mathbf{X}'_i} .$$

3.2.2 Coordinate-Based Algorithm

The Jacobian matrices $\{\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}}\}_{i=1}^{m}$ (necessary for completing the linear equation system (2.32)) of each *minimum distance point* $\mathbf{X}'_i$ on the current model feature can be directly derived from $\mathbf{X} = \mathbf{R}^{-1}\mathbf{x} + \mathbf{X}_o$ (1.3)

$$\begin{aligned} \mathbf{J}_{\mathbf{X}'_i,\mathbf{a}} &= \left. \frac{\partial \mathbf{X}}{\partial \mathbf{a}} \right|_{\mathbf{X}=\mathbf{X}'_i} \\ &= \left. \left(\mathbf{R}^{\mathrm{T}} \frac{\partial \mathbf{x}}{\partial \mathbf{a}} + \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}} [\mathbf{x}] + \frac{\partial \mathbf{X}_o}{\partial \mathbf{a}} \right) \right|_{\mathbf{x}=\mathbf{x}'_i} \\ &= \underbrace{\mathbf{R}^{\mathrm{T}} \left. \frac{\partial \mathbf{x}}{\partial \mathbf{a}} \right|_{\mathbf{x}=\mathbf{x}'_i}}_{n\times p} + \left(\underbrace{\mathbf{0}}_{n\times l} \;\middle|\; \underbrace{\mathbf{I}}_{n\times n} \;\middle|\; \underbrace{\frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}_{\mathrm{r}}} [\mathbf{x}'_i]}_{n\times s} \right) . \end{aligned} \tag{3.19}$$

The derivative matrix $\partial \mathbf{x}/\partial \mathbf{a}$ at $\mathbf{x} = \mathbf{x}'_i$ in (3.19) describes the variational behavior of the minimum distance point $\mathbf{x}'_i$ in xyz frame relative to the differential changes of the parameters vector **a**. Purposefully, $\partial \mathbf{x}/\partial \mathbf{a}$ is obtained from the minimum distance condition (3.5). Because (3.5) has an implicit form, its derivatives lead to

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{a}} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}_i} \frac{\partial \mathbf{x}_i}{\partial \mathbf{a}} + \frac{\partial \mathbf{f}}{\partial \mathbf{a}} = \mathbf{0} \quad \text{or} \quad \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{a}} = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}_i} \frac{\partial \mathbf{x}_i}{\partial \mathbf{a}} + \frac{\partial \mathbf{f}}{\partial \mathbf{a}} \right) , \tag{3.20}$$

where $\partial \mathbf{x}_i/\partial \mathbf{a}$ is, from $\mathbf{x}_i = \mathbf{R}(\mathbf{X}_i - \mathbf{X}_o)$ (3.2),

$$\begin{aligned} \frac{\partial \mathbf{x}_i}{\partial \mathbf{a}} &= \frac{\partial \mathbf{R}}{\partial \mathbf{a}} [\mathbf{X}_i - \mathbf{X}_o] - \mathbf{R} \frac{\partial \mathbf{X}_o}{\partial \mathbf{a}} \\ &= \left(\mathbf{0} \;\middle|\; -\mathbf{R} \;\middle|\; \frac{\partial \mathbf{R}}{\partial \mathbf{a}_{\mathrm{r}}} [\mathbf{X}_i - \mathbf{X}_o] \right) . \end{aligned}$$

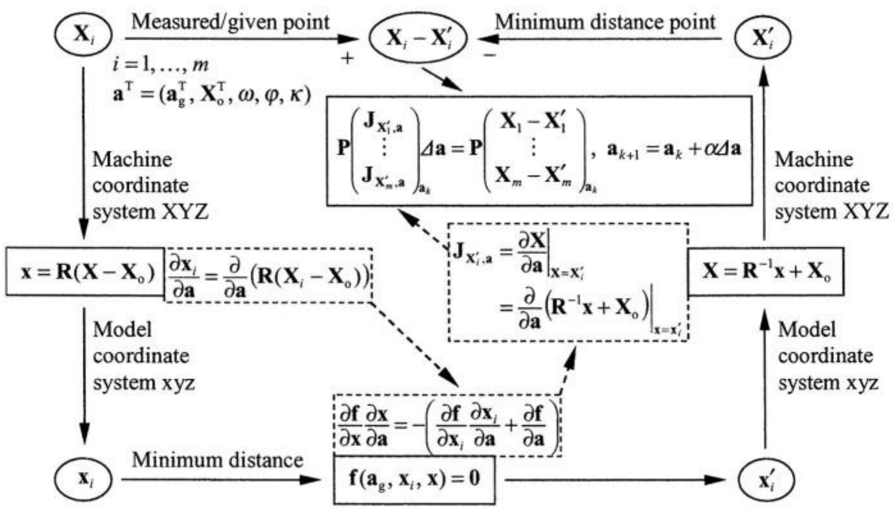

$\mathbf{X}_i$
Measured/given point
$i=1,\dots,m$
$\mathbf{a}^T=(\mathbf{a}_g^T, \mathbf{X}_o^T, \omega, \varphi, \kappa)$
$\mathbf{X}_i - \mathbf{X}'_i$
Minimum distance point
$\mathbf{X}'_i$
+
−
$\mathbf{P}\begin{pmatrix}\mathbf{J}_{\mathbf{X}'_1,\mathbf{a}}\\ \vdots \\ \mathbf{J}_{\mathbf{X}'_m,\mathbf{a}}\end{pmatrix}_{\mathbf{a}_k}\Delta\mathbf{a}=\mathbf{P}\begin{pmatrix}\mathbf{X}_1-\mathbf{X}'_1\\ \vdots \\ \mathbf{X}_m-\mathbf{X}'_m\end{pmatrix}_{\mathbf{a}_k}, \quad \mathbf{a}_{k+1}=\mathbf{a}_k+\alpha\Delta\mathbf{a}$
Machine coordinate system XYZ
Machine coordinate system XYZ
$\mathbf{x}=\mathbf{R}(\mathbf{X}-\mathbf{X}_o)$
$\frac{\partial \mathbf{x}_i}{\partial \mathbf{a}}=\frac{\partial}{\partial \mathbf{a}}(\mathbf{R}(\mathbf{X}_i-\mathbf{X}_o))$
$\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}}=\left.\frac{\partial \mathbf{X}}{\partial \mathbf{a}}\right|_{\mathbf{x}=\mathbf{x}'_i}=\left.\frac{\partial}{\partial \mathbf{a}}(\mathbf{R}^{-1}\mathbf{x}+\mathbf{X}_o)\right|_{\mathbf{x}=\mathbf{x}'_i}$
$\mathbf{X}=\mathbf{R}^{-1}\mathbf{x}+\mathbf{X}_o$
Model coordinate system xyz
Model coordinate system xyz
$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{a}}=-\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}_i}\frac{\partial \mathbf{x}_i}{\partial \mathbf{a}}+\frac{\partial \mathbf{f}}{\partial \mathbf{a}}\right)$
$\mathbf{x}_i$
Minimum distance
$\mathbf{f}(\mathbf{a}_g, \mathbf{x}_i, \mathbf{x})=\mathbf{0}$
$\mathbf{x}'_i$

Fig. 3.6. Information flow for the orthogonal distance fitting of implicit features (coordinate-based algorithm)

The other three matrices $\partial\mathbf{f}/\partial\mathbf{x}$, $\partial\mathbf{f}/\partial\mathbf{x}_i$, and $\partial\mathbf{f}/\partial\mathbf{a}$ in (3.6) and (3.20) are directly derived from (3.5). The elements of these three matrices are composed of simple linear combinations of components of the error vector $(\mathbf{x}_i - \mathbf{x})$ with elements of the following three vector/matrices ∇f, $\mathbf{H}$, and $\mathbf{G}$ (FHG matrix):

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)^{\mathrm{T}}, \quad \mathbf{H} = \frac{\partial}{\partial \mathbf{x}}\nabla f, \quad \mathbf{G} \triangleq \frac{\partial}{\partial \mathbf{a}_g}\begin{pmatrix} f \\ \nabla f\end{pmatrix}, \tag{3.21}$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} 0 & 0 & 0 \\ y_i - y & -(x_i - x) & 0 \\ -(z_i - z) & 0 & x_i - x \\ 0 & z_i - z & -(y_i - y) \end{pmatrix}\mathbf{H} + \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \\ \frac{\partial f}{\partial y} & -\frac{\partial f}{\partial x} & 0 \\ -\frac{\partial f}{\partial z} & 0 & \frac{\partial f}{\partial x} \\ 0 & \frac{\partial f}{\partial z} & -\frac{\partial f}{\partial y} \end{pmatrix},$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}_i} = \begin{pmatrix} 0 & 0 & 0 \\ -\frac{\partial f}{\partial y} & \frac{\partial f}{\partial x} & 0 \\ \frac{\partial f}{\partial z} & 0 & -\frac{\partial f}{\partial x} \\ 0 & -\frac{\partial f}{\partial z} & \frac{\partial f}{\partial y} \end{pmatrix},$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{a}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & y_i - y & -(x_i - x) & 0 \\ 0 & -(z_i - z) & 0 & x_i - x \\ 0 & 0 & z_i - z & -(y_i - y) \end{pmatrix}\left(\ \mathbf{G} \mid \mathbf{0} \mid \mathbf{0}\ \right).$$

Then (3.20) can be solved for $\partial\mathbf{x}/\partial\mathbf{a}$ at $\mathbf{x} = \mathbf{x}'_i$ and, consequently, the Jacobian matrix $\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}}$ (3.19) and the linear equation system (2.32) can be completed and solved for the parameter update $\Delta\mathbf{a}$.

For the sake of a common program module for curves and surfaces, without loss of generality, the second and the fourth row of (3.5) have been interchanged. Therefore, for plane curve fitting, only the first two rows of the modified equation (3.5) are considered, in other words only the upper-left block of the FHG matrix in (3.21) is taken into account (see Sect. A.1 for the case of a 2-D ellipse).

3.2.3 Comparison of the Two Algorithms

Both the distance-based algorithm (2.28) and the coordinate-based algorithms (2.32) for the ODF of implicit features are versatile and highly efficient from the viewpoint of their application to a new model feature. We would like to stress that only the standard feature equation $f(\mathbf{a}_g, \mathbf{x}) = 0$ (1.1), without involvement of the position/rotation parameters, is required by (3.21) for implementation of either of the two algorithms for a new implicit feature (the second derivatives $\partial\nabla f/\partial\mathbf{a}_g$ are required only by the coordinate-based algorithm). The overall structure of our algorithms remains unchanged for all model fitting problems of implicit features. All that is necessary for the ODF of a new implicit feature is the derivation of the FHG matrix of (3.21) from (1.1) of the new model feature and the provision of a set of initial parameter values $\mathbf{a}_0$ for iterations (2.28) or (2.32). This fact makes possible the realization of the versatile and very efficient ODF algorithms for implicit surfaces and plane curves. An overall schematic information flow is shown in Fig. 3.6 (coordinate-based algorithm). The two algorithms are compared below from the viewpoint of implementation and computing cost:

- Common features
  - For implementation to a new model feature, only the standard feature equation (1.1) is required eventually
  - The computing cost and the memory space usage are proportional to the number of data points
  - The two algorithms demand approximately the same computing cost because the time-consuming part of the overall procedure is to find the minimum distance points.
- Different features
  - The distance-based algorithm demands lower memory space usages (with plane curve fitting one half and with surface fitting one third of the requirements of the coordinate-based algorithm)
  - The implementation of the distance-based algorithm for a new model feature is relatively easy because it does not require the second derivatives $\partial\nabla f/\partial\mathbf{a}_g$ of the feature equation (1.1)
  - With the coordinate-based algorithm, each coordinate X_i, Y_i, Z_i of a measurement point can be individually weighted, while only point-wise weighting is possible with the distance-based algorithm. This algorithmic feature of the coordinate-based algorithm is practical for measuring devices where measuring accuracy is not identical between measuring axes. From this fact, it can be said that the coordinate-based algorithm is a more general one than the distance-based algorithm

– A comparison between (3.15) and (3.19) reveals

$$d_i = \mathbf{n}_i^{\mathrm{T}}(\mathbf{X}_i - \mathbf{X}'_i) \; , \quad \mathbf{J}_{d_i,\mathbf{a}} = -\mathbf{n}_i^{\mathrm{T}}\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}} \; , \quad \text{with } \mathbf{n}_i = \frac{\mathbf{X}_i - \mathbf{X}'_i}{\|\mathbf{X}_i - \mathbf{X}'_i\|} \; .$$

3.3 Fitting Examples

3.3.1 Superellipse Fitting

A superellipse [35] is the generalization of an ellipse and is described by

$$f(a, b, \varepsilon, \mathbf{x}) \triangleq (|x|/a)^{2/\varepsilon} + (|y|/b)^{2/\varepsilon} - 1 = 0 \; ,$$

where a, b are the axis-lengths and exponent ε is the shape coefficient. The use of superellipses in applications of image processing and pattern recognition is highly desirable because a superellipse can represent a variety of 2-D features such as a rectangle ($\varepsilon \ll 1$), ellipse ($\varepsilon = 1$), and star ($\varepsilon > 2$). Many researchers have attempted to fit superellipses using the moment method [18], [79] or the LSM with algebraic distances [68].

In order to demonstrate the outstanding performance of our ODF algorithms, an extreme shape of superellipse, a rectangle, is fitted. The initial parameter values are obtained from a circle fitting and an ellipse fitting, successively. In detail, the mass center and rms central distance of the set of given points are used as the initial parameter values for circle fitting, circle parameters for ellipse fitting [6], [7] and finally, ellipse parameters for superellipse fitting (Fig. 3.7, Table 3.2). The initial value of shape coefficient is usually set to $\varepsilon = 1$. Superellipse fitting to the eight points in Table 3.1 representing a rectangle terminated after seven iteration cycles for $\|\Delta\mathbf{a}\| = 7.8 \cdot 10^{-9}$ (Fig. 3.7d, Table 3.2).

Table 3.1. Eight coordinate pairs ($n = 2$) representing a rectangle

X	1	10	30	50	60	72	79	8
Y	0	5	15	25	30	15	0	−15

Table 3.2. Results of orthogonal distance fitting to the points in Table 3.1

Parameter $\hat{\mathbf{a}}$	σ_0	a	b	ε	X_o	Y_o	κ
Circle	12.3547	39.0709	—	—	44.5610	−12.9728	—
$\sigma(\hat{\mathbf{a}})$ II	—	4.3164	—	—	3.0228	6.3312	—
III	—	4.2428	—	—	2.9945	6.1921	—
Ellipse	9.6776	39.1512	22.1785	—	41.3761	0.4474	0.2837
$\sigma(\hat{\mathbf{a}})$ II	—	4.0283	6.5574	—	3.2054	4.7416	0.1578
III	—	3.8490	6.2789	—	3.1623	4.6243	0.1541
Superellipse	0.0031	35.3298	17.8892	0.0527	40.0000	0.0001	0.4637
$\sigma(\hat{\mathbf{a}})$ II	—	0.0015	0.0026	0.0002	0.0012	0.0025	0.0001
III	—	0.0015	0.0026	0.0002	0.0012	0.0025	0.0001

II: distance-based algorithm, III: coordinate-based algorithm.

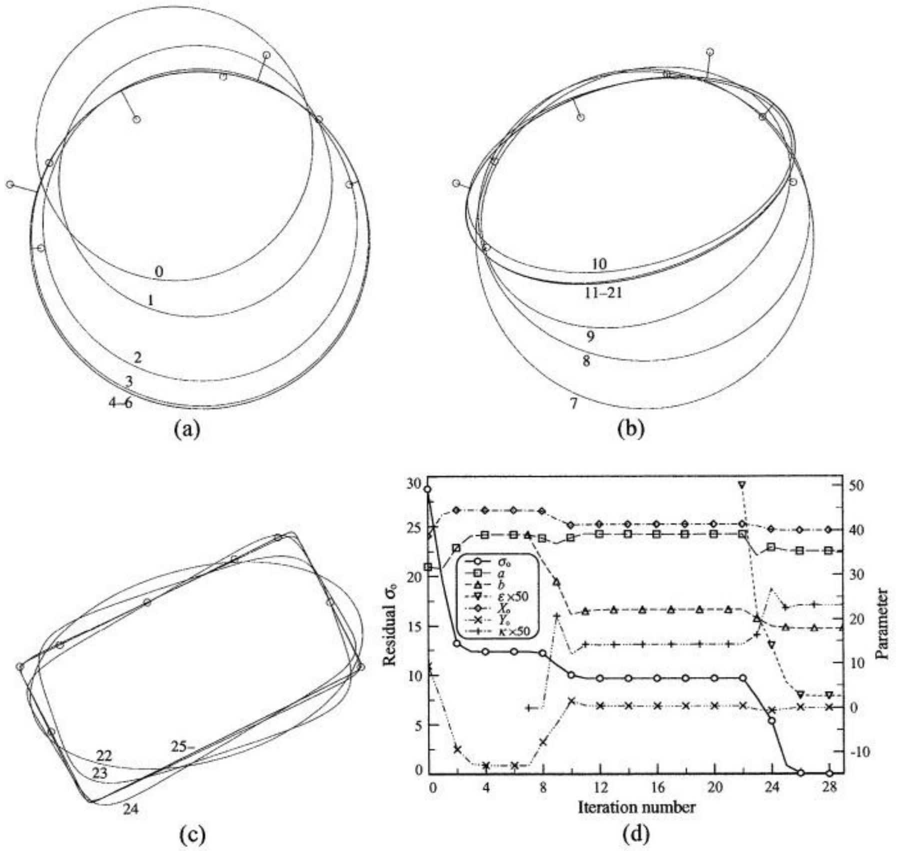


Fig. 3.7. Orthogonal distance fitting to the set of points in Table 3.1: (a) Circle fit; (b) Ellipse fit; (c) Superellipse fit; (d) Convergence of the fit. Iteration number 0–6: circle, 7–21: ellipse, and 22–: superellipse fit

3.3.2 Cone Fitting

The standard model feature of a cone in xyz frame can be described as below:

$$f(\psi, \mathbf{x}) \triangleq x^2 + y^2 - (z \tan(\psi/2))^2 = 0 ,$$

where ψ, the only form parameter, is the vertex angle of a cone. The position $\mathbf{X}_o$, the origin of xyz frame, is defined at the vertex. The orientation of a cone is represented by the direction cosine vector of the z-axis, $\mathbf{r}_z$ in (1.3). However, from the viewpoint of coordinate metrology (Appendix B and [60]), a better parameterization of a cone is

$$f(\psi, r, \mathbf{x}) \triangleq x^2 + y^2 - (r - z \tan(\psi/2))^2 = 0 ,$$

where the second form parameter r is the sectional radius of the cone cut by the xy-plane ($z = 0$). An additional constraint forces the position $\mathbf{X}_o$ of the cone to

Table 3.3. Ten coordinate triples ($n = 3$) representing a quarter of a cone slice

X [mm]	734.8905	739.8980	736.4229	850.6449	850.6271
Y [mm]	−720.8340	−736.6202	−750.8837	−699.2051	−718.8401
Z [mm]	−735.4193	−731.4877	−731.9028	−645.0159	−645.7938
X [mm]	919.1539	921.6025	935.7441	931.8560	927.4975
Y [mm]	−711.7191	−727.6805	−766.1543	−781.5263	−809.7823
Z [mm]	−527.1499	−519.4751	−391.4352	−372.0004	−388.0452

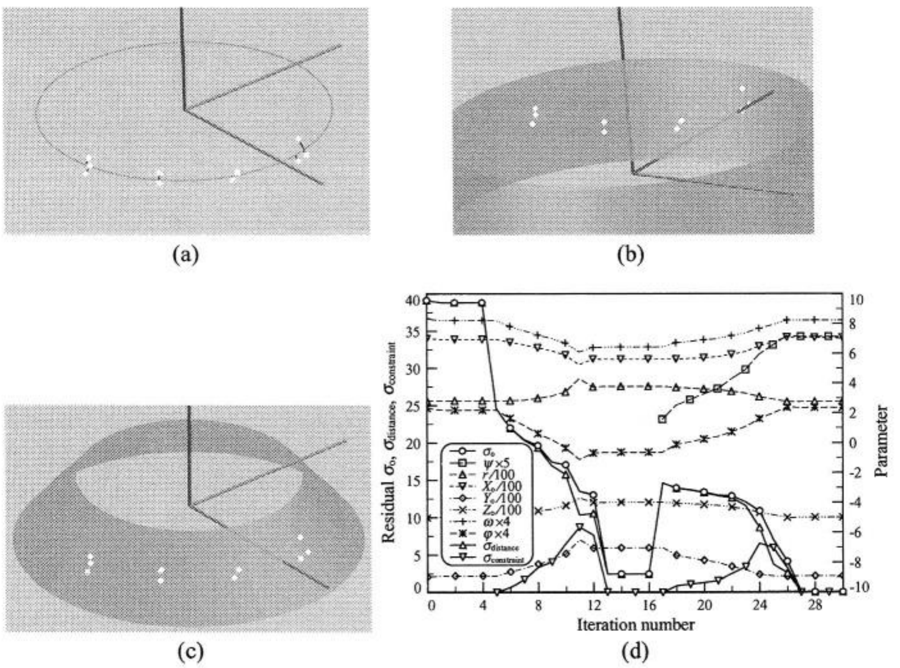


Fig. 3.8. Orthogonal distance fitting to the set of points in Table 3.3: (a) 3-D circle fit; (b) Cylinder fit; (c) Cone fit; (d) Convergence of the fit. Iteration number 0–4: 3-D circle, 5–16: cylinder, and 17–: cone fit with the initial value of $\psi = \pi/10$

be the nearest point on the cone axis $\mathbf{r}_z$ from the mass center $\bar{\mathbf{X}}$ of the set of given points (see (2.33), Sect. B.1.2 and Sect. B.2.8)

$$f_c(\mathbf{a}_p, \mathbf{a}_r) \triangleq (\mathbf{X}_o - \bar{\mathbf{X}})^T \mathbf{r}_z(\omega, \varphi) = 0 \,.$$

As one of the test data sets used in an authorized certification process [29], [60] of our algorithms, the ten points in Table 3.3 representing a quarter of a cone slice were prepared by the German federal authority PTB. The initial parameter values were obtained from a 3-D circle fitting (see Chap. 4) and a cylinder fitting, successively (Fig. 1.3, Fig. 3.8, and Table 3.4). The initial value of vertex angle is usually set to $\psi = 0$. The cone fitting to the set of points in Table 3.3 with the constraint weighting value of $w_c = 1.0$ terminated after 60 iteration cycles

Table 3.4. Results of orthogonal distance fitting to the set of points in Table 3.3

Parameter $\hat{\mathbf{a}}$	σ_0	ψ [rad]	r [mm]	X_o [mm]
3-D Circle	38.8480	—	283.0367	694.5271
$\sigma(\hat{\mathbf{a}})$	—	—	38.0351	38.1879
Cylinder	2.4655	—	379.0909	561.5321
$\sigma(\hat{\mathbf{a}})$	—	—	10.4544	8.6758
Cone	0.0357	1.4262	276.4373	706.7202
$\sigma(\hat{\mathbf{a}})$	—	0.0065	0.8173	1.2504

Parameter $\hat{\mathbf{a}}$	Y_o [mm]	Z_o [mm]	ω [rad]	φ [rad]
3-D Circle	−889.7335	−498.1031	2.0534	0.5477
$\sigma(\hat{\mathbf{a}})$	41.8194	22.2402	0.1137	0.1365
Cylinder	−702.1460	−398.2213	1.6090	−0.1578
$\sigma(\hat{\mathbf{a}})$	11.2828	6.1403	0.0432	0.0282
Cone	−890.5186	−499.1046	2.0554	0.5877
$\sigma(\hat{\mathbf{a}})$	0.0456	0.6477	0.0029	0.0037

for $\|\Delta\mathbf{a}\| = 2.3 \cdot 10^{-7}$ (coordinate-based algorithm). The convergence is somewhat slow because the initial cone parameters (i.e., the cylinder parameters) seem to provide a local minimum (stationary point). It is especially slow if the constraint weighting value w_c is large and therefore hinders the parameters from changing. If this quasi-equilibrium state is perturbed, similarly to the random walking technique, by adding a small artifact error to the initial parameter values, then a faster convergence can be expected. With a non-zero initial vertex angle value of $\psi = \pi/10$, the iteration terminates after only 13 cycles for $\|\Delta\mathbf{a}\| = 2.9 \cdot 10^{-6}$ (Fig. 3.8d).

3.3.3 Torus Fitting

A torus is a quartic surface described by

$$\begin{aligned} f(r_1, r_2, \mathbf{x}) \triangleq\; & x^4 + y^4 + z^4 + 2(x^2y^2 + y^2z^2 + z^2x^2) \\ & - 2(r_2^2 + r_1^2)(x^2 + y^2) + 2(r_2^2 - r_1^2)z^2 + (r_2^2 - r_1^2)^2 = 0 , \end{aligned}$$

where r_1 is the radius of the circular section of the tube and r_2 is the mean radius of the ring. The initial parameter values are obtained from a 3-D circle fitting (Chap. 4) (Fig. 1.3, Fig. 3.9) with

$$r_1 = \sqrt{\sigma^2_{0,\text{circle}}/m} , \qquad r_2 = r_{\text{circle}} .$$

Table 3.5. Ten coordinate triples ($n = 3$) representing a half torus

X	10	2	0	6	5	7	0	9	10	5
Y	7	11	−7	2	7	1	−6	1	−1	−5
Z	4	7	−1	2	8	−2	2	3	−1	1

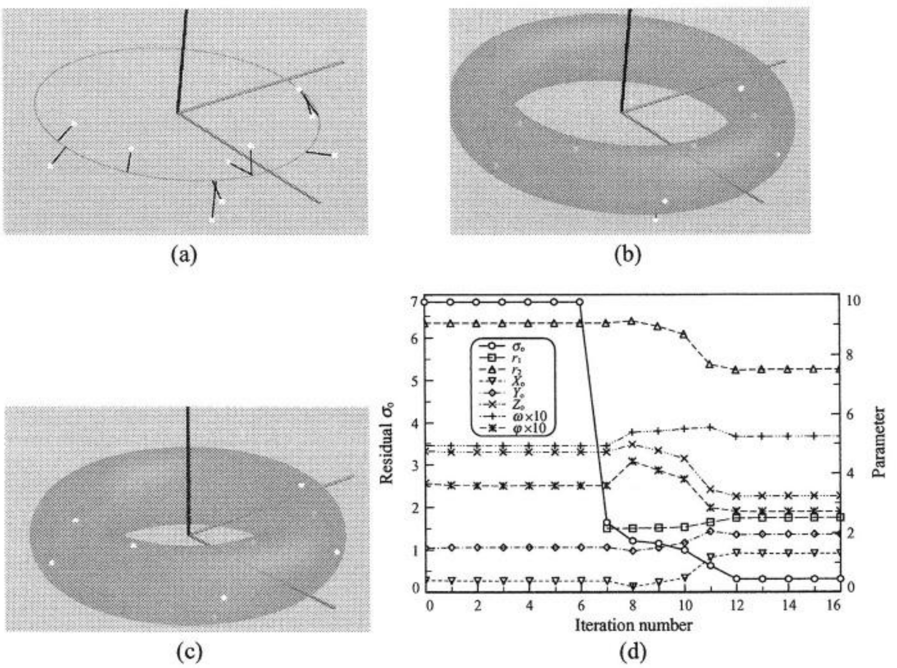


Fig. 3.9. Orthogonal distance fitting to the set of points in Table 3.5: (a) 3-D circle fit; (b) Initial torus; (c) Torus fit; (d) Convergence of the fit. Iteration number 0–6: 3-D circle, and 7–: torus fit

Table 3.6. Results of orthogonal distance fitting to the set of points in Table 3.5

Parameter $\hat{\mathbf{a}}$	σ_0	r_1	r_2	X_o	Y_o	Z_o	ω	φ
3-D Circle	6.8370	—	9.0588	0.3831	1.5271	4.7164	0.4932	0.3584
$\sigma(\hat{\mathbf{a}})$	—	—	2.0388	2.6476	1.6286	2.1410	0.1911	0.2992
Torus	0.3104	2.5103	7.5121	1.3159	1.9548	3.2324	0.5265	0.2720
$\sigma(\hat{\mathbf{a}})$	—	0.0844	0.1792	0.1908	0.1099	0.2056	0.0175	0.0268

Torus fitting to the ten points in Table 3.5 representing a half torus terminated after nine iteration cycles for $\|\Delta\mathbf{a}\| = 1.4 \cdot 10^{-7}$ (coordinate-based algorithm, Fig. 3.9d and Table 3.6).

3.3.4 Superellipsoid Fitting

A Superellipsoid (superquadric) [16] is the generalization of an ellipsoid and is described by

$$f(a,b,c,\varepsilon_1,\varepsilon_2,\mathbf{x}) \triangleq \left((|x|/a)^{2/\varepsilon_1} + (|y|/b)^{2/\varepsilon_1} \right)^{\varepsilon_1/\varepsilon_2} + (|z|/c)^{2/\varepsilon_2} - 1 = 0 \,,$$

where a, b, c are the axis-lengths and exponents ε_1, ε_2 are the shape coefficients. In comparison with the algebraic fitting algorithm [73], our algorithms can fit also

Table 3.7. 30 coordinate triples ($n = 3$) representing a box

X	−4	1	4	20	−11	−26	−3	−7	6	11
Y	3	16	−11	−17	1	7	−13	−26	19	24
Z	13	29	10	22	4	−8	1	−15	9	19
X	3	15	18	−21	−4	−2	−14	20	4	6
Y	−18	9	−3	3	−14	19	14	−17	−20	20
Z	−25	21	22	−13	−22	11	1	15	−18	24
X	22	30	−8	−16	8	26	−22	−2	−3	7
Y	−8	−9	15	15	−13	−14	12	−22	−3	1
Z	4	12	−9	−18	−15	17	−13	−20	9	−5

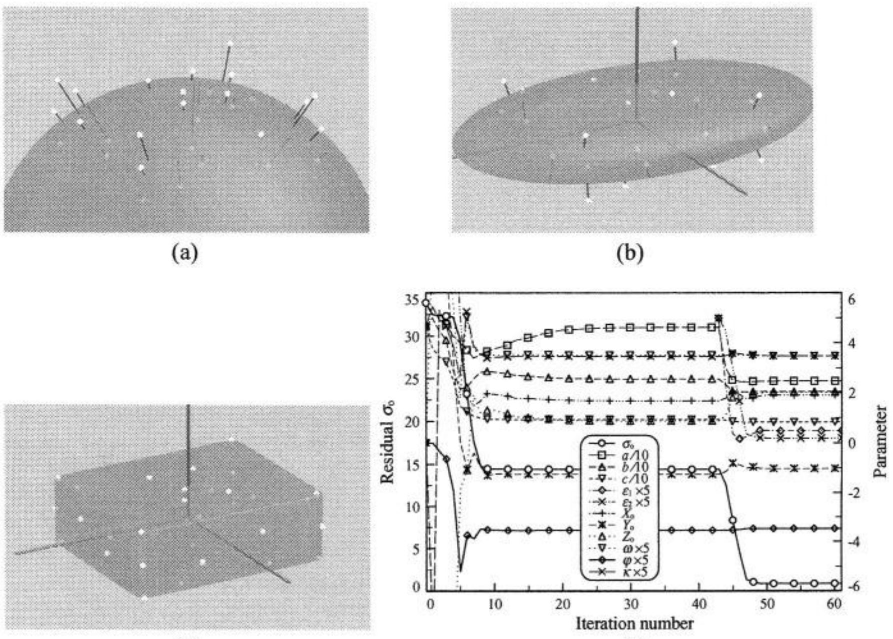


Fig. 3.10. Orthogonal distance fitting to the set of points in Table 3.7: (a) Sphere fit; (b) Ellipsoid fit; (c) Superellipsoid fit; (d) Convergence of the fit. Iteration number 0–42: ellipsoid and 43–: superellipsoid fit

extreme shapes of superellipsoid such as a box with $\varepsilon_{1,2} \ll 1$ or a star with $\varepsilon_{1,2} > 2$. After a series of experiments with numerous sets of data points, we have concluded that our algorithms safely converge with the parameter zone

$$0.002 \le (\varepsilon_1 \text{ and } \varepsilon_2) \le 10 \qquad \text{and} \qquad \varepsilon_1/\varepsilon_2 \le 500\ .$$

Otherwise, there is a danger of data overflow destroying the Jacobian matrices of the linear equation systems (2.27), (2.31), (3.6), and (3.9). Moreover, a sharp increase in computing cost associated with finding the minimum distance points is

Table 3.8. Results of orthogonal distance fitting to the set of points in Table 3.7

Parameter $\hat{\mathbf{a}}$	σ_0	a	b	c	ε_1	ε_2
Sphere	33.8999	46.5199	—	—	—	—
$\sigma(\hat{\mathbf{a}})$	—	9.6364	—	—	—	—
Ellipsoid	14.4338	46.3303	25.5975	9.3304	—	—
$\sigma(\hat{\mathbf{a}})$	—	13.4393	2.7616	0.9553	—	—
Superellipsoid	0.9033	24.6719	20.4927	8.2460	0.0946	0.0374
$\sigma(\hat{\mathbf{a}})$	—	0.1034	0.1026	0.0598	0.0151	0.0197

Parameter $\hat{\mathbf{a}}$	X_o	Y_o	Z_o	ω	φ	κ
Sphere	27.3955	18.2708	−20.8346	—	—	—
$\sigma(\hat{\mathbf{a}})$	7.6529	6.3646	6.8942	—	—	—
Ellipsoid	1.6769	−1.2537	0.8719	0.7016	−0.7099	0.6925
$\sigma(\hat{\mathbf{a}})$	2.3223	1.2274	2.9542	0.0545	0.0403	0.0849
Superellipsoid	1.9096	−1.0234	2.0191	0.6962	−0.6952	0.6960
$\sigma(\hat{\mathbf{a}})$	0.0750	0.0690	0.0774	0.0046	0.0031	0.0059

unavoidable. The initial parameter values are obtained from a sphere fitting and an ellipsoid fitting, successively (Fig. 1.3, Fig. 3.10, Table 3.8). The initial values of shape coefficients are usually set to $\varepsilon_1 = \varepsilon_2 = 1$.

Superellipsoid fitting to the 30 points in Table 3.7 representing a box terminated after 18 iteration cycles for $\|\Delta\mathbf{a}\| = 7.9 \cdot 10^{-7}$ (coordinate-based algorithm, Fig. 3.10d). The intermediate ellipsoid fitting $(a \geq b \geq c)$ showed a slow convergence in its second half-phase (iteration number 10–42 in Fig. 3.10d), because of a relatively large variance in the estimated major axis-length $\hat{a}$ of the ellipsoid caused by the distribution of the given points. In other words, the performance index σ_0^2 of the ellipsoid fitting in Fig. 3.10b is not changed very much by the variation of the estimated $\hat{a}$ (compare the standard deviation $\sigma(\hat{a})$ of the intermediate ellipsoid $\sigma(\hat{a})_{\text{ellip.}} = 13.4393$ with that of the superellipsoid $\sigma(\hat{a})_{\text{SQ}} = 0.1034$ in Table 3.8).

This page intentionally left blank

4. Orthogonal Distance Fitting of Parametric Curves and Surfaces

In Chap. 3, the distance-based algorithm (2.28) and the coordinate-based algorithm (2.32) are generally implemented for implicit features $F(\mathbf{X}, \mathbf{a}) = 0$ according to the variable-separation method (Algorithms II and III, respectively). This chapter describes in detail the general implementation of the three ODF Algorithms I–III (Table 2.3, Sect. 2.3, Table 4.1) for parametric features $\mathbf{X}(\mathbf{a}, \mathbf{u})$. Using the implementations, the model parameters a are grouped and simultaneously estimated in terms of form $\mathbf{a}_g$, position $\mathbf{a}_p$, and rotation parameters $\mathbf{a}_r$ (Sect. 1.1.2, Fig. 4.1). A sub-problem of the ODF of parametric features is the determination of the location parameters $\{\mathbf{u}_i\}_{i=1}^m$ of the minimum distance points $\{\mathbf{X}_i'\}_{i=1}^m$ on the model feature $\mathbf{X}(\mathbf{a}, \mathbf{u})$ from each given point $\{\mathbf{X}_i\}_{i=1}^m$. Algorithm I simultaneously determines the model parameters $\mathbf{a}$ and the location parameters $\{\mathbf{u}_i\}_{i=1}^m$ *(total method)*

$$\min_{\mathbf{a}\in\mathbb{R}^p,\{\mathbf{u}_i\}_{i=1}^m\in\mathbb{R}^2} \sigma_0^2\left(\{\mathbf{X}_i'(\mathbf{a},\mathbf{u})\}_{i=1}^m\right) \ , \tag{4.1}$$

whilst they are alternately determined in a nested iteration scheme by the Algorithms II and III *(variable-separation method)*

$$\min_{\mathbf{a}\in\mathbb{R}^p} \min_{\{\mathbf{u}_i\}_{i=1}^m\in\mathbb{R}^2} \sigma_0^2\left(\{\mathbf{X}_i'(\mathbf{a},\mathbf{u})\}_{i=1}^m\right) \ . \tag{4.2}$$

The implementation of Algorithm I, which is published in literature [74] (Table 1.2, Table 4.1), is described in Sect. 4.2.1. Section 4.2.2 describes the general implementation of Algorithm II *with* the parameter grouping of $\mathbf{a}^\mathrm{T} = (\mathbf{a}_g^\mathrm{T}, \mathbf{a}_p^\mathrm{T}, \mathbf{a}_r^\mathrm{T})$, whereas Algorithm II *without* the parameter grouping is known in literature [25]. The new Algorithm III for parametric features is described in Sect. 4.2.3.

Through the implementations of the Algorithms I–III for parametric features, the minimum distance information $\{d_i\}_{i=1}^m$, $\{\mathbf{X}_i'\}_{i=1}^m$, and their Jacobian matrices $\{\mathbf{J}_{d_i,\mathbf{a}}\}_{i=1}^m$, $\{\mathbf{J}_{\mathbf{X}_i',\mathbf{a}}\}_{i=1}^m$ are obtained from the standard feature description $\mathbf{x}(\mathbf{a}_g, \mathbf{u})$ (1.2) defined in a model coordinate frame xyz and from the coordinate transformation equation $\mathbf{X} = \mathbf{R}^{-1}\mathbf{x} + \mathbf{X}_o$ (1.3). To apply the Algorithms I–III to a new parametric feature, only the provision of the first and the second derivatives of the standard feature description (1.2), which usually has only a few form parameters $\mathbf{a}_g$, are required. The functional interpretation and treatment of the position $\mathbf{a}_p$ and the rotation parameters $\mathbf{a}_r$ concerning the coordinate transformation (1.3) are the same for all parametric features.

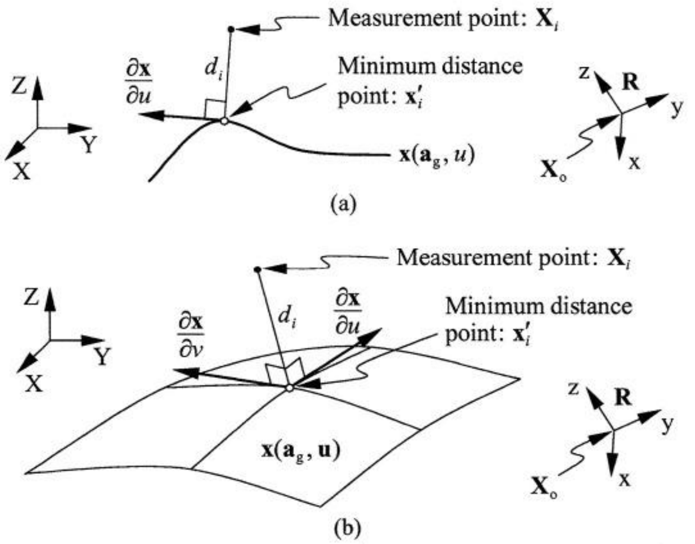

Measurement point: $\mathbf{X}_i$
$\frac{\partial \mathbf{x}}{\partial u}$
d_i
Minimum distance point: $\mathbf{x}'_i$
$\mathbf{x}(\mathbf{a}_g, u)$
Z
Y
X
z
R
y
x
$\mathbf{X}_o$
(a)
Measurement point: $\mathbf{X}_i$
$\frac{\partial \mathbf{x}}{\partial v}$
d_i
$\frac{\partial \mathbf{x}}{\partial u}$
Minimum distance point: $\mathbf{x}'_i$
$\mathbf{x}(\mathbf{a}_g, \mathbf{u})$
(b)

Fig. 4.1. Parametric features and the minimum distance point $\mathbf{x}'_i$ in xyz frame from the given point $\mathbf{X}_i$ in XYZ frame: (a) Curve; (b) Surface

Table 4.1. Orthogonal distance fitting algorithms for parametric features

Approach	Distance-based algorithm	Coordinate-based algorithm
Total method	Underdetermined linear equation system	Algorithm I: ETH [74]
Variable-separation method	Algorithm II: NPL [25], FhG [10]	Algorithm III: FhG [10]

Section 4.1 describes a rapid and robust algorithm for finding the minimum distance point on a parametric feature from a given point. Section 4.2 describes the general implementation of the distance-based algorithm (2.28) (Algorithm II) and the coordinate-based algorithm (2.32) (Algorithms I and III) for parametric features. Various fitting examples are given in Sect. 4.3.

4.1 Minimum Distance Point

Similarly to the case (3.1) of implicit features, the time-consuming part of the variable-separation method (4.2) (Algorithms II and III) is the inner iteration finding the location parameters $\{\mathbf{u}'_i\}_{i=1}^m$ of the minimum distance points $\{\mathbf{X}'_i\}_{i=1}^m$ on a parametric feature from each given point $\{\mathbf{X}_i\}_{i=1}^m$. Also, as with the total method (4.1) (Algorithm I), provided with the initial model parameters $\mathbf{a}_0$, the location parameters $\{\mathbf{u}'_i\}_{i=1}^m$ of the minimum distance points $\{\mathbf{X}'_i\}_{i=1}^m$ are used as the initial values for $\{\mathbf{u}_i\}_{i=1}^m$ of the iteration (4.1). For these reasons, an efficient and robust algorithm for finding the location parameters $\{\mathbf{u}_i'\}_{i=1}^m$ is essential to the ODF of parametric features in the case of both the total method (4.1) and the variable-separation method (4.2).

For a given point $\mathbf{x}_i$ in xyz frame (Fig. 4.1)

$$\mathbf{x}_i = \mathbf{R}(\mathbf{X}_i - \mathbf{X}_\mathrm{o}) \,, \tag{4.3}$$

the location parameters $\mathbf{u}_i'$ of the minimum distance point $\mathbf{x}_i'$ are determined on the standard model feature $\mathbf{x}(\mathbf{a}_\mathrm{g}, \mathbf{u})$ (1.2) defined in xyz frame. Then the minimum distance point $\mathbf{X}_i'$ in XYZ frame from the given point $\mathbf{X}_i$ can be obtained through the backward transformation of $\mathbf{x}_i'$ into XYZ frame

$$\mathbf{X}_i' = \mathbf{R}^{-1}\mathbf{x}_i' + \mathbf{X}_\mathrm{o} \,.$$

For some parametric features such as a circle, sphere, cylinder, cone, and torus, the location parameters $\mathbf{u}_i'$ can be determined in closed form at low computing cost. However, in general, $\mathbf{u}_i'$ must be found iteratively satisfying appropriate minimum distance conditions. In this section, first, the Newton method is described (Sect. 4.1.1) for finding the location parameter $\mathbf{u}_i'$. Then, the Newton method is enhanced by applying the Levenberg-Marquardt algorithm [50], [52] (Sect. 4.1.2) which permits a broader convergence domain than the Newton method. In the following subsections, the location parameters of the minimum distance point are marked with prime, as u_i' with curve and as $\mathbf{u}_i'$ with surface, once they have been determined and are ready for use. Otherwise, they are notated with u or $\mathbf{u}$.

4.1.1 Newton Method

The minimum distance problem with parametric features can be solved using the Newton method [47], [62], [77]. Our aim was to find the location parameters $\mathbf{u}$ that minimize the error distance d_i between the given point $\mathbf{x}_i$ and the corresponding point $\mathbf{x}$ on the standard model feature $\mathbf{x}(\mathbf{a}_\mathrm{g}, \mathbf{u})$ (1.2)

$$\begin{aligned} D(\mathbf{u}) &\triangleq d_i^2 \\ &= \|\mathbf{x}_i - \mathbf{x}(\mathbf{a}_\mathrm{g}, \mathbf{u})\|^2 \\ &= (\mathbf{x}_i - \mathbf{x}(\mathbf{a}_\mathrm{g}, \mathbf{u}))^\mathrm{T}(\mathbf{x}_i - \mathbf{x}(\mathbf{a}_\mathrm{g}, \mathbf{u})) \,. \end{aligned} \tag{4.4}$$

The first order necessary condition for a minimum of (4.4) as a function of $\mathbf{u}$ is

$$\begin{aligned} \mathbf{f}(\mathbf{x}_i, \mathbf{x}(\mathbf{a}_\mathrm{g}, \mathbf{u})) &\triangleq \frac{1}{2}\begin{pmatrix} \partial D(\mathbf{u})/\partial u \\ \partial D(\mathbf{u})/\partial v \end{pmatrix} \\ &= -\begin{pmatrix} (\mathbf{x}_i - \mathbf{x}(\mathbf{a}_\mathrm{g}, \mathbf{u}))^\mathrm{T}\mathbf{x}_u \\ (\mathbf{x}_i - \mathbf{x}(\mathbf{a}_\mathrm{g}, \mathbf{u}))^\mathrm{T}\mathbf{x}_v \end{pmatrix} = \mathbf{0} \,. \end{aligned} \tag{4.5}$$

Equation (4.5) denotes that the distance vector $(\mathbf{x}_i - \mathbf{x})$ and the feature tangent vectors $\partial\mathbf{x}/\partial\mathbf{u}$ at $\mathbf{x}$ should be orthogonal (Fig. 4.1). Equation (4.5) is solved for

$\mathbf{u}$ using the Newton method (how to derive the Hessian matrix $\partial\mathbf{f}/\partial\mathbf{u}$ is shown in Sect. 4.2.3)

$$\left.\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\right|_k \Delta\mathbf{u} = -\mathbf{f}(\mathbf{u})|_k \ , \qquad \mathbf{u}_{k+1} = \mathbf{u}_k + \alpha\Delta\mathbf{u} \ . \tag{4.6}$$

4.1.2 Levenberg-Marquardt Algorithm

The Newton method is known to converge fast at a quadratic rate [33]. A drawback of the Newton method is the relatively narrow convergence domain $\boxed{\text{B}}$, as illustrated in Fig. 4.2 and Table 4.2 for the case of a univariate function $D(u)$. Because both the minimum and maximum points of $D(u)$ satisfy the first order necessary condition $\partial D(u)/\partial u = 0$ (4.5) and could pull the Newton direction Δu, an additional condition (namely, the second order necessary condition) is required for the convergence of the iteration (4.6) to a local *minimum* point of $D(u)$ that the current

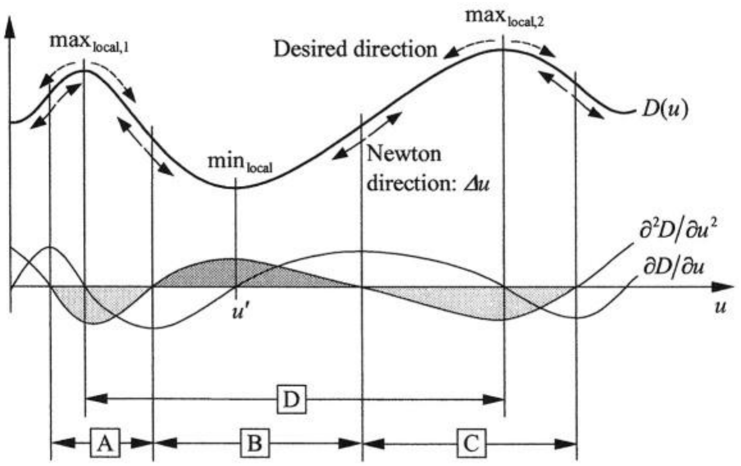


Fig. 4.2. Search for the local minimum point u' of a function $D(u)$

Table 4.2. Behavior of the Newton direction Δu in Fig. 4.2

$\Delta u = -\dfrac{\partial D/\partial u}{\partial^2 D/\partial u^2}$		$\partial^2 D/\partial u^2$		$D(u)$
		+	−	
$\partial D/\partial u$	+	−	+	Increase
	−	+	−	Decrease
$D(u)$		Concave: $\boxed{\text{B}}$	Convex: $\boxed{\text{A}}$, $\boxed{\text{C}}$	Desired: $\boxed{\text{D}}$

point u_k must lie inside the concave zone of $D(u)$ where $\partial^2 D/\partial u^2 > 0$ ($\boxed{\text{B}}$ in Fig. 4.2). Otherwise ($\boxed{\text{A}}$, $\boxed{\text{C}}$), the Newton iteration (4.6) directs toward a local *maximum* point of $D(u)$ or, in a worse case, diverges. Thus, in order to broaden the convergence domain from $\boxed{\text{B}}$ to $\boxed{\text{D}}$ in a straight manner, the sign of the Newton direction Δu is made dependent only on the sign of the slope $\partial D(u)/\partial u$ of $D(u)$ by blinding the sign of $\partial^2 D/\partial u^2$ as below:

$$\Delta u = -\left.\frac{\partial D/\partial u}{|\partial^2 D/\partial u^2| + t}\right|_{u_k} \qquad \text{with} \qquad t > 0\,,$$

where the small positive number t prevents the parameter update Δu from overshooting near the turning point of concave/convex where $|\partial^2 D/\partial u^2| \approx 0$.

A more robust and generally extendable method for overcoming the narrow convergence domain of the Newton method, also through making the sign of Δu dependent only on the sign of the slope $\partial D(u)/\partial u$, is to elevate the second derivative curve $\partial^2 D/\partial u^2$ in Fig. 4.2 by adding a sufficiently large positive number λ whenever the second derivative is negative as $\partial^2 D/\partial u^2 < 0$ at the current point u_k

$$\Delta u = -\left.\frac{\partial D/\partial u}{\partial^2 D/\partial u^2 + \lambda}\right|_{u_k} \qquad \text{with} \qquad \lambda > \left|\partial^2 D/\partial u^2\right|_{u_k} . \tag{4.7}$$

The basic idea in (4.7) is nothing other than that in the Levenberg-Marquardt algorithm [50], [52] for the multivariate minimization problem below:

$$\left.\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}} + \lambda \mathbf{I}\right)\right|_k \Delta \mathbf{u} = -\mathbf{f}(\mathbf{u})|_k\,, \qquad \mathbf{u}_{k+1} = \mathbf{u}_k + \alpha \Delta \mathbf{u}\,. \tag{4.8}$$

There are a variety of methods for determining (controlling) the correct size of the positive number λ in (4.8) [33], [64]. Because the search domain of the location

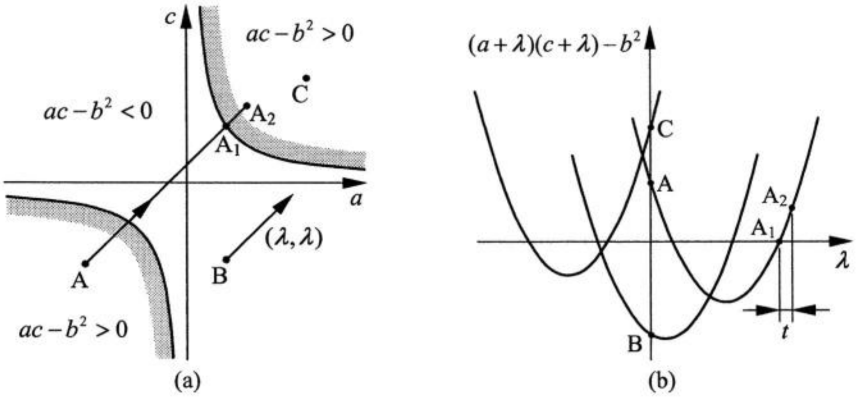


Fig. 4.3. Determination of λ for the Levenberg-Marquardt algorithm (4.8): (a) Determinants of the Hessian matrix (4.9) of the Newton method (4.6). A: $ac - b^2 > 0$ but $a \le 0$, B: $ac - b^2 \le 0$, C: $ac - b^2 > 0$ and $a > 0$; (b) Determination of λ forcing $(a+\lambda)(c+\lambda) - b^2 > 0$ and $(a + \lambda) > 0$

parameters for the minimum distance point on a parametric curve/surface has a maximum of two variables (univariate u with curve and bivariate $\mathbf{u}$ with surface), the application of the Levenberg-Marquardt algorithm (4.8) to the minimum distance problem in this section is relatively straightforward. Whenever the Hessian matrix of the Newton method (4.6) is not positive definite at the current point $\mathbf{u}_k$ (as at A or B in Fig. 4.3a)

$$\begin{pmatrix} a & b \\ b & c \end{pmatrix} \triangleq \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{u}_k} \quad \text{with} \quad \begin{cases} a \le 0 & & : \textbf{univariate} \\ a \le 0 \quad \text{or} \quad ac - b^2 \le 0 & & : \textbf{bivariate,} \end{cases} \tag{4.9}$$

we provide (4.8) with the positive number λ below (Fig. 4.3b):

$$\lambda = \begin{cases} -a + t & : \textbf{univariate} \\ -\frac{1}{2}\left(a + c - \sqrt{(a-c)^2 + 4b^2}\right) + t & : \textbf{bivariate.} \end{cases}$$

In practice, the small positive number $t = 1.0$ works well. When the Hessian matrix of the Newton method (4.6) is positive definite as at C in Fig. 4.3a

$$\begin{cases} a > 0 & : \textbf{univariate} \\ a > 0 \quad \text{and} \quad ac - b^2 > 0 & : \textbf{bivariate,} \end{cases}$$

the Levenberg-Marquardt algorithm is deactivated and the Newton method used by setting $\lambda = 0$ in (4.8). The Newton method (4.6) supported by the Levenberg-Marquardt algorithm (4.8) shows fast and robust convergence to a local minimum point even if the current point $\mathbf{u}_k$ lies remotely from the local minimum point. In this chapter, if not explicitly mentioned, the Newton method (4.6) implies the use of the Levenberg-Marquardt algorithm (4.8) whenever the condition (4.9) is invoked.

4.1.3 Initial Values

With the minimum distance problem, there is at least one local minimum distance point on a model feature from a given point (in other words, the problem is solvable eventually), because the distance function (4.4) is always non-negative (i.e., lower-bounded) by nature. On the other hand, generally speaking as far as minimization problems are concerned, only the *local* minimum points can be located, whereas there is no sufficient condition that a local minimum point found is just the *global* one. The only practical method is to solve the minimum distance problem from a number of different starting points and then use the best solution obtained [33]. Such strategies as the linear search and the clustering method [77] for selecting the number and distribution of the initial points are important for the effective search for the global minimum distance point. Because the Newton method (4.6) supported by the Levenberg-Marquardt algorithm (4.8) shows fast and robust convergence to a local minimum point even if the initial point $\mathbf{u}_k$ lies remotely from the local minimum point, only a sparse distribution of the initial points is required.

Moreover, with many well-known model features, the minimum distance problem can be solved in closed form or a single initial point $\mathbf{u}_k$ can be supplied from which the iteration (4.6) converges to the global minimum point.

With some simple model features having a linear/circular section such as a circle, sphere, cylinder, cone or torus, the minimum distance problem can be solved in closed form. For example, with a 3-D circle defined in xyz frame

$$\mathbf{x}(r,u) \triangleq (r\cos u,\ r\sin u,\ 0)^{\mathrm{T}} \qquad \text{with} \qquad -\pi < u \leq \pi\ ,$$

the location parameter value u_i' of the global minimum distance point $\mathbf{x}_i'$ on the 3-D circle from a given point $\mathbf{x}_i$ is

$$u_i' = \tan^{-1}\frac{y_i}{x_i}\ . \tag{4.10}$$

With other model features of a more general shape, the location parameters $\mathbf{u}_i'$ have to be found using iteration (4.6) starting from a number of different initial points. Fortunately, with many well-known model features having axis or plane symmetry (e.g., ellipse, parabola, ellipsoid, paraboloid, elliptic cone, etc.), the two points $\mathbf{x}_i'$ and $\mathbf{x}_i$ lie in the same half-plane/space of xyz frame. From this fact, the appropriate domain of the location parameters, from which the iteration (4.6) converges to the global minimum distance point, is selected. For example, with a 3-D ellipse defined in xyz frame

$$\mathbf{x}(a,b,u) \triangleq (a\cos u,\ b\sin u,\ 0)^{\mathrm{T}} \qquad \text{with} \qquad a \geq b, \quad -\pi < u \leq \pi\ ,$$

the initial location parameter value u_k that leads the iteration (4.6) to the global minimum distance point can be obtained as shown below:

$$u_k = \begin{cases} \pi/2 & : y_i \geq 0 \\ -\pi/2 & : y_i < 0\ . \end{cases} \tag{4.11}$$

Furthermore, the closed form solution of the minimum distance point on a *circular* feature such as a circle, sphere, cylinder can be used as reasonable initial values for the iterative search (4.6) for the global minimum distance point on the corresponding *elliptical* feature (i.e., ellipse, ellipsoid, elliptic cylinder). In other words, (4.10) is used instead of (4.11) as the initial value for finding the global minimum distance point on a 3-D ellipse.

Finally, a method is given as to how to handle the singular point of a parametric feature where the feature tangents vanish and/or are parallel to each other, as $\mathbf{x}_u = \mathbf{0}$ with curve and as $\mathbf{x}_u \times \mathbf{x}_v = \mathbf{0}$ with surface [27]. An example of a singular point is the north/south pole of an ellipsoid where $u =$ undefined, $v = \pm\pi/2$, and $\mathbf{x}_u = \mathbf{0}$,

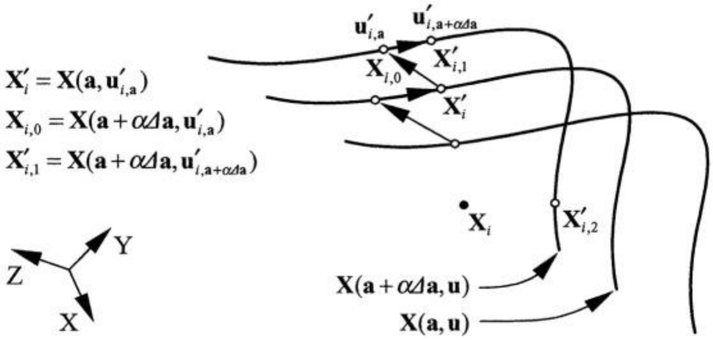
$$\mathbf{x}(a,b,c,\mathbf{u}) \triangleq (a\cos u\cos v,\ b\sin u\cos v,\ c\sin v)^{\mathrm{T}}\ .$$

The tangents at the north/south pole of an ellipsoid are well-defined in geometric space xyz but not in parametric space uv (i.e., the north/south pole of an ellipsoid has G^1 continuity but not C^1 continuity). When *not* supported by the Levenberg-Marquardt algorithm (4.8), the Newton method (4.6) is prone to converge to a singular point if the current point lies near the singular point, even though the singular point might not be a local minimum distance point. Supported by the Levenberg-Marquardt algorithm, the Newton method has no serious problems with singular points. Nevertheless, once the iteration (4.6) is caught by a singular point (e.g., by a curve point with $\mathbf{x}_u = \mathbf{0}$), no further update of the location parameters from (4.6) takes place. Here, in order to give a chance for (4.6) to escape from the singular point, the current location parameters (of the singular point) are perturbed by adding small random errors and then the iteration (4.6) will be restarted. When (4.6) repeatedly falls into the singular point, this may be accepted as a local minimum distance point.

4.1.4 Acceleration of Finding the Minimum Distance Point

The computing cost of finding the global minimum distance point by starting from a number of different initial points is generally high, even using a search strategy such as the clustering method. Fortunately, the location parameters $\mathbf{u}'_i$ of the minimum distance point change their values highly incrementally between two consecutive outer iteration cycles in the second half-phase of the outer iteration of (4.2) where the parameter update $\Delta\mathbf{a}$ will have been moderate. Thus, to save the computing cost in a similar manner as with the case of implicit features (Sect. 3.1.4), the resultant location parameters $\mathbf{u}'_i$ from the inner iteration inside the last outer iteration are used as the reasonable initial values for the current inner iteration

$$\mathbf{u}_{i,\mathbf{a}+\alpha\Delta\mathbf{a}} = \mathbf{u}'_{i,\mathbf{a}} \; . \tag{4.12}$$



$\mathbf{u}'_{i,\mathbf{a}}$
$\mathbf{u}'_{i,\mathbf{a}+\alpha\Delta\mathbf{a}}$
$\mathbf{X}'_{i,1}$
$\mathbf{X}_{i,0}$
$\mathbf{X}'_i$
$\mathbf{X}'_i = \mathbf{X}(\mathbf{a}, \mathbf{u}'_{i,\mathbf{a}})$
$\mathbf{X}_{i,0} = \mathbf{X}(\mathbf{a}+\alpha\Delta\mathbf{a}, \mathbf{u}'_{i,\mathbf{a}})$
$\mathbf{X}'_{i,1} = \mathbf{X}(\mathbf{a}+\alpha\Delta\mathbf{a}, \mathbf{u}'_{i,\mathbf{a}+\alpha\Delta\mathbf{a}})$
$\mathbf{X}_i$
$\mathbf{X}'_{i,2}$
Y
Z
X
$\mathbf{X}(\mathbf{a}+\alpha\Delta\mathbf{a}, \mathbf{u})$
$\mathbf{X}(\mathbf{a}, \mathbf{u})$

Fig. 4.4. Minimum distance points $\mathbf{X}'_{i,1}$ and $\mathbf{X}'_{i,2}$ on $\mathbf{X}(\mathbf{a}+\alpha\Delta\mathbf{a},\mathbf{u})$ (updated parametric feature) from the given point $\mathbf{X}_i$. Iterative search for the minimum distance point may converge to $\mathbf{X}'_{i,1}$, if the iteration (4.6) started from the location parameter $\mathbf{u}'_{i,\mathbf{a}}$ of the minimum distance point $\mathbf{X}'_i$ on $\mathbf{X}(\mathbf{a},\mathbf{u})$ (see (4.12)), while $\mathbf{X}'_{i,2}$ is the global minimum distance point being $\|\mathbf{X}_i - \mathbf{X}'_{i,1}\| > \|\mathbf{X}_i - \mathbf{X}'_{i,2}\|$

Also, in order to overcome the pitfall of using (4.12) (Fig. 4.4), similarly as with the case of implicit features (see Fig. 3.5), the location parameters $\mathbf{u}_i'$ are periodically refreshed during the second half-phase of the outer iteration by starting the iteration (4.6) from the initial points described in Sect. 4.1.3.

The strategy for an efficient and inexpensive finding of the location parameters $\mathbf{u}_i'$ for the minimum distance point on a parametric feature is summarized below:

- Use the closed form solution of $\mathbf{u}_i'$, if available
- In the first half-phase of the outer iteration of (4.2), start the inner iteration (4.6) from the initial points described in Sect. 4.1.3
- In the second half-phase of the outer iteration, start the inner iteration from the initial point (4.12)
- Periodically, in the second half-phase of the outer iteration, start the inner iteration from the initial points described in Sect. 4.1.3.

4.2 Orthogonal Distance Fitting

4.2.1 Algorithm I (ETH)

Algorithm I (ETH) [34], [74] is based on the performance index (2.21) and simultaneously estimates the model parameters **a** and the location parameters $\{\mathbf{u}_i\}_{i=1}^m$ (*total method,* see (4.1)). The new estimation parameter vector **b** is defined consisting of **a** and $\{\mathbf{u}_i\}_{i=1}^m$

$$\begin{aligned}\mathbf{b}^{\mathrm{T}} &\triangleq (\mathbf{a}_{\mathrm{g}}^{\mathrm{T}}, \mathbf{a}_{\mathrm{p}}^{\mathrm{T}}, \mathbf{a}_{\mathrm{r}}^{\mathrm{T}}, \mathbf{u}_1^{\mathrm{T}}, \ldots, \mathbf{u}_m^{\mathrm{T}}) \\ &= (a_1, \ldots, a_p, u_1, v_1, \ldots, u_m, v_m)\ .\end{aligned} \tag{4.13}$$

Using a procedure similar to (2.29)–(2.31) carried out in Sect. 2.3.2, the Gauss-Newton iteration for determining the extended parameters **b** (4.13) which minimize the performance index (2.21) can be constructed

$$\mathbf{P}\frac{\partial \mathbf{X}'}{\partial \mathbf{b}}\bigg|_k \Delta\mathbf{b} = \mathbf{P}(\mathbf{X} - \mathbf{X}')|_k\ , \qquad \mathbf{b}_{k+1} = \mathbf{b}_k + \alpha\Delta\mathbf{b}\ . \tag{4.14}$$

In order to complete the linear equation system (4.14), the Jacobian matrices $\{\mathbf{J}_{\mathbf{X}_i',\mathbf{b}}\}_{i=1}^m$ of each corresponding point $\{\mathbf{X}_i'\}_{i=1}^m$ on the model feature are required. To note is that $\{\mathbf{X}_i'\}_{i=1}^m$ are not necessarily the minimum distance points from the given points $\{\mathbf{X}_i\}_{i=1}^m$. From $\mathbf{x}(\mathbf{a}_{\mathrm{g}}, \mathbf{u})$ (1.2) and $\mathbf{X} = \mathbf{R}^{-1}\mathbf{x} + \mathbf{X}_{\mathrm{o}}$ (1.3), the following can be derived:

$$\begin{aligned}
\mathbf{J}_{\mathbf{X}'_i,\mathbf{b}} &= \left.\frac{\partial \mathbf{X}}{\partial \mathbf{b}}\right|_{\mathbf{X}=\mathbf{X}'_i} \\
&= \left.\frac{\partial}{\partial \mathbf{b}}\left(\mathbf{R}^{\mathrm{T}}\mathbf{x}(\mathbf{a}_{\mathrm{g}},\mathbf{u})+\mathbf{X}_{\mathrm{o}}\right)\right|_{\mathbf{u}=\mathbf{u}_i} \\
&= \left.\left(\mathbf{R}^{\mathrm{T}}\frac{\partial \mathbf{x}}{\partial \mathbf{b}}+\frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{b}}\mathbf{x}+\frac{\partial \mathbf{X}_{\mathrm{o}}}{\partial \mathbf{b}}\right)\right|_{\mathbf{u}=\mathbf{u}_i} \\
&= \left.\left(\underbrace{\underbrace{\mathbf{R}^{\mathrm{T}}\frac{\partial \mathbf{x}}{\partial \mathbf{a}_{\mathrm{g}}}}_{n\times l}\,\Bigg|\,\underbrace{\mathbf{I}}_{n\times n}\,\Bigg|\,\underbrace{\frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}_{\mathrm{r}}}[\mathbf{x}]}_{n\times s}}_{n\times p}\,\Bigg|\,\underbrace{\mathbf{0}_1,\cdots,\mathbf{0}_{i-1},\mathbf{R}^{\mathrm{T}}\frac{\partial \mathbf{x}}{\partial \mathbf{u}},\mathbf{0}_{i+1},\cdots,\mathbf{0}_m}_{\substack{n\times m \quad \text{: curve (univariate)}\\ n\times 2m \quad \text{: surface (bivariate)}}}\right)\right|_{\mathbf{u}=\mathbf{u}_i} \\
&\triangleq \left.\left(\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}}\,\Big|\,\mathbf{0}_1,\cdots,\mathbf{0}_{i-1},\mathbf{J}_{\mathbf{X}'_i,\mathbf{u}},\mathbf{0}_{i+1},\cdots,\mathbf{0}_m\right)\right|_{\mathbf{u}=\mathbf{u}_i} .
\end{aligned} \tag{4.15}$$

The unfolded form of the linear equation system (4.14) is shown below and can be solved for the parameter update $\Delta\mathbf{b}$:

$$\mathbf{P}\underbrace{\begin{pmatrix}\mathbf{J}_{\mathbf{X}'_1,\mathbf{a}} & \mathbf{J}_{\mathbf{X}'_1,\mathbf{u}} & & \mathbf{0}\\ \vdots & & \ddots & \\ \mathbf{J}_{\mathbf{X}'_m,\mathbf{a}} & \mathbf{0} & & \mathbf{J}_{\mathbf{X}'_m,\mathbf{u}}\end{pmatrix}}_{\substack{nm\times(p+m) \quad \text{: curve (univariate)}\\ nm\times(p+2m) \quad \text{: surface (bivariate)}}}\begin{pmatrix}\Delta\mathbf{a}\\ \Delta\mathbf{u}_1\\ \vdots\\ \Delta\mathbf{u}_m\end{pmatrix} = \mathbf{P}\underbrace{\begin{pmatrix}\mathbf{X}_1-\mathbf{X}'_1\\ \vdots\\ \mathbf{X}_m-\mathbf{X}'_m\end{pmatrix}}_{nm\times 1} . \tag{4.16}$$

The advantage of Algorithm I is the low implementation cost for application to a new parametric feature, because merely the provision of (4.15) is required with only the first derivatives $\partial\mathbf{x}/\partial\mathbf{u}$ and $\partial\mathbf{x}/\partial\mathbf{a}_{\mathrm{g}}$ of the standard feature description $\mathbf{x}(\mathbf{a}_{\mathrm{g}},\mathbf{u})$ (1.2) of the new parametric feature. A disadvantage of Algorithm I is that memory space and computing cost increase rapidly with the number of data points unless the sparse system (4.16) is handled beforehand by a sparse matrix algorithm [38] (assuming $m \gg p$ and the use of the SVD for solving (4.16), memory space usage and computing cost are proportional to m^2 and m^3, respectively (see Table 2.4)).

The convergence of the iteration (4.16) is highly affected by the selection of the initial values for the location parameters $\{\mathbf{u}_i\}_{i=1}^m$. Provided with the initial model parameters $\mathbf{a}_0$, the best initial values for $\{\mathbf{u}_i\}_{i=1}^m$ would be the location parameters $\{\mathbf{u}'_i\}_{i=1}^m$ of the minimum distance points. Also, to interrupt the unintended development illustrated in Fig. 4.4, which also occurs with Algorithm I in a similar manner, the algorithms for finding the minimum distance point should be used to periodically refresh the location parameters $\{\mathbf{u}_i\}_{i=1}^m$ during the iteration (4.16). For these reasons, Algorithm I should still be assisted by the algorithm for finding the minimum distance points which requires the second derivatives $\partial^2\mathbf{x}/\partial\mathbf{u}^2$ of $\mathbf{x}(\mathbf{a}_{\mathrm{g}},\mathbf{u})$

(Sect. 4.1). Due to this real practice, the advantage of Algorithm I noted above with regard to implementation costs would be partially invalidated.

With the existence of the minimum distance condition (4.5), which is one of the primary criteria for ODF to be strictly fulfilled, the location parameters $\{\mathbf{u}_i\}_{i=1}^m$ are dependent on the model parameters $\mathbf{a}$

$$\frac{\partial \mathbf{u}_i}{\partial \mathbf{a}} \neq \mathbf{0}\,, \qquad i = 1, \ldots, m\,. \tag{4.17}$$

In other words, provided with the model parameters $\mathbf{a}$ and the given points $\{\mathbf{X}_i\}_{i=1}^m$, the location parameters $\{\mathbf{u}_i\}_{i=1}^m$ are to be determined eventually from the minimum distance condition (4.5) (Sect. 4.1). However, parameterization (4.13) and the linear equation system (4.16) ignore the parameter dependencies (4.17) and treat both $\mathbf{a}$ and $\{\mathbf{u}_i\}_{i=1}^m$ as independent variables. If the current model parameters $\mathbf{a}_k$ are not close enough to the final estimation $\hat{\mathbf{a}}$ during the iteration (4.16), the updates $\{\Delta\mathbf{u}_i\}_{i=1}^m$ of the location parameters are large enough to head for a divergence domain (parameter overshooting). Consequently, the step size factor α is forced to be small enough to hold the updated location parameters $\{\mathbf{u}_i + \alpha\Delta\mathbf{u}_i\}_{i=1}^m$ in a convergence domain. However, the small α stabilizes the iteration (4.16) at the cost of a side-effect that the resulting update $\alpha\Delta\mathbf{a}$ of the model parameters becomes unnecessarily small, causing a slow overall convergence of the iteration (4.16) (see Fig. 4.8a). Whilst the parameter dependencies (4.17) are overlooked by Algorithm I, they are properly suppressed in product form with (4.5) by Algorithm II (Sect. 4.2.2) or are derived from (4.5) and fully utilized by Algorithm III (Sect. 4.2.3).

4.2.2 Algorithm II (NPL, FhG)

Algorithm II is based on the performance index (2.20) and alternately estimates the parameters $\mathbf{a}$ and $\{\mathbf{u}_i\}_{i=1}^m$ (*variable-separation method,* see (4.2)). The inner iteration of (4.2) determines the location parameters $\{\mathbf{u}_i'\}_{i=1}^m$ for the minimum distance points $\{\mathbf{X}_i'\}_{i=1}^m$ on the current model feature from each given point $\{\mathbf{X}_i\}_{i=1}^m$ (Sect. 4.1). The outer iteration updates the model parameters $\mathbf{a}$. In this section, the general implementation of the distance-based algorithm (2.28) is described which updates the model parameters $\mathbf{a}$ minimizing the performance index (2.20). In order to complete and solve the linear equation system (2.28) for the parameter update $\Delta\mathbf{a}$, provided with the minimum distances $\{d_i\}_{i=1}^m$ from the inner iteration of (4.2), the Jacobian matrices $\{\mathbf{J}_{d_i,\mathbf{a}}\}_{i=1}^m$ of each minimum distance $\{d_i\}_{i=1}^m$ must be supplied.

With the initial NPL algorithm [25], [77], the parametric feature is defined in a coordinate frame XYZ *without* parameter grouping in terms of form, position, and rotation parameters (model parameters $\mathbf{b}$ *without* parameter grouping are differentiated from model parameters $\mathbf{a}$ *with* parameter grouping)

$$\mathbf{X}(\mathbf{b}, \mathbf{u}) \qquad \text{with} \qquad \mathbf{b}^{\mathrm{T}} = (b_1, \ldots, b_p)\,. \tag{4.18}$$

From $d_i = \|\mathbf{X}_i - \mathbf{X}_i'\|$ and (4.18), the Jacobian matrices $\{\mathbf{J}_{d_i,\mathbf{b}}\}_{i=1}^m$ of each *minimum distance* d_i are derived:

$$\begin{aligned}\mathbf{J}_{d_i,\mathbf{b}} &= \frac{\partial d_i}{\partial \mathbf{b}} \\ &= -\frac{(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}}{\|\mathbf{X}_i - \mathbf{X}'_i\|}\left(\frac{\partial \mathbf{X}}{\partial \mathbf{b}} + \frac{\partial \mathbf{X}}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{b}}\right)\Bigg|_{\mathbf{u}=\mathbf{u}'_i} .\end{aligned} \tag{4.19}$$

With the minimum distance condition (4.5) at $\mathbf{u} = \mathbf{u}'_i$, which is adapted for the distance function $D(\mathbf{u}) = \|\mathbf{X}_i - \mathbf{X}(\mathbf{b},\mathbf{u})\|^2$ defined in XYZ frame, the Jacobian matrix $\mathbf{J}_{d_i,\mathbf{b}}$ (4.19) can be simplified as follows:

$$(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}\frac{\partial \mathbf{X}}{\partial \mathbf{u}}\bigg|_{\mathbf{u}=\mathbf{u}'_i} = \mathbf{0}^{\mathrm{T}} ,$$

thus,

$$\mathbf{J}_{d_i,\mathbf{b}} = -\underbrace{\frac{(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}}{\|\mathbf{X}_i - \mathbf{X}'_i\|}}_{1\times n}\underbrace{\frac{\partial \mathbf{X}}{\partial \mathbf{b}}}_{n\times p}\Bigg|_{\mathbf{u}=\mathbf{u}'_i} . \tag{4.20}$$

Now, provided with $\{\mathbf{J}_{d_i,\mathbf{b}}\}_{i=1}^m$ and $\{d_i\}_{i=1}^m$, the linear equation system (2.28) is completed and can be solved for the parameter update $\Delta\mathbf{b}$.

In order to implement the parameter grouping of $\mathbf{a}^{\mathrm{T}} = (\mathbf{a}_{\mathrm{g}}^{\mathrm{T}}, \mathbf{a}_{\mathrm{p}}^{\mathrm{T}}, \mathbf{a}_{\mathrm{r}}^{\mathrm{T}})$, a highly desirable algorithmic feature for applications, the initial NPL algorithm is modified into a new algorithm [10]. From $d_i = \|\mathbf{X}_i - \mathbf{X}'_i\|$, $\mathbf{x}(\mathbf{a}_{\mathrm{g}},\mathbf{u})$ (1.2), and $\mathbf{X} = \mathbf{R}^{-1}\mathbf{x} + \mathbf{X}_{\mathrm{o}}$ (1.3), the Jacobian matrices $\{\mathbf{J}_{d_i,\mathbf{a}}\}_{i=1}^m$ of each *minimum distance* d_i are derived:

$$\begin{aligned}\mathbf{J}_{d_i,\mathbf{a}} &= \frac{\partial d_i}{\partial \mathbf{a}} \\ &= -\frac{(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}}{\|\mathbf{X}_i - \mathbf{X}'_i\|}\frac{\partial \mathbf{X}}{\partial \mathbf{a}}\bigg|_{\mathbf{u}=\mathbf{u}'_i} \\ &= -\frac{(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}}{\|\mathbf{X}_i - \mathbf{X}'_i\|}\left(\mathbf{R}^{\mathrm{T}}\left(\frac{\partial \mathbf{x}}{\partial \mathbf{a}} + \frac{\partial \mathbf{x}}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{a}}\right) + \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}}\mathbf{x} + \frac{\partial \mathbf{X}_{\mathrm{o}}}{\partial \mathbf{a}}\right)\Bigg|_{\mathbf{u}=\mathbf{u}'_i} .\end{aligned} \tag{4.21}$$

With (1.3) and (4.5) at $\mathbf{u} = \mathbf{u}'_i$, the Jacobian matrix $\mathbf{J}_{d_i,\mathbf{a}}$ (4.21) can be simplified as follows:

$$\begin{aligned}(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\bigg|_{\mathbf{u}=\mathbf{u}'_i} &= (\mathbf{x}_i - \mathbf{x}'_i)^{\mathrm{T}}\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\bigg|_{\mathbf{u}=\mathbf{u}'_i} \\ &= \mathbf{0}^{\mathrm{T}} ,\end{aligned}$$

thus,

$$\mathbf{J}_{d_i,\mathbf{a}} = -\underbrace{\frac{(\mathbf{X}_i - \mathbf{X}_i')^{\mathrm{T}}}{\|\mathbf{X}_i - \mathbf{X}_i'\|}}_{1\times n} \left(\underbrace{\mathbf{R}^{\mathrm{T}}\frac{\partial \mathbf{x}}{\partial \mathbf{a}_{\mathrm{g}}}}_{n\times l}\bigg|_{\mathbf{u}=\mathbf{u}_i'} \;\bigg|\; \underbrace{\mathbf{I}}_{n\times n} \;\bigg|\; \underbrace{\frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}_{\mathrm{r}}}[\mathbf{x}_i']}_{n\times s} \right) . \tag{4.22}$$

Compared with the initial NPL algorithm (4.20), the new Algorithm II (4.22) with the linear equation system (2.28) estimates the model parameters in terms of form $\mathbf{a}_{\mathrm{g}}$, position $\mathbf{a}_{\mathrm{p}}$, and rotation parameters $\mathbf{a}_{\mathrm{r}}$.

The advantage of Algorithm II is the lowest memory space usage of all three ODF Algorithms I–III for parametric features. A drawback of Algorithm II, both the initial NPL algorithm (4.20) and the new Algorithm II (4.22), is that the convergence and the accuracy of 3-D curve fitting such as fitting a circle in space are relatively poor [10], [77]. As a worse case, if the initial parameters $\mathbf{a}_0$ are not close enough to the final estimation $\hat{\mathbf{a}}$, the convergence performance of 3-D curve fitting with Algorithm II becomes unbearably poor. 2-D curve fitting or surface fitting with Algorithm II does not suffer from such convergence problems.

4.2.3 Algorithm III (FhG)

The new Algorithm III [10] is based on the performance index (2.21) and alternately estimates the parameters $\mathbf{a}$ and $\{\mathbf{u}_i\}_{i=1}^m$ (*variable-separation method,* see (4.2)). The location parameters $\{\mathbf{u}_i'\}_{i=1}^m$ of the minimum distance points $\{\mathbf{X}_i'\}_{i=1}^m$ on the current model feature from each given point $\{\mathbf{X}_i\}_{i=1}^m$ are found by the algorithm described in Sect. 4.1 (inner iteration). In this section, the coordinate-based algorithm (2.32) (outer iteration) which updates the model parameters $\mathbf{a}$ minimizing the performance index (2.21) is generally implemented.

From $\mathbf{x}(\mathbf{a}_{\mathrm{g}}, \mathbf{u})$ (1.2) and $\mathbf{X} = \mathbf{R}^{-1}\mathbf{x} + \mathbf{X}_{\mathrm{o}}$ (1.3), the Jacobian matrices $\{\mathbf{J}_{\mathbf{X}_i',\mathbf{a}}\}_{i=1}^m$ of each *minimum distance point* $\mathbf{X}_i'$ are derived:

$$\begin{aligned}
\mathbf{J}_{\mathbf{X}_i',\mathbf{a}} &= \frac{\partial \mathbf{X}}{\partial \mathbf{a}}\bigg|_{\mathbf{X}=\mathbf{X}_i'} \\
&= \left(\mathbf{R}^{\mathrm{T}}\left(\frac{\partial \mathbf{x}}{\partial \mathbf{a}} + \frac{\partial \mathbf{x}}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{a}}\right) + \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}}\mathbf{x} + \frac{\partial \mathbf{X}_{\mathrm{o}}}{\partial \mathbf{a}}\right)\bigg|_{\mathbf{u}=\mathbf{u}_i'} \\
&= \underbrace{\mathbf{R}^{\mathrm{T}}\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{a}}}_{n\times p}\bigg|_{\mathbf{u}=\mathbf{u}_i'} + \left(\underbrace{\mathbf{R}^{\mathrm{T}}\frac{\partial \mathbf{x}}{\partial \mathbf{a}_{\mathrm{g}}}}_{n\times l}\bigg|_{\mathbf{u}=\mathbf{u}_i'} \;\bigg|\; \underbrace{\mathbf{I}}_{n\times n} \;\bigg|\; \underbrace{\frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}_{\mathrm{r}}}[\mathbf{x}_i']}_{n\times s} \right) .
\end{aligned} \tag{4.23}$$

The derivative matrix $\partial\mathbf{u}/\partial\mathbf{a}$ at $\mathbf{u} = \mathbf{u}_i'$ in (4.23) represents the variational behavior of the location parameters $\mathbf{u}_i'$ of the minimum distance point relative to the differential changes of the model parameters $\mathbf{a}$. Purposefully, $\partial\mathbf{u}/\partial\mathbf{a}$ is derived from the minimum distance condition (4.5) which, provided with the model parameters $\mathbf{a}$ and the given point $\mathbf{X}_i$, constrains the location parameters $\mathbf{u}$ to represent the minimum distance point. Because (4.5) has an implicit form, its derivatives lead to

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{a}} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}_i}\frac{\partial \mathbf{x}_i}{\partial \mathbf{a}} + \frac{\partial \mathbf{f}}{\partial \mathbf{a}} = \mathbf{0} \quad \text{or} \quad \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{a}} = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}_i}\frac{\partial \mathbf{x}_i}{\partial \mathbf{a}} + \frac{\partial \mathbf{f}}{\partial \mathbf{a}}\right) , \quad (4.24)$$

where $\partial \mathbf{x}_i/\partial \mathbf{a}$ is, from $\mathbf{x}_i = \mathbf{R}(\mathbf{X}_i - \mathbf{X}_o)$ (4.3),

$$\begin{aligned}\frac{\partial \mathbf{x}_i}{\partial \mathbf{a}} &= \frac{\partial \mathbf{R}}{\partial \mathbf{a}}[\mathbf{X}_i - \mathbf{X}_o] - \mathbf{R}\frac{\partial \mathbf{X}_o}{\partial \mathbf{a}} \\ &= \left(\mathbf{0} \;\middle|\; -\mathbf{R} \;\middle|\; \frac{\partial \mathbf{R}}{\partial \mathbf{a}_r}[\mathbf{X}_i - \mathbf{X}_o]\right) .\end{aligned}$$

The other three matrices $\partial \mathbf{f}/\partial \mathbf{u}$, $\partial \mathbf{f}/\partial \mathbf{x}_i$, and $\partial \mathbf{f}/\partial \mathbf{a}$ in (4.6) and (4.24) can be directly derived from (4.5). The elements of these three matrices are composed of simple linear combinations of components of the error vector $(\mathbf{x}_i - \mathbf{x})$ with elements of the following three vector/matrices $\partial \mathbf{x}/\partial \mathbf{u}$, $\mathbf{H}$, and $\mathbf{G}$ (XHG matrix):

$$\frac{\partial \mathbf{x}}{\partial \mathbf{u}} = (\mathbf{x}_u \;\; \mathbf{x}_v) , \quad \mathbf{H} = \begin{pmatrix}\mathbf{x}_{uu} & \mathbf{x}_{uv} \\ \mathbf{x}_{vu} & \mathbf{x}_{vv}\end{pmatrix} , \quad \mathbf{G} = \begin{pmatrix}\mathbf{G}_0 \\ \mathbf{G}_1 \\ \mathbf{G}_2\end{pmatrix} \triangleq \frac{\partial}{\partial \mathbf{a}_g}\begin{pmatrix}\mathbf{x} \\ \mathbf{x}_u \\ \mathbf{x}_v\end{pmatrix} , \quad (4.25)$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = (\mathbf{x}_u \;\; \mathbf{x}_v)^{\mathrm{T}}(\mathbf{x}_u \;\; \mathbf{x}_v) - \begin{pmatrix}(\mathbf{x}_i - \mathbf{x})^{\mathrm{T}}\mathbf{x}_{uu} & (\mathbf{x}_i - \mathbf{x})^{\mathrm{T}}\mathbf{x}_{uv} \\ (\mathbf{x}_i - \mathbf{x})^{\mathrm{T}}\mathbf{x}_{vu} & (\mathbf{x}_i - \mathbf{x})^{\mathrm{T}}\mathbf{x}_{vv}\end{pmatrix} ,$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}_i} = -(\mathbf{x}_u \;\; \mathbf{x}_v)^{\mathrm{T}} , \quad \frac{\partial \mathbf{f}}{\partial \mathbf{a}} = \left(\begin{matrix}\mathbf{x}_u^{\mathrm{T}}\mathbf{G}_0 - (\mathbf{x}_i - \mathbf{x})^{\mathrm{T}}\mathbf{G}_1 \\ \mathbf{x}_v^{\mathrm{T}}\mathbf{G}_0 - (\mathbf{x}_i - \mathbf{x})^{\mathrm{T}}\mathbf{G}_2\end{matrix} \;\middle|\; \mathbf{0} \;\middle|\; \mathbf{0}\right) .$$

Now (4.24) can be solved for $\partial \mathbf{u}/\partial \mathbf{a}$ at $\mathbf{u} = \mathbf{u}'_i$, then the Jacobian matrix (4.23) and the linear equation system (2.32) can be completed and solved for the parameter update $\Delta \mathbf{a}$.

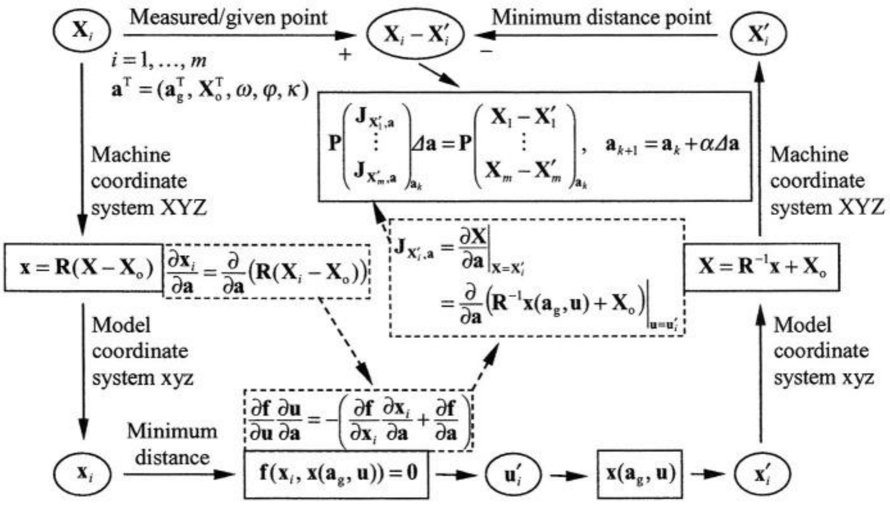

Measured/given point
$\mathbf{X}_i$
$i = 1, \ldots, m$
$\mathbf{a}^{\mathrm{T}} = (\mathbf{a}_g^{\mathrm{T}}, \mathbf{X}_o^{\mathrm{T}}, \omega, \varphi, \kappa)$
$\mathbf{X}_i - \mathbf{X}'_i$
+
−
Minimum distance point
$\mathbf{X}'_i$
$\mathbf{P}\begin{pmatrix}\mathbf{J}_{\mathbf{X}'_1,\mathbf{a}} \\ \vdots \\ \mathbf{J}_{\mathbf{X}'_m,\mathbf{a}}\end{pmatrix}_{\mathbf{a}_k}\Delta\mathbf{a} = \mathbf{P}\begin{pmatrix}\mathbf{X}_1 - \mathbf{X}'_1 \\ \vdots \\ \mathbf{X}_m - \mathbf{X}'_m\end{pmatrix}_{\mathbf{a}_k}, \quad \mathbf{a}_{k+1} = \mathbf{a}_k + \alpha\Delta\mathbf{a}$
Machine coordinate system XYZ
Machine coordinate system XYZ
$\mathbf{x} = \mathbf{R}(\mathbf{X} - \mathbf{X}_o)$
$\frac{\partial \mathbf{x}_i}{\partial \mathbf{a}} = \frac{\partial}{\partial \mathbf{a}}(\mathbf{R}(\mathbf{X}_i - \mathbf{X}_o))$
$\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}} = \left.\frac{\partial \mathbf{X}}{\partial \mathbf{a}}\right|_{\mathbf{X}=\mathbf{X}'_i} = \left.\frac{\partial}{\partial \mathbf{a}}(\mathbf{R}^{-1}\mathbf{x}(\mathbf{a}_g, \mathbf{u}) + \mathbf{X}_o)\right|_{\mathbf{u}=\mathbf{u}'_i}$
$\mathbf{X} = \mathbf{R}^{-1}\mathbf{x} + \mathbf{X}_o$
Model coordinate system xyz
Model coordinate system xyz
$\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{a}} = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}_i}\frac{\partial \mathbf{x}_i}{\partial \mathbf{a}} + \frac{\partial \mathbf{f}}{\partial \mathbf{a}}\right)$
$\mathbf{x}_i$
Minimum distance
$\mathbf{f}(\mathbf{x}_i, \mathbf{x}(\mathbf{a}_g, \mathbf{u})) = \mathbf{0}$
$\mathbf{u}'_i$
$\mathbf{x}(\mathbf{a}_g, \mathbf{u})$
$\mathbf{x}'_i$

Fig. 4.5. Information flow with Algorithm III (FhG) for parametric features

To note is that only the standard feature description $\mathbf{x}(\mathbf{a}_g, \mathbf{u})$ (1.2), without involvement of the position/rotation parameters, is required in (4.25). The overall structure of Algorithm III remains unchanged for all fitting problems of parametric features (Fig. 4.5). All that is required for a new parametric feature is to derive the XHG matrix (4.25) from (1.2) of the new model feature and to supply a set of initial parameter values $\mathbf{a}_0$ for iteration (2.32). Algorithm III shows robust and fast convergence for 2-D/3-D curve and surface fitting. Memory space usage and computing cost associated with Algorithm III are proportional to the number of data points. In comparison with Algorithm I (Sect. 4.2.1) and Algorithm II (Sect. 4.2.2), a disadvantage of Algorithm III is the involvement of the second derivatives $\partial^2\mathbf{x}/\partial\mathbf{u}\partial\mathbf{a}_g$ (see $\mathbf{G}_1$ and $\mathbf{G}_2$ in (4.25)).

4.2.4 Comparison of the Three Algorithms

In the previous sections, the distance-based algorithm (2.28) (Algorithm II) and the coordinate-based algorithm (2.32) (Algorithms I and III) are implemented in a general and well-organized manner for parametric features. Algorithm I (ETH) is known in literature [34], [74]. The initial Algorithm II (NPL) without the parameter grouping of $\mathbf{a}^T = (\mathbf{a}_g^T, \mathbf{a}_p^T, \mathbf{a}_r^T)$ is also known in literature [25], [77], which we modified into a new Algorithm II (FhG) with the parameter grouping. Algorithm III (FhG) is a new algorithm recently developed at the Fraunhofer IPA [10]. All of the three Algorithms I–III with parameter grouping are applicable to any kind of parametric feature. To apply the three algorithms to a new parametric feature, merely the provision of the first and the second derivatives of the standard feature description $\mathbf{x}(\mathbf{a}_g, \mathbf{u})$ (1.2) which contains only a few form parameters $\mathbf{a}_g$ are required. The three Algorithms I–III are compared as follows:

- Computing cost
  - If not supported by a sparse matrix algorithm, Algorithm I has the computing cost of $O(m^3)$. Supported by a sparse matrix algorithm, it is of $O(m)$ [74]. But, to note is, a sparse matrix algorithm does not work at a null cost of implementation and computing
  - Algorithms II and III have about the same computing cost of $O(m)$. The time-consuming and shared procedure of them is finding of the location parameters of the minimum distance points.
- Memory space usage
  - If not supported by a sparse matrix algorithm, Algorithm I has a memory space usage of $O(m^2)$. Supported by a sparse matrix algorithm, it is of $O(m)$ [74]
  - Memory space usage of the Algorithms II and III is of $O(m)$. Algorithm II demands one half (2-D feature fitting) or one third (3-D feature fitting) of the memory space usage of Algorithm III.
- Implementation cost
  - Algorithm I requires only the first derivatives of $\mathbf{x}(\mathbf{a}_g, \mathbf{u})$. If the minimum distance point algorithm (Sect. 4.1) is to be used for supplying the initial values and/or for refreshing the location parameters, the second derivatives $\partial^2\mathbf{x}/\partial\mathbf{u}^2$ should also be provided

  - The Algorithms II and III require the first derivatives of $\mathbf{x}(\mathbf{a}_g, \mathbf{u})$ and the second derivatives $\partial^2\mathbf{x}/\partial\mathbf{u}^2$
  - The second derivatives $\partial^2\mathbf{x}/\partial\mathbf{u}\partial\mathbf{a}_g$ are additionally required by Algorithm III.
- Convergence performance
  - Algorithm I shows slow convergence if the initial model parameters $\mathbf{a}_0$ are not close enough to the final estimation $\hat{\mathbf{a}}$
  - Algorithm II has poor convergence performance for 3-D curve fitting
  - Algorithm III shows fast and robust convergence for 2-D/3-D curve fitting and surface fitting.

On reviewing the above comparison, it is seen that each of the three Algorithms I–III has at least one relatively weak point for applications, which the other two algorithms do not have. The weak points are

- Algorithm I: high computing cost and slow convergence
- Algorithm II: poor convergence with 3-D curve fitting
- Algorithm III: involvement of the second derivatives $\partial^2\mathbf{x}/\partial\mathbf{u}\partial\mathbf{a}_g$.

Otherwise, the three algorithms are all highly applicable to the ODF problems of general parametric curves and surfaces. Thus, the three algorithms are to be purposefully applied, taking into consideration the number of data points, the type of model feature, the goodness of initial values, and the frequency of task run.

4.3 Fitting Examples

In order to compare experimentally the three Algorithms I–III, this section gives examples of 3-D curve and surface fitting.

4.3.1 Helix Fitting

As an example of 3-D curve fitting, the ODF of a helix is shown. The standard feature description (1.2) of a helix can be defined in xyz frame as follows:

$$\mathbf{x}(r, h, u) \triangleq (r\cos u,\ r\sin u,\ hu/2\pi)^{\mathrm{T}} \qquad \text{with} \qquad -\infty \leq u \leq \infty\,,$$

where r and h are the radius and the pitch of a helix, respectively. Note that there are an infinite number of sets of the extrinsic parameter values $\{\mathbf{X}_o, \kappa\}$ representing the same spatial instance of a helix (please imagine corkscrewing a helix), thus showing the same performance value of (2.20)–(2.21) with a set of given points. In order to remove this freedom in parameter space, $\mathbf{X}_o$ is constrained to be the nearest point on the helix axis $\mathbf{r}_z$ (1.3) from the mass center $\bar{\mathbf{X}}$ of the set of given points (see (2.33) and Sect. B.1.2)

$$f_c(\mathbf{a}_p, \mathbf{a}_r) \triangleq (\mathbf{X}_o - \bar{\mathbf{X}})^{\mathrm{T}}\mathbf{r}_z(\omega, \varphi) = 0\,.$$

The initial parameter values are obtained from a 3-D circle fitting and a cylinder fitting successively (Fig. 1.3a). The helix fitting to the set of points in Table 4.3, using Algorithm III with the initial values of$h = 5$ and $\kappa = \pi/2$, terminated after seven iteration cycles 0.07 s for $\|\Delta\mathbf{a}\| = 9.2 \cdot 10^{-7}$ with a Pentium 133 MHz PC (Fig. 4.6, Table 4.4). They were ten iteration cycles 0.16s for $\|\Delta\mathbf{a}\| = 4.7 \cdot 10^{-7}$ using Algorithm I and, 128 iteration cycles 2.21 s for $\|\Delta\mathbf{a}\| = 6.2 \cdot 10^{-6}$ using Algorithm II. Algorithm II showed relatively slow convergence for the 3-D circle and the helix fitting (3-D curve fitting).

Table 4.3. Ten coordinate triples ($n = 3$) representing a helix

X	7	5	3	1	−1	−3	−4	−5	−5	−5
Y	1	3	4	4	4	4	2	1	−1	−3
Z	3	4	4	4	3	2	1	0	−1	−1

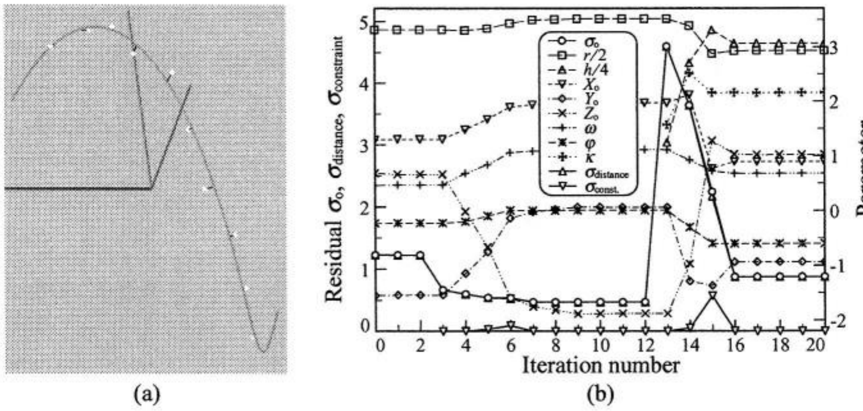


Fig. 4.6. Orthogonal distance fitting to the set of points in Table 4.3: (a) Helix fit; (b) Convergence of the fit (Algorithm III). Iteration number 0–2: 3-D circle, 3–12: circular cylinder, and 13–: helix fit with the initial values of $h = 5$ and $\kappa = \pi/2$

Table 4.4. Results of orthogonal distance fitting (Algorithm III) to the points in Table 4.3

Parameter $\hat{\mathbf{a}}$	σ_0	r	h	X_o	Y_o
Circle	1.2264	6.6484	—	1.3055	−1.5365
$\sigma(\hat{\mathbf{a}})$	—	0.4262	—	0.3662	0.5513
Cylinder	0.4696	7.0495	—	1.9752	0.0669
$\sigma(\hat{\mathbf{a}})$	—	0.4841	—	0.3381	0.7822
Helix	0.8736	5.8695	12.2904	0.8919	−0.9342
$\sigma(\hat{\mathbf{a}})$	—	0.7900	5.2913	0.5218	0.7419

Parameter $\hat{\mathbf{a}}$	Z_o	ω	φ	κ
Circle	0.6629	0.4707	−0.2235	—
$\sigma(\hat{\mathbf{a}})$	0.4289	0.0854	0.0544	—
Cylinder	−1.8749	1.1132	0.0086	—
$\sigma(\hat{\mathbf{a}})$	1.1540	0.2390	0.0963	—
Helix	1.0216	0.6774	−0.6012	2.1575
$\sigma(\hat{\mathbf{a}})$	0.5099	0.1516	0.1986	0.2848

4.3.2 Ellipsoid Fitting

To compare the convergence and computing cost of the three Algorithms I–III for surface fitting, ellipsoids are fitted to the sets of 10–100 points (Fig. 4.7). The standard feature description (1.2) of an ellipsoid can be defined in xyz frame as follows:

$$\mathbf{x}(a,b,c,\mathbf{u}) \triangleq \begin{pmatrix} a\cos u\cos v \\ b\sin u\cos v \\ c\sin v \end{pmatrix} \quad \text{with} \quad -\pi < u \le \pi\,, \quad -\pi/2 \le v \le \pi/2\,.$$

There is no significant difference in convergence performance between Algorithms II and III (Fig. 4.8a). Algorithm I shows poor convergence if the initial pa-

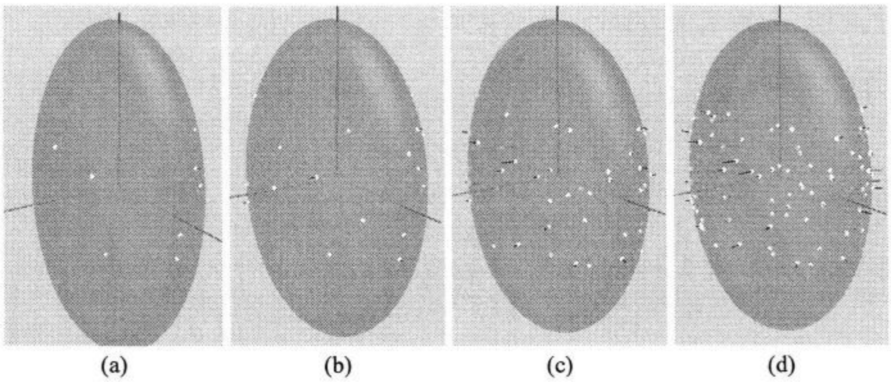


Fig. 4.7. Ellipsoid fit to 10–100 points randomly distributed on the parametric domain uv of $-1.0 \le u \le 3.0$ and $-0.5 \le v \le 0.5$ with an rms error distance of 0.1 for $\mathbf{a}_g = (10, 15, 20)^T$, $\mathbf{a}_p = (10, -10, 10)^T$ and $\mathbf{a}_r = (0.5, -0.5, 0.5)^T$. Distance error bars are elongated ten times: Ellipsoid fit to the (a) 10, (b) 20, (c) 50, and (d) 100 points

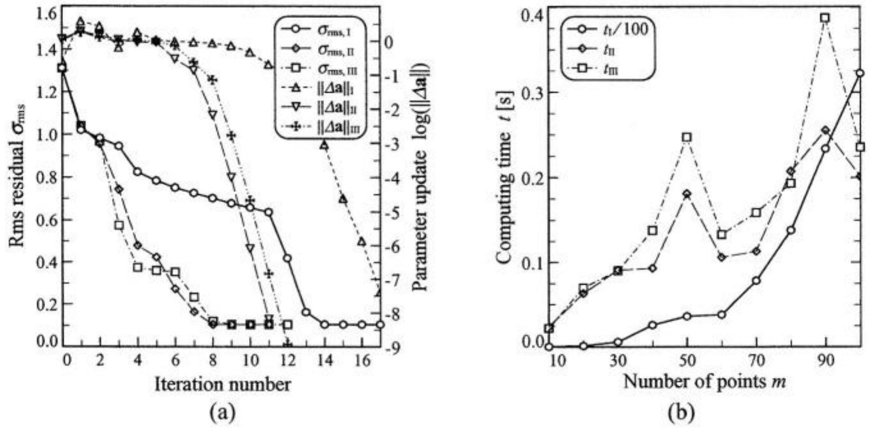


Fig. 4.8. Convergence and computing cost of the three Algorithms I–III. The initial parameter values are taken from the sphere fitting to each set of points: (a) Convergence of the ellipsoid fit to the 100 points in Fig. 4.7d; (b) Computing time with a Pentium 866 MHz PC

rameter values are not close enough to the final estimation values. Furthermore, the computing cost associated with Algorithm I increases rapidly with the number of data points. For $m = 100$, the computing cost of Algorithm I is about 140 times higher than that of Algorithm II or III. For $m = 10$, it is 1.7 times high (see Fig. 4.8b). Algorithm III demands a slightly higher computing cost than Algorithm II because the number of rows in the linear equation system (2.32) ($3m$, Algorithm III with surface fitting) is three times great relative to that of the linear equation system (2.28) (m, Algorithm II). Not forgetting that the time-consuming procedure, compared with solving the linear equation systems (2.28) and (2.32), is the determination of the location parameters $\{\mathbf{u}'_i\}_{i=1}^{m}$ of the minimum distance points, a factor which is shared by Algorithms II and III. The rise in the computing time of Algorithms II and III with sets of 50 and 90 points is due to the refreshment of the location parameters of the minimum distance points after a rearrangement of the parameter values of $\mathbf{a}_\mathrm{g}$ and $\mathbf{a}_\mathrm{r}$. For the purpose of a directly comparable presentation of the resultant ellipsoids in Fig. 4.7, parameter rearrangement is optionally activated to hold $a < b < c$ (see Sect. B.1.4).

5. Object Reconstruction from Unordered Point Cloud

The goal of *object reconstruction* is to generate the list of objects in a real environment, together with the object model parameters and the geometric interrelations between the objects. A successful object reconstruction presupposes the general information about the objects (such as object data base) and a large amount of raw data directly obtained from the objects (such as a point cloud, object surface color and texture). Because the algorithmic method for this object reconstruction utilizing all these information and raw data will not be available in the near future, this chapter concentrates only on semi-automatic object reconstruction from a point cloud. Of course, in comparison with the methods that are currently available, the object reconstruction presented in this chapter possesses a number of advanced algorithmic features, e.g. accuracy, flexibility, and degree of automation.

The accurate estimation of object model parameters from a point cloud is a substantial element of object reconstruction. The ODF algorithms described in the previous chapters play a fundamental role in object reconstruction because they estimate the model parameters in terms of form, position and rotation parameters by minimizing the square sum of the geometric error distances. Describing the ODF algorithms in the previous chapters, the data points are supposed to have been obtained from a single object. However, with many applications, the number and the type of object model features contained in a point cloud are generally unknown. Thus, *object recognition* - including object detection and identification - is an essential element of object reconstruction. Whilst fully automatic object recognition in a point cloud may only be possible by analyzing all the available information mentioned above, a semi-automatic procedure is presented in this chapter for object recognition in a point cloud, which requires minimal human operator assistance [11].

Given the initial type of model feature, the procedure is triggered by picking one or two seed points in the point cloud. Then, the potential membership points of the model feature are collected and outliers are eliminated. Finally, the model feature is fitted to the cleaned set of points (inliers). If necessary, the model type is updated and the procedure is repeated from the point-collecting step until the given break condition is satisfied. Two examples of object recognition in a point cloud taken from real objects are given in this chapter.

5.1 Applications of Object Reconstruction

Object reconstruction has been eagerly pursued in computer/machine vision with the aim of providing a computing machine with the ability (comparable with that of human intelligence) of seeing and understanding the real geometric environment. Although the goal is still far away, current results from research efforts to reach this objective can be used for a variety of applications in and out of computer/machine vision. Whilst computer vision is more interested in the semantic interpretation of a real environment, machine vision is application-oriented towards the generation of the necessary information for controlling processes and systems in an industrial environment.

Object reconstruction also finds applications in coordinate metrology, reverse engineering, and CAD/CAM where engineering workpieces need to be measured then represented by their model parameters. Compared with computer/machine vision, these application fields demand a relatively highly accurate determination of object model parameters. Once the workpieces have been represented by their model parameters, they can be modified and combined together in a virtual environment in the computer. The reconstructed computational model is used for designing and machining of new workpieces.

As efficient and high-throughput optical 3-D measuring devices are available, the digital documentation of the engineering environment, e.g. a whole factory or engineering plant, is emerging as a new application of object reconstruction. Current optical 3-D measuring devices such as laser radar are capable of measuring millions of dense 3-D points in a few seconds. The engineering environment reconstructed from the point cloud obtained is used for various applications, e.g. digital factory, factory planning and facility maintenance. Considering the ever-increasing number of objects and data points to be processed, the computational efficiency, flexibility and user interface of the data processing software are of major importance to the digital documentation of an engineering environment.

The digital documentation of technical drawings would be an interesting application of object reconstruction. A technical drawing contains numbers of lines and curves which are generally supposed to have been constrained by geometric interrelations such as continuity, relative angle, position and size. The ODF with geometric constraints (Sect. 2.3.3) is a powerful tool for digitally documenting of technical drawings.

Object reconstruction also plays an important role in medical imaging, an emerging application field of image processing and pattern recognition. Information extraction and interpretation, including the procedures of image segmentation and model fitting in 2-D/3-D medical images taken by computer tomography or magnetic resonance technology, are essential elements in the visualization and diagnostics of human body and organs.

5.2 Semi-automatic Object Recognition

For object recognition in a point cloud, an effective search (segmentation) method first has to be found for the measurement points potentially belonging to the target model feature. Also, outliers need to be detected and excluded from model fitting because they distort results. In reality, segmentation and outlier elimination are the most tedious and time-consuming procedures of object reconstruction. Once the point cloud has been segmented and outliers eliminated, the model fitting follows. By analyzing the results of model fitting as described in Sect. 1.1.3 and Sect. 2.3.4, the reliability of the estimated model parameters can be tested.

The ODF algorithms described in the previous chapters possess highly advantageous algorithmic features for segmentation, outlier detection and model fitting in a point cloud. Such features include geometric error definition and parameter grouping in terms of form, position and rotation. The geometric (shortest) distance between the model feature and a measurement point is indisputably the best error measure for determining whether the point is an inlier point of model feature, because a measurement point is the probable observation of the *nearest* object point from the measurement point. And, as will be shown in the following sections, parameter grouping makes possible an efficient segmentation of a point cloud.

We propose a semi-automatic object recognition procedure 'Click & Clear' (Fig. 5.1 [11]). This is an interactive procedure between segmentation, outlier elimination and model fitting, in which the ODF algorithms and the properties of implicit features play an important role. The target model feature can be refined according to the model selection criteria discussed in Sect. 1.1.3. Once an object has been successfully recognized, its membership points are eliminated from the point cloud. The procedure is repeated until no more object structure is visible in the point cloud. The proposed procedure is minimally assisted by a human operator and considerably enhances the current practice of object reconstruction in real applications.

For an accurate model fitting, the ODF algorithms for implicit features described in Chap. 3 are applied. Also, to achieve a high automation degree of the overall procedures of the segmentation and outlier elimination in point cloud, the properties of implicit features are utilized. The properties of the applied fitting algorithms and of implicit features [11] are summarized as follows:

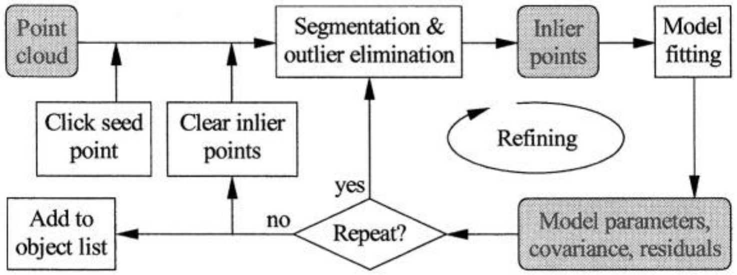


Fig. 5.1. Object recognition scheme 'Click & Clear' in a point cloud

- The ODF algorithms possess the following algorithmic features:
  - Geometric (Euclidean) definition of the error distance
  - Parameter grouping in terms of form, position, and rotation
  - Low implementation cost for a new model feature
  - The computing time and memory space usage are proportional to the number of data points
  - Automatic determination of the initial values for model parameters
  - Provision of the parameter covariance matrix
  - Allowance of parameter constraints, e.g. regarding the size, shape, position and orientation of model features.
- The mathematical model description in implicit form $F(\mathbf{a}, \mathbf{X}) = 0$ has the following useful properties:
  - Simple on-off and inside-outside decision
  - Links up multiple object patches with/without overlap
  - Suitable for representing unordered sets of points with an uneven point density
  - The iso-feature $F(\mathbf{a}, \mathbf{X}) - \text{const} = 0$ represents an approximate equidistance (offset) model feature to the implicit feature $F(\mathbf{a}, \mathbf{X}) = 0$.

5.2.1 Segmentation, Outlier Elimination, and Model Fitting

Utilizing the algorithmic features listed in the last subsection, the following interactive procedure between segmentation, outlier elimination and model fitting in point cloud (Fig. 5.1) was used. The procedure starts by picking one or two seed points in a point cloud and choosing a low-level model type [9], [11]. If necessary, the model type can be updated, e.g. from a sphere to an ellipsoid and vice versa during the model refining cycles in Fig. 5.1.

1. From the point list, clear the measurement points that belong to the known objects such as the measuring stage or already-detected and identified objects.
2. Pick the seed point and select a low-level model type, e.g. a sphere.
3. Initialize the model parameters through a model fitting to a small patch of the point cloud around the seed point. Also initialize the safety margin t of the point searching domain volume to be used in the next step, e.g. $t_0 = r/2$ for a sphere.
4. For the current model feature, determine the shortest error distances d_i of each measurement point lying inside the domain volume (see the next section for a detailed description) and evaluate the rms error distance $\sigma_{\text{error}} = \sqrt{\frac{1}{m}\sum_{i=1}^{m} d_i^2}$, where m is the number of measurement points lying inside the domain volume.
5. Determine the inlier set of measurement points lying inside the domain volume and having error distances of $d_i \leq \alpha_{\text{threshold}} \cdot \sigma_{\text{error}}$, where $\alpha_{\text{threshold}}$ is the outlier threshold factor ($2.0 \leq \alpha_{\text{threshold}} \leq 3.0$ works well). Optionally, if there is no change in the inlier set, terminate the procedure.
6. Fit the model feature to the inlier set and save the resultant rms error distance $\sigma_{\text{fit\_error}}$. If $\sigma_{\text{fit\_error}}$ is smaller than the given error size concerning the accuracy of the measuring device used, terminate the procedure.

7. Update the safety margin $t \leftarrow \alpha_{\text{growing}} \cdot \alpha_{\text{threshold}} \cdot \sigma_{\text{fit\_error}}$ of the domain volume, where α_{growing} is the domain growing factor ($1.0 \leq \alpha_{\text{growing}} \leq 1.2$ works well).
8. If necessary, according to the model selection criteria described in Sect. 1.1.3, change the model type, e.g. from a sphere to an ellipsoid and vice versa.
9. Repeat from the fourth step onwards.

5.2.2 Domain Volume for Measurement Points

In order not only to exclude the outliers from model fitting but also to avoid unnecessary determination of the minimum distance points in the fourth step of the procedure described in the last subsection, a two-step domain volume criterion was applied (Fig. 5.2 [11]). Note that the determination of the minimum distance points is the time-consuming element in the overall procedure. Also, outlier elimination is carried out by analyzing the statistical distribution of the minimum distances. The two-step criterion is described in detail as follows.

First, the domain of the measurement points is restricted within a box with a safety margin t and containing the interesting portion of the standard model feature $f(\mathbf{a}_g, \mathbf{x}) = 0$ (1.1) defined in xyz frame. For example, the domain box of an ellipsoid with axis-lengths a, b, c can be defined as follows:

$$|x| \leq a + t\,, \qquad |y| \leq b + t\,, \qquad \text{and} \qquad |z| \leq c + t\,.$$

With the domain box criterion, the potential measurement points of the target object can be screened from a large set of measurement points at a minimal computing cost (for each measurement point, only a few multiplications are required for coordinate transformation $\mathbf{x} = \mathbf{R}(\mathbf{X} - \mathbf{X}_o)$ (1.3)).

Next, from the measurement points lying inside the domain box, the minimum distance point is finally determined, if the measurement point also lies between two iso-features $f(\mathbf{a}_g, \mathbf{x}) - \text{const} = 0$ with $\text{const}_{\text{inner}} < 0$ and $\text{const}_{\text{outer}} > 0$, respectively. In particular, for an ellipsoid, the following are used:

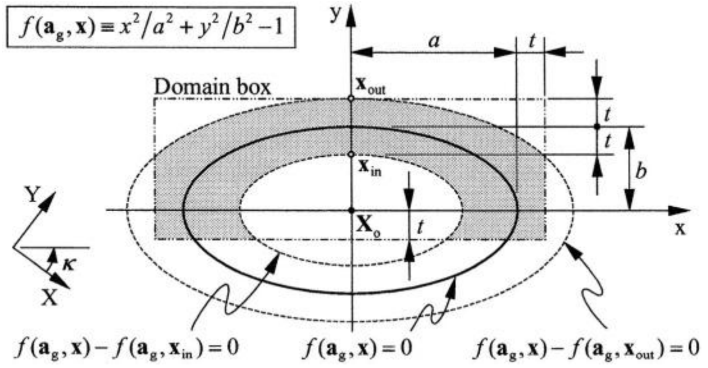

$f(\mathbf{a}_g, \mathbf{x}) \equiv x^2/a^2 + y^2/b^2 - 1$
Domain box
y
a
t
$\mathbf{X}_{\text{out}}$
t
t
b
$\mathbf{X}_{\text{in}}$
$\mathbf{X}_o$
t
x
Y
κ
X
$f(\mathbf{a}_g, \mathbf{x}) - f(\mathbf{a}_g, \mathbf{x}_{\text{in}}) = 0$
$f(\mathbf{a}_g, \mathbf{x}) = 0$
$f(\mathbf{a}_g, \mathbf{x}) - f(\mathbf{a}_g, \mathbf{x}_{\text{out}}) = 0$

Fig. 5.2. Example of the domain area (shaded with half tone) for an ellipse with $a \geq b$

$$\mathrm{const}_{\mathrm{inner}} = f(\mathbf{a}_g, \mathbf{x}_{\mathrm{inner}}) \qquad \text{with}$$

$$\mathbf{x}_{\mathrm{inner}} = \begin{cases} (\max(a/2,\ a-t),\ 0,\ 0)^{\mathrm{T}} & : a = \min(a,b,c) \\ (0,\ \max(b/2,\ b-t),\ 0)^{\mathrm{T}} & : b = \min(a,b,c) \\ (0,\ 0,\ \max(c/2,\ c-t))^{\mathrm{T}} & : c = \min(a,b,c)\ , \end{cases}$$

and

$$\mathrm{const}_{\mathrm{outer}} = f(\mathbf{a}_g, \mathbf{x}_{\mathrm{outer}}) \qquad \text{with}$$

$$\mathbf{x}_{\mathrm{outer}} = \begin{cases} (a+t,\ 0,\ 0)^{\mathrm{T}} & : a = \min(a,b,c) \\ (0,\ b+t,\ 0)^{\mathrm{T}} & : b = \min(a,b,c) \\ (0,\ 0,\ c+t)^{\mathrm{T}} & : c = \min(a,b,c)\ . \end{cases}$$

The above two-step criterion which defines the domain volume at a low computing cost by utilizing parameter grouping in terms of form, position and rotation, makes possible a simple and inexpensive search for the measurement point candidates which potentially belong to the model feature. The time-consuming determination of the minimum distance points is carried out only for the measurement point candidates lying inside the domain volume.

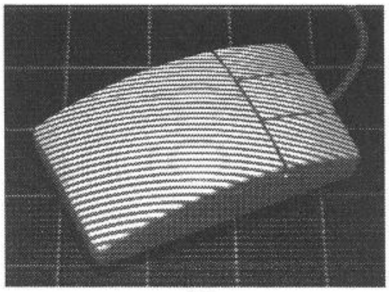
5.3 Experimental Results with Real 3-D Measurement Points

5.3.1 3-D Point Cloud from Stripe Projection Method

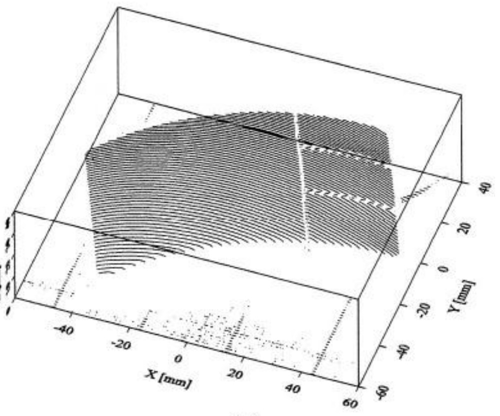
In order to show how the ODF algorithms and the procedure 'Click & Clear' (Fig. 5.1) perform using real data, a sphere and an ellipsoid are fitted to the same set of measurement points of a sphere-like object surface (Fig. 5.3a). By comparing the results from the sphere and the ellipsoid fitting, the correct model feature for the measured object surface can be selected (model selection).

The set of points is acquired using our stripe projecting 3-D measurement system using the Gray code [41] with the Coded Light Approach (CLA) [13], [80]. The measuring system consists of a standard CCD camera and an LCD stripe projector of ABW [54] and is calibrated by photogrammetric bundle adjustment [57]. In order to automate the calibration procedure as far as possible, our patented circular coded targets are used [4], [5], [7]. Our 3-D measurement software determines the coordinates of the object surface points along the stripe edges (Fig. 5.3b). The number of the measured surface points is about 29 400; every eighth point (subtotal 3675 points) is used for the sphere and the ellipsoid fitting in this section (Fig. 5.4). In the first step of the procedure described in Sect. 5.2.1,256 of 3675 measurement points were detected and removed from the point list as they belonged to the measuring stage ($Z \approx 0$).

With the ellipsoid fitting, the parameter constraints of $\omega = 0$ and $\varphi = 0$ with a weighting value of $w_c = 1.0 \cdot 10^8$ (see Sect. 2.3.3) were applied as the object surface seemed to be the top face of an ellipsoid lying *horizontally*. The computing

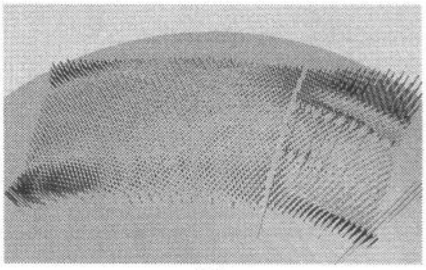


(a)

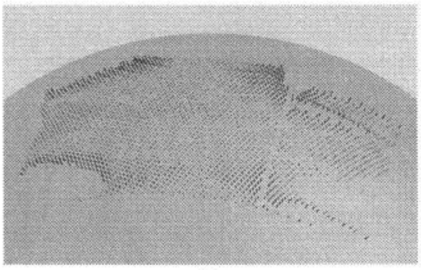


(b)

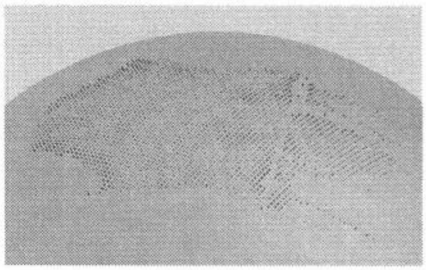
Fig. 5.3. Sphere-like object surface: (a) Stripe projection; (b) Measured point cloud



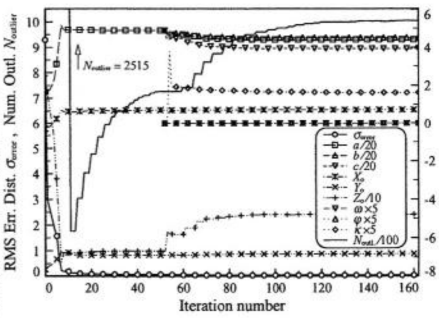
(a)



(b)



(c)



(d)

Fig. 5.4. Results of model fitting to the point cloud (3419 = 3675 – 256 points) from Fig. 5.3b. Distance error bars are elongated ten times: (a) Sphere fit without and; (b) with outlier elimination; (c) Ellipsoid fit with outlier elimination, under the constraints of $\omega = 0$ and $\varphi = 0$; (d) Convergence of the fit with outlier elimination. Iteration number 0–51: 17 rounds of sphere fitting and, 52–162: 26 rounds of ellipsoid fitting. The initial sphere parameters are taken from the mass center and the rms central distance of the data points. After the first round of the sphere fitting, the initial number of the outliers (including the out-of-domain points) $N_{\text{outlier}} = 2515$ is reduced to 177 and, thereafter, increased

Table 5.1. Results of the coordinate-based orthogonal distance fitting to the point cloud of Fig. 5.4b (sphere fitting) and Fig. 5.4c (ellipsoid fitting)

Parameter $\hat{\mathbf{a}}$	σ_{error} [mm]	a [mm]	b [mm]	c [mm]	X_o [mm]
Sphere	0.0596	97.7622	—	—	0.7201
$\sigma(\hat{\mathbf{a}})$	—	0.0306	—	—	0.0051
Ellipsoid	0.0309	88.2968	90.1912	78.8084	0.7280
$\sigma(\hat{\mathbf{a}})$	—	0.2775	0.2936	0.5385	0.0030
Parameter $\hat{\mathbf{a}}$	**Y_o [mm]**	**Z_o [mm]**	**ω [rad]**	**φ [rad]**	**κ [rad]**
Sphere	−6.8813	−66.7420	—	—	—
$\sigma(\hat{\mathbf{a}})$	0.0069	0.0321	—	—	—
Ellipsoid	−6.8170	−47.8462	0.0000	−0.0000	0.3254
$\sigma(\hat{\mathbf{a}})$	0.0042	0.5377	0.0000	0.0000	0.0050

Table 5.2. Correlation coefficients of the sphere parameters in Table 5.1

Cor($\hat{\mathbf{a}}$)	a	X_o	Y_o	Z_o
a	1.00			
X_o	0.15	1.00		
Y_o	−0.03	−0.35	1.00	
Z_o	−1.00	−0.14	0.03	1.00

Table 5.3. Correlation coefficients of the ellipsoid parameters in Table 5.1

Cor($\hat{\mathbf{a}}$)	a	b	c	X_o	Y_o	Z_o	ω	φ	κ
a	1.00								
b	1.00	1.00							
c	1.00	1.00	1.00						
X_o	−0.06	−0.07	−0.07	1.00					
Y_o	−0.05	−0.04	−0.05	−0.39	1.00				
Z_o	−1.00	−1.00	−1.00	0.07	0.05	1.00			
ω	−0.00	−0.00	−0.00	0.00	−0.00	0.00	1.00		
φ	0.00	0.00	0.00	0.00	−0.00	−0.00	−0.00	1.00	
κ	0.12	0.12	0.11	0.09	−0.09	−0.11	−0.00	0.00	1.00

time with a Pentium III 866 MHz PC was a total of 7.8 s for 17 rounds of sphere fitting with a final parameter update size of $\|\Delta\mathbf{a}\| = 3.7 \cdot 10^{-7}$ and a total of 94.1 s for 26 rounds of ellipsoid fitting with $\|\Delta\mathbf{a}\| = 6.5 \cdot 10^{-7}$. The number of outliers (including the out-of-domain points) was 727 for $\sigma_{\text{error}} = 0.0596\,\text{mm}$ and 1005 for $\sigma_{\text{error}} = 0.0309\,\text{mm}$ with the sphere and the ellipsoid fitting, respectively (Fig. 5.4d).

If Fig. 5.4b and Fig. 5.4c are compared, an ellipsoid appears to be a better model feature than a sphere for the measured object surface. However, a look at Table 5.1 and Table 5.3 reveals the relatively large variances and perfect correlations (i.e. +1.0) of the axis-lengths of the estimated ellipsoid which indicate an overparameterization. From this fact, a sphere could be considered rather as the correct model feature for the measured object surface. Note the perfect anti-correlation (i.e. –1.0) between the position parameter Z_0 and the sphere radius a (also the axis-length c of the ellipsoid, Table 5.2 and Table 5.3) which is caused by the one-sided distribution of the data points on the top face of the sphere/ellipsoid. On the other hand, there is no correlation between the first two rotation parameters (ω and φ) and the other parameters of the ellipsoid, which is conditioned by the parameter constraints of $\omega = 0$ and $\varphi = 0$.

5.3.2 3-D Point Cloud from Laser Radar

To show another application example of the procedure 'Click & Clear', a point cloud of about 18 300 points with multiple objects (Fig. 5.5) taken by laser radar [43] is segmented and recognized. Strategically, in order to reduce the overall computing and user working cost, the recognition of simple model features, e.g. a plane and sphere, is to precede the recognition of complex model features. Especially plane recognition is computationally inexpensive (plane fitting is a non-iterative procedure) and generally removes a large portion of the point cloud in the early phase of the overall procedure (Fig. 5.6a and Fig. 5.6b). Consequently, the overall computing cost of the succeeding procedures for the recognition of complex objects can be reduced. The initial parameter values, which are provided from a model fitting to a small but well-conditioned patch of the point cloud around the seed point, generally guarantee stable and robust object recognition. Even when the initial parameter values are falsified (Fig. 5.6a and Fig. 5.6c), the proposed procedure described in Sect. 5.2.1 leads object recognition to the correct result (Fig. 5.6b and Fig. 5.6d).

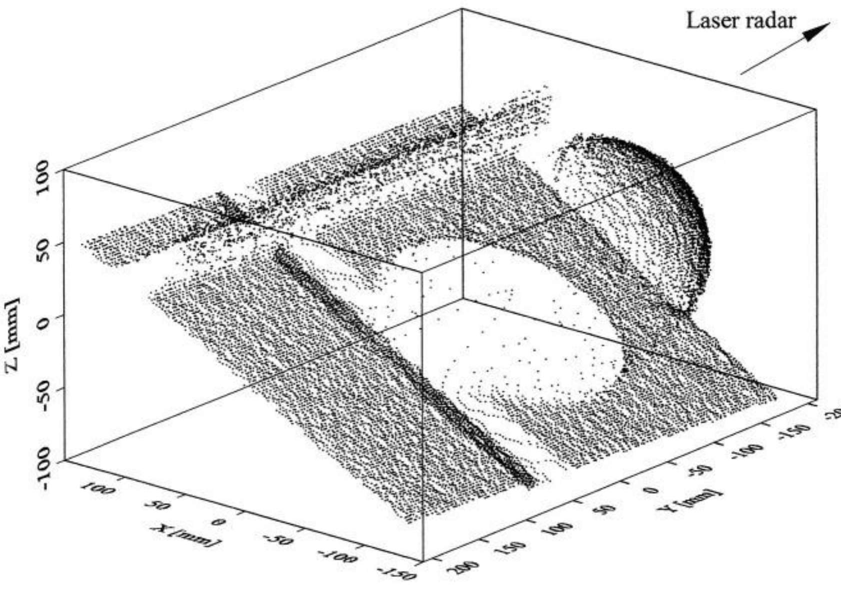


Fig. 5.5. Point cloud (about 18 300 points) taken by laser radar [43]

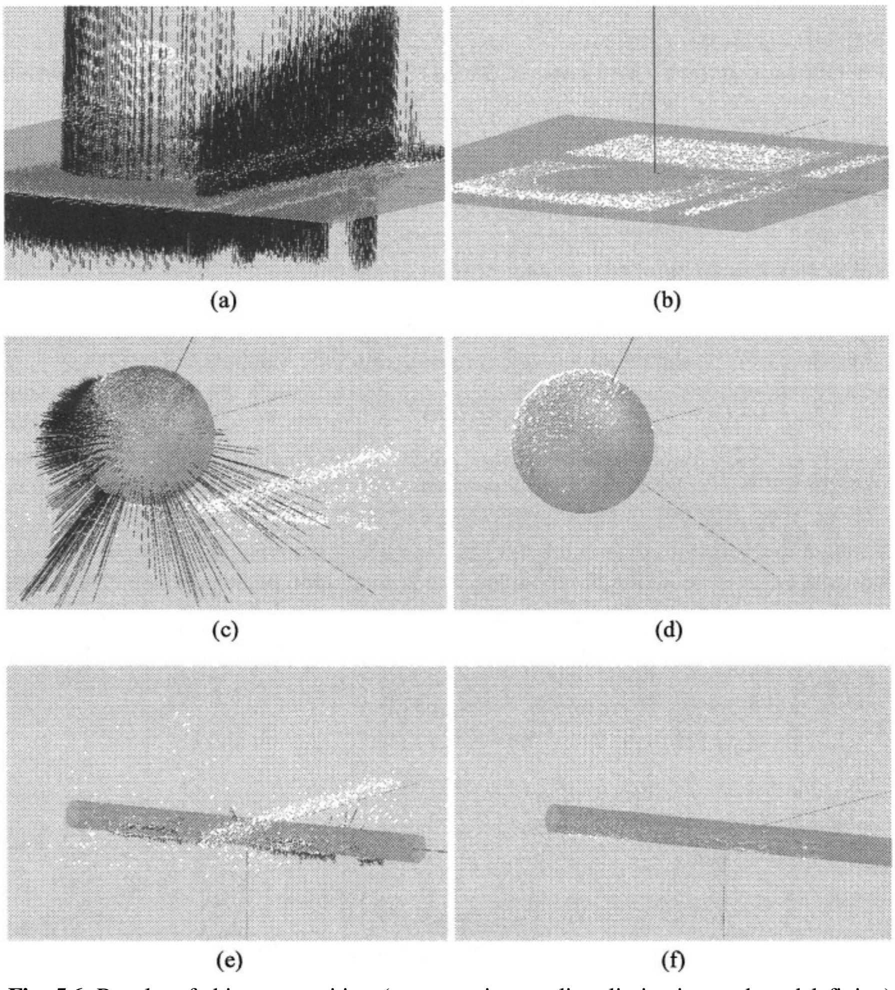


Fig. 5.6. Results of object recognition (segmentation, outlier elimination and model fitting) in the point cloud shown in Fig. 5.5. Distance error bars are elongated four times: (a), (c), and (e) Initial states of the single model fittings; (b), (d), and (f) Segmented point clusters and the fitting results; (a) and (b) Plane recognition; (c) and (d) Sphere recognition; (e) and (f) Cylinder recognition. The membership points are cleared from the point cloud after each successful model recognition

6. Conclusions

6.1 Summary

Orthogonal distance fitting (ODF) estimates the parameters of a model feature by minimizing the square sum of the shortest distances *(geometric distances)* between the model feature and the given points. Due to the applied error measure, the ODF is a substantial mathematical tool for many scientific and engineering disciplines in which coordinate data and dimensional models are processed. The ODF algorithms are of vital importance to coordinate metrology which aims to accurately estimate the model parameters of measurement objects from their measurement points. The recently ratified ISO 10360-6 for testing the data processing softwares for coordinate metrology prescribes the geometric distance as the error measure to be minimized. In addition, with the segmentation and outlier elimination in a point cloud, the geometric distance is the best error measure for discriminating between the inlier and the outlier points of a model feature. Moreover, demands made on 3-D measurement techniques and the data volumes to be processed are rapidly increasing with the progress of science and engineering in the age of computer technology. Advanced optical 3-D measuring devices such as laser radar can generate millions of dense 3-D points within a few seconds. For this reason, the development of efficient and flexible ODF algorithms for processing of coordinate data has been eagerly pursued in the last decade. The ODF problem is a *nonlinear minimization problem* and has been widely recognized as being analytically and computationally difficult.

In this work, the ODF problem has been tackled in a well-structured and easily understandable manner. First, the square sum of the geometric error distances was concisely formulated into two cost functions (2.20) and (2.21). Then, two appropriate numerical methods were proposed, the *distance-based algorithm* (2.28) and the *coordinate-based algorithm* (2.32) for minimizing the two cost functions (2.20) and (2.21), respectively. There are also two strategies for minimizing the cost functions by iteration, the *total method* and the *variable-separation method.* From these two numerical methods and two minimization strategies, the following three realistic algorithmic approaches for solving the nonlinear ODF problem by iteration were identified:

- Algorithm I: combination of the coordinate-based algorithm with the total method
- Algorithm II: combination of the distance-based algorithm with the variable-separation method

- Algorithm III: combination of the coordinate-based algorithm with the variable-separation method.

As the general implementations of the Algorithms I–III can be applied to a wide range of model feature types, five ODF algorithms were described in detail for *implicit* (Chap. 3) and *parametric* (Chap. 4) curves/surfaces in 2-D/3-D space. For each of the five ODF algorithms, the model parameters **a** are grouped and simultaneously estimated in terms of *form* $\mathbf{a}_g$,*position* $\mathbf{a}_p$, and *rotation* parameters $\mathbf{a}_r$, thus providing applications with highly advantageous algorithmic features. From the resulting parameter covariance matrix, useful information can be extracted for testing the reliability of the estimated model parameters. With the exception of Algorithm I for parametric features, the other four algorithms, the Algorithms II and III for implicit and parametric features, have been newly developed in the scope of this work.

Note that the linear equation systems (2.28) and (2.32) for Algorithms II and III, respectively, are constructed without yet assuming the type of the model feature to be fitted. Thus, given the set of points $\{\mathbf{X}_i\}_{i=1}^m$ and initial parameter values $\mathbf{a}_0$, the numerical content of (2.28) and (2.32) is the same between the implicit and the parametric forms of a model feature if the two model forms are characterized by the same set of parameters **a** (e.g., a sphere can be described in both implicit and parametric forms with $\mathbf{a} = (r, X_o, Y_o, Z_o)^\mathrm{T}$). Consequently, except for the numerical inaccuracies caused by the limited machine accuracy of the computer used, between the implicit and the parametric features there would be no difference of the convergence behavior and intermediate results of the outer iteration cycles using the Algorithms II (2.28) and III (2.32). This fact was of considerable assistance in developing and testing the four new ODF algorithms.

Once all the formulas for the four algorithms had been presumably correctly derived and coded into the program source, it was possible to mutually check the correctness of the whole work between implicit and parametric features. The numerical content of (2.28) or (2.32) were compared for some typical model features such as an ellipsoid possessing form, position, and three rotation parameters and described in both implicit and parametric form. If any misconception or fault has occurred in the mathematical formulations and program-coding task, the numerical content of (2.28) or (2.32) between implicit and parametric features is different. Once the algorithmic structures of the four ODF algorithms have been correctly implemented and stabilized, their extension to other model features is trivial. As a result, only the FHG (3.21) or the XHG matrix (4.25) of the standard model features (1.1) or (1.2) needs to be derived, respectively. The algorithmic features of the four new ODF algorithms can be summarized as follows:

- Estimation of the model parameters minimizing the square sum of the shortest distances between the model feature and the given points
- Generally applicable to implicit and parametric curves/surfaces defined in 2-D or 3-D space

- Algorithms II and III for implicit features (2-D curve and surface) are numerically compatible with those for parametric features, respectively
- Estimation of the model parameters in terms of form, position, and rotation parameters
- Solving of the ODF problems with additional constraints
- Provision of parameter testing possibilities
- Robust and fast convergence, except for the parametric 3-D curve fitting using Algorithm II
- Computing cost and memory space usage are proportional to the number of data points
- Automatic provision of the initial parameter values for iteration
- Low implementation cost for a new model feature.

The software tool developed by us for coordinate metrology based on the new ODF algorithms has successfully passed the software testing procedures carried out by the German federal authority PTB and has obtained the supreme grade of software accuracy (higher than $0.1\,\mu\mathrm{m}$ for length unit and higher than $0.1\,\mu\mathrm{rad}$ for angle unit for all parameters of all test model features with all test data sets). The stand-alone run-time program has a total size of approximately 200 kB including all the necessary implementations for more than 20 implicit and parametric curves and surfaces in 2-D/3-D space. The object reconstruction from real 3-D measurement points taken by the stripe projection method and laser radar was used as an application example of the new ODF algorithms.

6.2 Future Work

This work terminates by proposing future work for enhancing the performance and for extending the application fields of the new ODF algorithms described.

Minimization with the L_1- and L_∞-norms. The least-squares curves/surfaces fitting to a set of given points in 2-D/3-D space can be generalized by introducing the L_p-norm cost function below:

$$(\sigma_0^p)^{\frac{1}{p}} \triangleq \left[\sum_{i=1}^{m} (w_i \|\mathbf{X}_i - \mathbf{X}'_i(\mathbf{a})\|)^p\right]^{\frac{1}{p}} \qquad \text{with} \qquad 1 \leq p \leq \infty\,.$$

The error distance between the two points $\mathbf{X}'_i$ and $\mathbf{X}_i$ inside the above L_p-norm cost function is computed according to the L_2-norm (Euclidean norm) below:

$$\begin{aligned}\|\mathbf{X}_i - \mathbf{X}'_i\| &= \sqrt{(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}(\mathbf{X}_i - \mathbf{X}'_i)} \\ &= \left[(X_i - X'_i)^2 + (Y_i - Y'_i)^2 + (Z_i - Z'_i)^2\right]^{\frac{1}{2}}\,.\end{aligned}$$

The following three ODF problems are of particular interest for applications.

- ODF problem with the L_1-norm cost function:

$$\min_{\mathbf{a}} \min_{\{\mathbf{X}'_i\}_{i=1}^m \in S} \sigma_0^1(\{\mathbf{X}'_i(\mathbf{a})\}_{i=1}^m) = \min_{\mathbf{a}} \sum_{i=1}^m \left(w_i \min_{\mathbf{X}'_i \in S} \|\mathbf{X}_i - \mathbf{X}'_i(\mathbf{a})\| \right)$$

- ODF problem with the L_2-norm cost function:

$$\min_{\mathbf{a}} \min_{\{\mathbf{X}'_i\}_{i=1}^m \in S} \sigma_0^2(\{\mathbf{X}'_i(\mathbf{a})\}_{i=1}^m) = \min_{\mathbf{a}} \sum_{i=1}^m \left(w_i \min_{\mathbf{X}'_i \in S} \|\mathbf{X}_i - \mathbf{X}'_i(\mathbf{a})\| \right)^2$$

- ODF problem with the L_∞-norm cost function:

$$\begin{aligned} &\lim_{p\to\infty} \min_{\mathbf{a}} \min_{\{\mathbf{X}'_i\}_{i=1}^m \in S} [\sigma_0^p(\{\mathbf{X}'_i(\mathbf{a})\}_{i=1}^m)]^{\frac{1}{p}} \\ &= \lim_{p\to\infty} \min_{\mathbf{a}} \left[\sum_{i=1}^m \left(w_i \min_{\mathbf{X}'_i \in S} \|\mathbf{X}_i - \mathbf{X}'_i(\mathbf{a})\| \right)^p \right]^{\frac{1}{p}} \\ &= \min_{\mathbf{a}} \max_{i} \{ w_i \min_{\mathbf{X}'_i \in S} \|\mathbf{X}_i - \mathbf{X}'_i(\mathbf{a})\| \}_{i=1}^m . \end{aligned}$$

Independent of the applied L_p-norms, the procedures for finding the minimum distance points $\{\mathbf{X}'_i\}_{i=1}^m$ on the model feature S from each given point $\{\mathbf{X}_i\}_{i=1}^m$ (see Sect. 3.1 and Sect. 4.1) are common to all of the above ODF problems.

This work has dealt with the standard L_2-norm cost functions σ_0^2 as shown in (2.20) and (2.21) (thus, least-squares problems). Parameter estimation using the L_1-norm is known to be robust regarding outliers. The L_∞-norm is called the Tchebychev norm or minmax norm and leads the ODF problem to the minimum-zone problem which is of practical interest when determining the geometric tolerances of workpieces. If the new ODF algorithms described in this work could be extended to the L_1- and L_∞-norm cost functions, they would be more powerful tools for various industrial applications.

Parallel processing for real-time applications. The time-consuming part of the new ODF algorithms is finding of the minimum distance points $\{\mathbf{X}'_i\}_{i=1}^m$ (inner iteration). It was possible for us to accelerate the procedures by utilizing the minimum distance points found inside the last outer iteration cycle (Sect. 3.1.4 and Sect. 4.1.4). However, the running time of a standard PC available on the current market is still too long for real-time applications (for fitting an ellipsoid to a typical set of approximately 1000 points, about one second is required).

Note that the minimum distance points $\{\mathbf{X}'_i\}_{i=1}^m$ are found sequentially and independently of each other using the current implementation of these new ODF algorithms. If the minimum distance points could be found in parallel by auxiliary multi-processing hardware, the overall running time of a PC could be reduced to well below one second with most ODF tasks arising in usual applications.

Concurrent reconstruction of multiple objects. In Sect. 2.3.3 the reconstruction of multiple objects from their measurement points was briefly mentioned, using which it should be possible to satisfy certain geometric constraints on object models. Geometric constraints include the known size, position or orientation of an object model, or the relative sizes, positions and orientations between multiple object models. The relative geometric constraints are usually characterized by their geometric interpretation such as parallelism, orthogonality, concentricity, continuity, tangent, contact condition, etc. Because the model parameters a are grouped and simultaneously estimated by the new ODF algorithms in terms of form, position, and rotation parameters, the mathematical formulations of the geometric constraints and their integration into the new ODF algorithms are straightforward. This is demonstrated by several examples of constrained model fitting given in this work (Fig. 2.3b, Sect. 3.3.2, Fig. 5.4, and [8]).

The q geometric constraints on the model parameters $\{\mathbf{a}_i\}_{i=1}^{n}$ of n object models can be formulated as follows

$$f_{ci}(\mathbf{a}_1,\ldots,\mathbf{a}_n) - \mathrm{const}_i = 0 \qquad \text{with} \qquad i = 1,\ldots,q\ ,$$

or, in vector form

$$\mathbf{f}_c(\mathbf{a}_1,\ldots,\mathbf{a}_n) - \mathbf{const} = \mathbf{0}\ .$$

Then, for example, the linear equation system of the distance-based algorithm for constrained model fitting (2.35) can be extended to n object models which are interrelated through the q geometric constraints as below:

$$\begin{pmatrix} (\mathbf{PJ})_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & (\mathbf{PJ})_n \\ \mathbf{W}_c\mathbf{J}_{c,\mathbf{a}_1} & \cdots & \mathbf{W}_c\mathbf{J}_{c,\mathbf{a}_n} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{a}_1 \\ \vdots \\ \Delta\mathbf{a}_n \end{pmatrix} = - \begin{pmatrix} (\mathbf{Pd})_1 \\ \vdots \\ (\mathbf{Pd})_n \\ \mathbf{W}_c(\mathbf{f}_c(\mathbf{a}_1,\ldots,\mathbf{a}_n) - \mathbf{const}) \end{pmatrix}, \tag{6.1}$$

where

$$\mathbf{J}_{c,\mathbf{a}_i} = \begin{pmatrix} \dfrac{\partial f_{c1}}{\partial \mathbf{a}_i} \\ \vdots \\ \dfrac{\partial f_{cq}}{\partial \mathbf{a}_i} \end{pmatrix} \qquad \text{with} \qquad i = 1,\ldots,n\ .$$

Both implicit and parametric features may be embedded in (6.1) according to the distance-based algorithm (2.35) or the coordinate-based algorithm (2.36) in an arbitrary combination. Also, (6.1) is capable of integrating multiple point cloud patches taken from the same set of measurement objects but with different viewing angles.

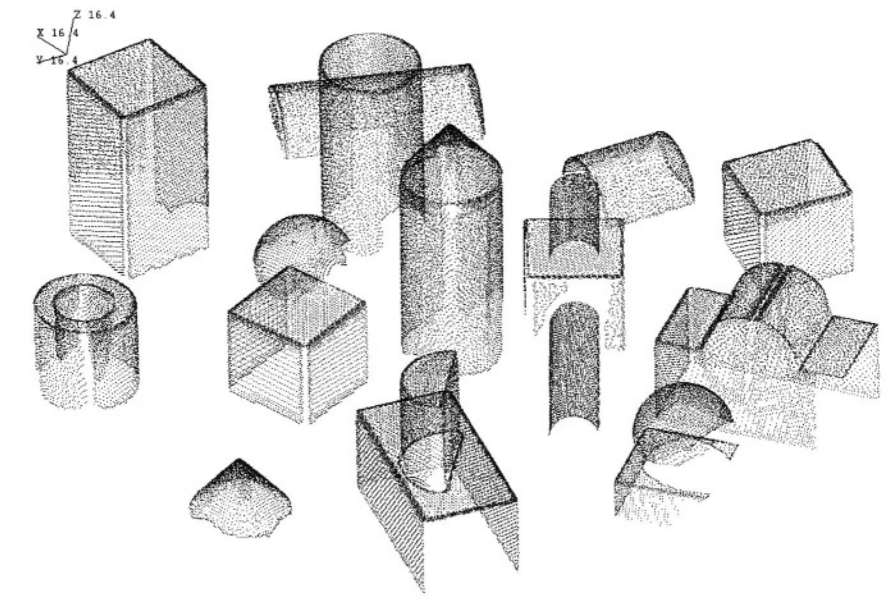
If the ultimate goal is to automatically reconstruct multiple *unknown* objects in a point cloud, much work needs to be done in the future. Even once the point cloud has been semi-automatically segmented and associated with the object models (see Fig. 5.6), the automatic determination of the number and type of appropriate geometric constraints between the object models is a challenging task. In particular, the detection and elimination of the contradictory constraints is essential to a successful reconstruction of multiple objects. Otherwise, (6.1) is ill-conditioned and the iteration with (6.1) will consequently not converge.

As it will not be possible to develop an algorithmic technique for fully automatic reconstruction of multiple objects in the near future, a semi-automatic solution is encouraged where the computer and human operator work interactively. Section 5.3 described semi-automatic object reconstruction without considering the geometric interrelations between the object models. The semi-automatic reconstruction of *multiple objects with geometric interrelations* is being pursued in our current research work.

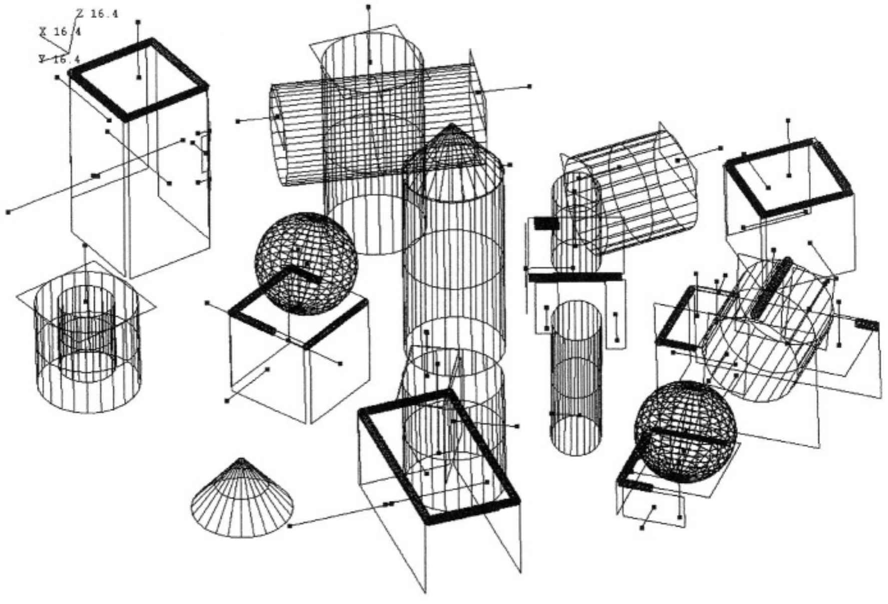
Automatic object recognition and reconstruction by integrating the point reflectance image into the point cloud. Recently, as an updated version of the semi-automatic software tool presented in Chap. 5, we have developed a software tool for fully automatic segmentation and parameter estimation of exact features (plane, sphere, cylinder, cone and torus) in point cloud (see Fig. 6.1) and, integrated the software tool in a commercial software package for reverse engineering [55]. On the other hand, it may only be possible to realize a folly automatic recognition and reconstruction of *multiple interrelated* objects - which is comparable to the ability of human intelligence - through the use of highly sophisticated hardware and software techniques capable of analyzing all available information such as point cloud, object database, CAD information, object surface color and texture, etc. Certainly, all the necessary technical and theoretical tools will not be available in the near future. Nevertheless, the aim to increase the degree of automation of object recognition and reconstruction is ever-present.

Generally, an optical 3-D measuring device supplies not only a point cloud but also the surface image of an object (e.g., the point reflectance image using laser radar or the gray value image using the stripe projection method). The object surface image is usually taken from the same viewing (measuring) angle as that with the point cloud (range image), hence it provides each measurement point with the corresponding intensity value of the object surface point (or, vice versa, the point cloud provides each pixel of the object surface image with the corresponding coordinate values of the object surface point).

The point cloud and object surface image are complementary from the viewpoint of information processing. The point cloud contains the coordinate information of each object point but not sufficient information about the object connectivity or boundary. Such a balance of information is reversed within the object surface image. Thus, whilst segmentation in a point cloud is a difficult and time-consuming task, a variety of automatic and efficient techniques for image segmentation are currently available as described in literature. Integrated processing of an object surface image and point cloud will considerably increase the degree of automation of the overall object recognition and reconstruction procedures.



(a)



(b)

Fig. 6.1. Fully automatic segmentation and parameter estimation of exact features in point cloud: (a) Point cloud (about 68 600 points); (b) Detected exact features

This page intentionally left blank

References

1. Adcock, R.J.: Note on the method of least squares. The Analyst **4** (1877) 183–184
2. Adcock, R.J.: A problem in least squares. The Analyst **5** (1878) 53–54
3. Angel, E.: Interactive Computer Graphics: A Top-Down Approach with OpenGL. 3rd Ed., Addison-Wesley (2002)
4. Ahn, S.J., Oberdorfer, B.: Optoelektronisch erfaßbares Identifizierungs- oder Zielelement sowie Verfahren zu seiner Erfassung. German Patent 196 32 058 C1 (1998)
5. Ahn, S.J.: Calibration of the Stripe Projecting 3D-Measurement System. Proc. 13th Korea Automatic Control Conf. (KACC 98). Pusan, Korea (1998) 1857–1862 (Korean)
6. Ahn, S.J., Rauh, W.: Geometric least squares fitting of circle and ellipse. Int'l J. of Pattern Recognition and Artificial Intelligence **13** (1999) 987–996
7. Ahn, S.J., Rauh, W., Kim, S.I.: Circular coded target for automation of optical 3D-measurement and camera calibration. Int'l J. of Pattern Recognition and Artificial Intelligence **15** (2001) 905–919
8. Ahn, S.J., Rauh, W., Warnecke, H.-J.: Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola. Pattern Recognition **34** (2001) 2283–2303
9. Ahn, S.J., Rauh, W., Cho, H.S., Warnecke, H.-J.: Orthogonal Distance Fitting of Implicit Curves and Surfaces. IEEE Trans. Pattern Analysis and Machine Intelligence **24** (2002) 620–638
10. Ahn, S.J., Rauh, W, Westkämper, E.: Fitting of Parametric Space Curves and Surfaces by Using the Geometric Error Measure. Proc. 24th DAGM Symp. Pattern Recognition. Lecture Notes in Computer Science **2249** (2002) 548–556
11. Ahn, S.J., Effenberger, I., Rauh, W., Cho, H.S., Westkämper, E.: Automatic segmentation and model identification in unordered 3D-point cloud. Proc. SPIE Conf. Optomechatronic Systems III **4902** (2002) 723–733
12. Alciatore, D., Miranda, R.: The Best Least-Squares Line Fit. in Graphics Gems V. A.W. Paeth (Ed.), Academic Press (1995) 91–97
13. Altschuler, M.D., Altschuler, B.R., Taboada, J.: Measuring surfaces space-coded by a laser-projected dot matrix. Proc. SPIE Conf. Imaging Applications for Automated Industrial Inspection and Assembly **182** (1979) 187–191
14. Anthony, G.T., Anthony, H.M., Cox, M.G., Forbes, A.B.: The parametrization of fundamental geometric form. BCR Report EUR 13517 EN. Commission of the European Communities, Luxemburg (1991)
15. Arun, K.S., Huang, T.S., Blostein, S.D.: Least-Squares Fitting of Two 3-D Point Sets. IEEE Trans. Pattern Analysis and Machine Intelligence **9** (1987) 698–700
16. Barr, A.H.: Superquadrics and Angle-Preserving Transformations. IEEE Computer Graphics and Applications **1** (1981) 11–23
17. Beer, F.P., Johnston, E.R.: Vector Mechanics for Engineers: Dynamics. McGraw-Hill International (1977)
18. Bennamoun, M., Boashash, B.: A Structural-Description-Based Vision System for Automatic Object Recognition. IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics **27** (1997) 893–906

19. Besl, P.J., McKay, N.D.: A Method for Registration of 3-D Shapes. IEEE Trans. Pattern Analysis and Machine Intelligence **14** (1992) 239–256
20. Boggs, P.T., Byrd, R.H., Schnabel, R.B.: A stable and efficient algorithm for nonlinear orthogonal distance regression. SIAM J. of Scientific and Statistical Computing **8** (1987) 1052–1078
21. Boggs, P.T., Donaldson, J.R., Byrd, R.H., Schnabel, R.B.: Algorithm 676 - ODRPACK: Software for Weighted Orthogonal Distance Regression. ACM Trans. Mathematical Software **15** (1989) 348–364
22. Bookstein, F.L.: Fitting conic sections to scattered data. Computer Graphics and Image Processing **9** (1979) 56–71
23. Bronštejn, I.N., Semendjajew, K.A., Musiol, G., Mühlig, H.: Taschenbuch der Mathematik. 2. Auflage, Verlag Harri Deutsch, Thun, Germany (1995)
24. Busch, K., Wäldele, F.: Testing Coordinate Measuring Machine Algorithms - Phase II: Inquiry on Software for Coordinate Metrology. BCR Report EUR 13418 EN. Commission of the European Communities, Luxemburg (1991)
25. Butler, B.P., Forbes, A.B., Harris, P.M.: Algorithms for Geometric Tolerance Assessment. Report DITC 228/94. NPL, Teddington, UK (1994)
26. Cao, X., Shrikhande, N., Hu, G.: Approximate orthogonal distance regression method for fitting quadric surfaces to range data. Pattern Recognition Letters **15** (1994) 781–796
27. do Carmo, M.P.: Differential Geometry of Curves and Surfaces. Prentice-Hall, Englewood Cliffs, NJ(1976)
28. Draper, N.R., Smith, H.: Applied Regression Analysis. 3rd ed., John Wiley and Sons, New York (1998)
29. Drieschner, R., Bittner, B., Elligsen, R., Wäldele, F.: Testing Coordinate Measuring Machine Algorithms: Phase II. BCR Report EUR 13417 EN. Commission of the European Communities, Luxemburg (1991)
30. Farin, G., Hoschek, J., Kim, M.-S.: Handbook of Computer Aided Geometric Design. Eds., Elsevier Science, Amsterdam (2002)
31. Faugeras, O.D., Hebert, M.: A 3-D recognition and positioning algorithm using geometrical matching between primitive surfaces. Proc. 8th Int'l Joint Conf. Artificial Intelligence. Karlsruhe, Germany (1983) 996–1002
32. Fitzgibbon, A.W., Pilu, M., Fisher, R.B.: Direct Least Squares Fitting of Ellipses. Proc. 13th Int'l Conf. on Pattern Recognition. Vienna (1996) 253–257
33. Fletcher, R.: Practical methods of optimization. John Wiley and Sons, New York (1987)
34. Gander, W., Golub, G.H., Strebel, R.: Least-squares fitting of circles and ellipses. BIT **34** (1994) 558–578
35. Gardiner, M.: The superellipse: A curve that lies between the ellipse and the rectangle. Scientific American **213** (1965) 222–234
36. Gauss, C.F.: Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections [Theoria motus corporum coelestium in sectionibus conicis solem ambientum]. First published in 1809, translation by C.H. Davis. Dover, New York (1963)
37. Geise, G., Schipke, S.: Ausgleichsgerade, -kreis, -ebene und -kugel im Raum. Mathematische Nachrichten **62** (1974) 65–75
38. George, A., Liu, J.W.H.: Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall, Englewood Cliffs, NJ (1981)
39. Goldman, R.N.: Two Approaches to a Computer Model for Quadric Surfaces. IEEE Computer Graphics and Applications **3** (1983) 21–24
40. Golub, G.H., Reinsch, C.: Singular value decomposition and least squares solutions. Numerische Mathematik **14** (1970) 403–120
41. Gray, F.: Pulse code communication. US Patent 2 632 058 (1953)
42. Harvie, A., The intercomparison of three-dimensional measurements taken from coordinate measuring machines (CMMs). BCR Report EUR 10100. Commission of the European Communities, Luxemburg (1985)

43. Heinz, I., Mettenleiter, M., Härtl, F., Fröhlich, C.: 3-D Ladar for Inspection of Real World Environments. Proc. 5th Conf. Optical 3-D Measurement Techniques. Vienna (2001) 10–17
44. Helfrich, H.-P., Zwick, D.: A trust region method for implicit orthogonal distance regression. Numerical Algorithms **5** (1993) 535–545
45. Helfrich, H.-P., Zwick, D.: A trust region algorithm for parametric curve and surface fitting. J. of Computational and Applied Mathematics **73** (1996) 119–134
46. Horn, B.K.P.: Closed-form solution of absolute orientation using unit quaternions. J. of the Optical Society of America **A-4** (1987) 629–642
47. Hoschek, J., Schneider, F.-J., Wassum, P.: Optimal approximate conversion of spline surfaces. Computer Aided Geometric Design **6** (1989) 293–306
48. Hu, G., Shrikhande, N.: Estimation of surface parameters using orthogonal distance criterion. Proc. 5th Int'l Conf. Image Processing and its Applications. London, UK (1995) 345–349
49. Kasa, I.: A Circle Fitting Procedure and its Error Analysis. IEEE Trans. Instrumentation and Measurement **25** (1976) 8–14
50. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Quart. Applied Mathematics **2** (1944) 164–168
51. Lukacs, G., Marshall, A.D., Martin, R.R.: Faithful Least-Squares Fitting of Spheres, Cylinders, Cones and Tori for Reliable Segmentation. Proc. 5th European Conf. on Computer Vision (ECCV '98). Freiburg, Germany. vol. 1 (1998) 671–686
52. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. SIAM J. on Applied Mathematics **11** (1963) 431–41
53. Marshall, D., Lukacs, G., Martin, R.: Robust Segmentation of Primitives from Range Data in the Presence of Geometric Degeneracy. IEEE Trans. Pattern Analysis and Machine Intelligence **23** (2001) 304–314
54. N.N.: ABW GmbH. http://www.abw-3d.de/abw/startseite/startseite-en.php (2004)
55. N.N.: Knotenpunkt GmbH. http://www.knotenpunkt.com/home_fs_E.htm (2004)
56. N.N.: 2004 3D Laser Scanner Hardware and Software Survey. Point of Beginning, http://www.pobonline.com/FILES/HTML/PDF/0104laser-survey.pdf (2004) 20–25
57. N.N.: CAP - Combined Adjustment Program: A software package for a combined adjustment of photogrammetric and geodetic networks. K2-Photogrammetry, http://www.k2-photogrammetry.de/products/Capinfo.pdf (2000)
58. N.N.: Coordinate metrology; geometrical fundamental principles, terms and definitions. German Standard DIN 32880-1. Beuth Verlag, Berlin (1986)
59. N.N.: Industrial automation systems and integration - Product data representation and exchange - Part 1: Overview and fundamental principles. Int'l Standard ISO 10303. ISO, Geneva, Switzerland (2002)
60. N.N.: Geometrical Product Specifications (GPS) - Acceptance and reverification test for coordinate measuring machines (CMM) - Part 6: Estimation of errors in computing Gaussian associated features. Int'l Standard ISO 10360-6. ISO, Geneva, Switzerland (2001)
61. Pearson, K.: On Lines and Planes of Closest Fit to Systems of Points in Space. The Philosophical Magazine, Series 6-**2** (1901) 559–572
62. Piegl, L., Tiller, W.: The NURBS Book. 2nd Ed., Springer, Berlin (1997)
63. Porta, C., Wäldele, F.: Testing of Three Coordinate Measuring Machine Evaluation Algorithms. BCR Report EUR 10909 EN. Commission of the European Communities, Luxemburg (1986)
64. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press, Cambridge, UK (1988)
65. Rivlin, T.J.: Approximation by circles. in Approximation Theory II. G.G. Lorentz et al. (Eds.), Academic Press, New York (1976) 513–517

66. Rosin, P.L.: A note on the least squares fitting of ellipses. Pattern Recognition Letters **14** (1993) 799–808
67. Rosin, P.L.: Analyzing Error of Fit Functions for Ellipses. Pattern Recognition Letters **17**(1996)1461–1470
68. Rosin, P.L.: Fitting Superellipses. IEEE Trans. Pattern Analysis and Machine Intelligence **22** (2000) 726–732
69. Sabata, B., Aggarwal, J.K.: Estimation of Motion from a Pair of Range Images: A Review. CVGIP: Image Understanding **54** (1991) 309–324
70. Safaee-Rad, R., Tchoukanov, I., Benhabib, B., Smith, K.C.: Accurate Parameter Estimation of Quadric Curves from Grey-Level Images. CVGIP: Image Understanding **54** (1991)259–274
71. Sampson, P.D.: Fitting conic sections to 'very scattered' data: An iterative refinement of the Bookstein algorithm. Computer Graphics and Image Processing **18** (1982) 97–108
72. von Seggern, D.H.: CRC standard curves and surfaces. 2nd Ed., CRC Press, Boca Raton (1993)
73. Solina, F., Bajcsy, R.: Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations. IEEE Trans. Pattern Analysis and Machine Intelligence **12** (1990) 131–147
74. Sourlier, D.: Three Dimensional Feature Independent Bestfit in Coordinate Metrology. Ph.D. Thesis 11319. ETH Zurich, Switzerland (1995)
75. Sullivan, S., Sandford, L., Ponce, J.: Using Geometric Distance Fits for 3-D Object Modeling and Recognition. IEEE Trans. Pattern Analysis and Machine Intelligence **16** (1994) 1183–1196
76. Taubin, G.: Estimation of Planar Curves, Surfaces, and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation. IEEE Trans. Pattern Analysis and Machine Intelligence **13** (1991) 1115–1138
77. Turner, D.A.: The approximation of Cartesian coordinate data by parametric orthogonal distance regression. Ph.D. Thesis. University of Huddersfield, UK (1999)
78. Ullrich, A., Reichert, R., Schwarz, R., Riegl, J.: Time-of-flight-based 3-D imaging sensor with true-color channel for automated texturing. Proc. 5th Conf. Optical 3-D Measurement Techniques, Vienna (2001) 2–9
79. Voss, K., Süße, H.: A New One-Parametric Fitting Method for Planar Objects. IEEE Trans. Pattern Analysis and Machine Intelligence **21** (1999) 646–651
80. Wahl, F.M.: A Coded-Light Approach for 3-Dimensional (3D) Vision. Research Report 1452. IBM Zurich Research Laboratory, Switzerland (1984)
81. Wäldele, W.: Intercomparison of three coordinate measuring machine measurements: synthesis report, Part 1: Geometrical standard. BCR Report EUR 11853 EN. Commission of the European Communities, Luxemburg (1988)
82. Wäldele, F., Bittner, B., Busch, K., Drieschner, R., Elligsen, R.: Tesing of coordinate measuring machine software. Precision Engineering **15** (1993) 121–123
83. Warnecke, H.-J.: Übersicht über den Entwicklungsstand bei Mehrkoordinaten-Messgeräten. Proc. Fachtagung Erfahrungsaustausch Drei-Koordinaten-Messgeräte. Fraunhofer IPA, Stuttgart, Germany (1977)
84. Warnecke, H.-J., Dutschke, W.: Fertigungsmeßtechnik: Handbuch für Industrie und Wissenschaft. Springer, Berlin (1984)
85. Weckenmann, A.: Notwendigkeit und Möglichkeiten der Normung von Antaststrategien und Auswertegrundsätzen. VDI-Berichte 529: Koordinatenmeßtechnik (1984) 209–227
86. Weckenmann, A., Heinrichowski, M.: Problems with software for running coordinate measuring machines. Precision Engineering **7** (1985) 87–91
87. Wollersheim, H.-R.: Theorie und Lösung ausgewählter Probleme der Form- und Lageprüfung auf Koordinaten-Meßgeräten. Dissertation. Fortschritt Bericht 8(78). RWTH Aachen, Germany (1984)

Index

A. Implementation Examples

In this work, the ODF algorithms are described generally for implicit and parametric curves/surfaces. In practice, they are implemented in a highly modular manner. The algorithms are applicable for fitting a general curve/surface to a set of given data points in 2-D/3-D space. To implement them for a new curve/surface, the user merely needs to supply the FHG matrix (3.21) or XHG matrix (4.25) of the standard model feature (1.1) or (1.2), respectively (see Sect. B.2). Nevertheless, in order to assist the interested reader in realizing his/her own implementation, two examples of how some key intermediate equations really look like with the ODF of implicit/parametric ellipse are given below.

A.1 Implicit 2-D Ellipse (Chap. 3)

Parameters of an ellipse in XY-plane ($p = l + n + s = 2 + 2 + 1 = 5,\ q = 0$):

$$\mathbf{a} \triangleq (a, b, X_o, Y_o, \kappa)^T .$$

Standard feature equation (1.1) defined in xy frame:

$$f(a, b, \mathbf{x}) \triangleq \frac{1}{2}\left(\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1\right) = 0 .$$

Rigid body motion (1.3) of the model in XY frame:

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} C_\kappa & -S_\kappa \\ S_\kappa & C_\kappa \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} X_o \\ Y_o \end{pmatrix} \quad \text{with} \quad C_\kappa = \cos(\kappa)\ ,\ S_\kappa = \sin(\kappa) .$$

FHG matrix (3.21):

$$\nabla f = \begin{pmatrix} x/a^2 \\ y/b^2 \end{pmatrix} , \quad \mathbf{H} = \begin{pmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{pmatrix} , \quad \mathbf{G} = \begin{pmatrix} -x^2/a^3 & -y^2/b^3 \\ -2x/a^3 & 0 \\ 0 & -2y/b^3 \end{pmatrix} .$$

Minimum distance condition (3.5):

$$\mathbf{f}(a, b, \mathbf{x}_i, \mathbf{x}) \triangleq \begin{pmatrix} (x^2/a^2 + y^2/b^2 - 1)/2 \\ (y_i - y)x/a^2 - (x_i - x)y/b^2 \end{pmatrix} = \mathbf{0} .$$

Linear equation system (3.6) of the generalized Newton method:

$$\left(\begin{pmatrix} 0 & 0 \\ y_i - y & -(x_i - x) \end{pmatrix}\begin{pmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{pmatrix} + \begin{pmatrix} x/a^2 & y/b^2 \\ y/b^2 & -x/a^2 \end{pmatrix}\right)\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \\ -\begin{pmatrix} \left(x^2/a^2 + y^2/b^2 - 1\right)/2 \\ (y_i - y)x/a^2 - (x_i - x)y/b^2 \end{pmatrix} .$$

Linear equation system (3.9) of the method of Lagrangian multipliers:

$$\begin{pmatrix} 2 + \lambda/a^2 & 0 & x/a^2 \\ 0 & 2 + \lambda/b^2 & y/b^2 \\ x/a^2 & y/b^2 & 0 \end{pmatrix}\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta\lambda \end{pmatrix} = \begin{pmatrix} 2(x_i - x) - \lambda x/a^2 \\ 2(y_i - y) - \lambda y/b^2 \\ -\left(x^2/a^2 + y^2/b^2 - 1\right)/2 \end{pmatrix} .$$

Jacobian matrix $\mathbf{J}_{d_i,\mathbf{a}}$ (3.17) for the distance-based algorithm:

$$\mathbf{J}_{d_i,\mathbf{a}} = \begin{pmatrix} -g_i x_i'^2/a^3 & -g_i y_i'^2/b^3 & 0 & 0 & 0 \end{pmatrix} \\ -\frac{1}{d_i}\begin{pmatrix} X_i - X_i' \\ Y_i - Y_i' \end{pmatrix}^{\mathrm{T}}\begin{pmatrix} 0 & 0 & 1 & 0 & -(Y_i' - Y_o) \\ 0 & 0 & 0 & 1 & X_i' - X_o \end{pmatrix} ,$$

with

$$g_i = \frac{\operatorname{sign}\left((x_i - x_i')x_i'/a^2 + (y_i - y_i')y_i'/b^2\right)}{\sqrt{(x_i'/a^2)^2 + (y_i'/b^2)^2}} .$$

Jacobian matrix $\mathbf{J}_{\mathbf{X}_i',\mathbf{a}}$ (3.19) for the coordinate-based algorithm:

$$\mathbf{J}_{\mathbf{X}_i',\mathbf{a}} = -\begin{pmatrix} C_\kappa & -S_\kappa \\ S_\kappa & C_\kappa \end{pmatrix}\left(\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\right)^{-1}\left.\left(\frac{\partial\mathbf{f}}{\partial\mathbf{x}_i}\frac{\partial\mathbf{x}_i}{\partial\mathbf{a}} + \frac{\partial\mathbf{f}}{\partial\mathbf{a}}\right)\right|_{\mathbf{x}=\mathbf{x}_i'} \\ + \begin{pmatrix} 0 & 0 & 1 & 0 & -(Y_i' - Y_o) \\ 0 & 0 & 0 & 1 & X_i' - X_o \end{pmatrix} ,$$

with

$$\frac{\partial\mathbf{f}}{\partial\mathbf{x}} = \begin{pmatrix} 0 & 0 \\ y_i - y & -(x_i - x) \end{pmatrix}\begin{pmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{pmatrix} + \begin{pmatrix} x/a^2 & y/b^2 \\ y/b^2 & -x/a^2 \end{pmatrix} ,$$

$$\frac{\partial\mathbf{f}}{\partial\mathbf{x}_i} = \begin{pmatrix} 0 & 0 \\ -y/b^2 & x/a^2 \end{pmatrix} , \qquad \frac{\partial\mathbf{x}_i}{\partial\mathbf{a}} = \begin{pmatrix} 0 & 0 & -C_\kappa & -S_\kappa & y_i \\ 0 & 0 & S_\kappa & -C_\kappa & -x_i \end{pmatrix} ,$$

$$\frac{\partial\mathbf{f}}{\partial\mathbf{a}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & y_i - y & -(x_i - x) \end{pmatrix}\begin{pmatrix} -x^2/a^3 & -y^2/b^3 & 0 & 0 & 0 \\ -2x/a^3 & 0 & 0 & 0 & 0 \\ 0 & -2y/b^3 & 0 & 0 & 0 \end{pmatrix} .$$

A.2 Parametric 3-D Ellipse (Chap. 4)

Parameters of a 3-D ellipse ($p = l + n + s = 2 + 3 + 3 = 8,\ q = 0$):

$$\mathbf{a} \triangleq (a, b, X_o, Y_o, Z_o, \omega, \varphi, \kappa)^{\mathrm{T}} .$$

Standard feature description (1.2) defined in xyz frame:

$$\mathbf{x}(a, b, u) \triangleq \begin{pmatrix} a\cos u \\ b\sin u \\ 0 \end{pmatrix} \qquad \text{with} \qquad -\pi < u \leq \pi .$$

Rigid body motion (1.3) of the model in XYZ frame:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \mathbf{R}^{\mathrm{T}}_{\omega,\varphi,\kappa} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} X_o \\ Y_o \\ Z_o \end{pmatrix} .$$

XHG matrix (4.25):

$$\frac{\partial \mathbf{x}}{\partial u} = \begin{pmatrix} -a\sin u \\ b\cos u \\ 0 \end{pmatrix} , \qquad \mathbf{H} = \begin{pmatrix} -a\cos u \\ -b\sin u \\ 0 \end{pmatrix} ,$$

$$\mathbf{G}_0 = \begin{pmatrix} \cos u & 0 \\ 0 & \sin u \\ 0 & 0 \end{pmatrix} , \qquad \mathbf{G}_1 = \begin{pmatrix} -\sin u & 0 \\ 0 & \cos u \\ 0 & 0 \end{pmatrix} .$$

Minimum distance condition (4.5):

$$f(\mathbf{x}_i, \mathbf{x}(a, b, u)) \triangleq - \begin{pmatrix} x_i - a\cos u \\ y_i - b\sin u \\ z_i \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -a\sin u \\ b\cos u \\ 0 \end{pmatrix} = 0 .$$

Linear equation system (4.6):

$$\left(\begin{pmatrix} -a\sin u \\ b\cos u \\ 0 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -a\sin u \\ b\cos u \\ 0 \end{pmatrix} - \begin{pmatrix} x_i - a\cos u \\ y_i - b\sin u \\ z_i \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -a\cos u \\ -b\sin u \\ 0 \end{pmatrix} \right) \Delta u =$$

$$\begin{pmatrix} x_i - a\cos u \\ y_i - b\sin u \\ z_i \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -a\sin u \\ b\cos u \\ 0 \end{pmatrix} .$$

Jacobian matrix $\mathbf{J}_{\mathbf{X}'_i,\mathbf{b}}$ (4.15) for Algorithm I (ETH):

$$\mathbf{b} \triangleq (a, b, X_o, Y_o, Z_o, \omega, \varphi, \kappa, u_1, \ldots, u_m)^{\mathrm{T}} .$$

$$\begin{aligned}
\mathbf{J}_{\mathbf{X}'_i,\mathbf{b}} &= \left. \left(\mathbf{R}^{\mathrm{T}}\mathbf{G}_0 \middle| \; \mathbf{I} \; \middle| \; \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}_{\mathrm{r}}} [\mathbf{x}] \; \middle| \; \mathbf{0}_1, \cdots, \mathbf{0}_{i-1}, \mathbf{R}^{\mathrm{T}} \frac{\partial \mathbf{x}}{\partial u}, \mathbf{0}_{i+1}, \cdots, \mathbf{0}_m \right) \right|_{u=u_i} \\
&= \left(\mathbf{R}^{\mathrm{T}} \begin{pmatrix} \cos u_i & 0 \\ 0 & \sin u_i \\ 0 & 0 \end{pmatrix} \middle| \; \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \; \middle| \right. \\
&\qquad \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \omega} \begin{pmatrix} a\cos u_i \\ b\sin u_i \\ 0 \end{pmatrix}, \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \varphi} \begin{pmatrix} a\cos u_i \\ b\sin u_i \\ 0 \end{pmatrix}, \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \kappa} \begin{pmatrix} a\cos u_i \\ b\sin u_i \\ 0 \end{pmatrix} \; \Bigg| \\
&\qquad \left. \mathbf{0}_1, \cdots, \mathbf{0}_{i-1}, \mathbf{R}^{\mathrm{T}} \begin{pmatrix} -a\sin u_i \\ b\cos u_i \\ 0 \end{pmatrix}, \mathbf{0}_{i+1}, \cdots, \mathbf{0}_m \right) .
\end{aligned}$$

Jacobian matrix $\mathbf{J}_{d_i,\mathbf{a}}$ (4.22) for Algorithm II (FhG):

$$\begin{aligned}
\mathbf{J}_{d_i,\mathbf{a}} &= - \frac{(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}}{\|\mathbf{X}_i - \mathbf{X}'_i\|} \left. \left(\mathbf{R}^{\mathrm{T}}\mathbf{G}_0 \; \middle| \; \mathbf{I} \; \middle| \; \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}_{\mathrm{r}}} [\mathbf{x}] \right) \right|_{u=u'_i} \\
&= - \frac{(\mathbf{X}_i - \mathbf{X}'_i)^{\mathrm{T}}}{\|\mathbf{X}_i - \mathbf{X}'_i\|} \left(\mathbf{R}^{\mathrm{T}} \begin{pmatrix} \cos u'_i & 0 \\ 0 & \sin u'_i \\ 0 & 0 \end{pmatrix} \middle| \; \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \; \middle| \right. \\
&\qquad \left. \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \omega} \begin{pmatrix} a\cos u'_i \\ b\sin u'_i \\ 0 \end{pmatrix}, \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \varphi} \begin{pmatrix} a\cos u'_i \\ b\sin u'_i \\ 0 \end{pmatrix}, \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \kappa} \begin{pmatrix} a\cos u'_i \\ b\sin u'_i \\ 0 \end{pmatrix} \right) ,
\end{aligned}$$

with

$$\mathbf{X}'_i = \mathbf{R}^{\mathrm{T}} \begin{pmatrix} a\cos u'_i \\ b\sin u'_i \\ 0 \end{pmatrix} + \mathbf{X}_o .$$

Jacobian matrix $\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}}$ (4.23) for Algorithm III (FhG):

$$\begin{aligned}
\mathbf{J}_{\mathbf{X}'_i,\mathbf{a}} &= \left. - \mathbf{R}^{\mathrm{T}} \begin{pmatrix} -a\sin u'_i \\ b\cos u'_i \\ 0 \end{pmatrix} \left(\frac{\partial f}{\partial u} \right)^{-1} \left(\frac{\partial f}{\partial \mathbf{x}_i} \frac{\partial \mathbf{x}_i}{\partial \mathbf{a}} + \frac{\partial f}{\partial \mathbf{a}} \right) \right|_{u=u'_i} \\
&\quad + \left(\mathbf{R}^{\mathrm{T}} \begin{pmatrix} \cos u'_i & 0 \\ 0 & \sin u'_i \\ 0 & 0 \end{pmatrix} \middle| \; \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \; \middle| \; \frac{\partial \mathbf{R}^{\mathrm{T}}}{\partial \mathbf{a}_{\mathrm{r}}} \begin{bmatrix} a\cos u'_i \\ b\sin u'_i \\ 0 \end{bmatrix} \right) ,
\end{aligned}$$

with

$$\frac{\partial f}{\partial u} = \begin{pmatrix} -a\sin u \\ b\cos u \\ 0 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -a\sin u \\ b\cos u \\ 0 \end{pmatrix} - \begin{pmatrix} x_i - a\cos u \\ y_i - b\sin u \\ z_i \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -a\cos u \\ -b\sin u \\ 0 \end{pmatrix} ,$$

$$\frac{\partial f}{\partial \mathbf{x}_i} = -\begin{pmatrix} -a\sin u \\ b\cos u \\ 0 \end{pmatrix}^{\mathrm{T}} , \quad \frac{\partial \mathbf{x}_i}{\partial \mathbf{a}} = \left(\begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix} \;\middle|\; -\mathbf{R} \;\middle|\; \frac{\partial \mathbf{R}}{\partial \mathbf{a}_{\mathrm{r}}} [\mathbf{X}_i - \mathbf{X}_{\mathrm{o}}] \right) ,$$

$$\frac{\partial f}{\partial \mathbf{a}} = \left(\begin{pmatrix} -a\sin u \\ b\cos u \\ 0 \end{pmatrix}^{\mathrm{T}} \mathbf{G}_0 - \begin{pmatrix} x_i - a\cos u \\ y_i - b\sin u \\ z_i \end{pmatrix}^{\mathrm{T}} \mathbf{G}_1 \;\middle|\; \mathbf{0}^{\mathrm{T}} \;\middle|\; \mathbf{0}^{\mathrm{T}} \right) .$$

B. CMM Software Tools Fulfilling ISO 10360-6

The international standard ISO 10360-6 for testing the data processing softwares for coordinate metrology is outlined in Sect. 1.2.4. This appendix provides detailed information for implementing the general ODF algorithms on the nine test model features defined in ISO 10360-6. Our software tool has successfully passed the testing procedures carried out by the German federal authority PTB and has obtained the supreme grade of software accuracy. Section B.1 reviews the geometric definition (parameterization) of the nine model features in space and Sect. B.2 gives the detailed formulas necessary for implementing the ODF algorithms.

B.1 Curves and Surfaces Defined in ISO 10360-6

In order to reduce manufacturing cost, products are extensively designed using simple geometric elements such as lines, planes, circles, cylinders, etc. because the form and motion of manufacturing tools are usually composed of linear or circular elements. The nine model features listed in Table B.1 are the most common geometric elements found in an engineering environment, particularly as far as mechanical functional workpieces are concerned. The applied parameterization of the nine model features, i.e. the choice of model parameters that geometrically define the model feature in space, has been carefully examined within the scope of the international research activities for establishing ISO 10360-6 and is accepted as the best parameterization from the viewpoint of coordinate metrology [14], [60]. In the following sections, the basic ideas put into the applied parameterization are reviewed, which would be also helpful for parameterizing other model features and for practicing coordinate metrology.

B.1.1 Competent Parameterization

With curve/surface (model) fitting, an initial work step is to mathematically describe the model feature with unknown model parameters. Section 1.1.1 compares the different forms of the mathematical description of curves/surfaces, i.e., explicit, implicit, and parametric form. With the ODF algorithms presented in this work, there is practically no difference in results of model fitting between using the implicit form and using the parametric form, provided the model feature can be described in

Table B.1. Curves and surfaces defined in ISO 10360-6 [60]

Model feature	Parameter				Description
	Position (mm)	Orien-tation	Size (mm)	Angle (rad)	
Line (2-D)	X_o, Y_o	—	—	—	MC of the data set
	—	a, b	—	—	DC of the line
Line (3-D)	X_o, Y_o, Z_o	—	—	—	MC of the data set
	—	a, b, c	—	—	DC of the line
Plane	X_o, Y_o, Z_o	—	—	—	MC of the data set
	—	a, b, c	—	—	DC of the normal to the plane
Circle (2-D)	X_o, Y_o	—	—	—	Center of the circle
	—	—	r	—	Radius of the circle
Circle (3-D)	X_o, Y_o, Z_o	—	—	—	Center of the circle
	—	a, b, c	—	—	DC of the normal to the plane containing the circle
	—	—	r	—	Radius of the circle
Sphere	X_o, Y_o, Z_o	—	—	—	Center of the sphere
	—	—	r	—	Radius of the sphere
Cylin-der	X_o, Y_o, Z_o	—	—	—	Point on the axis of the cylinder closest to the MC of the data set
	—	a, b, c	—	—	DC of the axis of the cylinder
	—	—	r	—	Radius of the cylinder
Cone	X_o, Y_o, Z_o	—	—	—	Point on the axis of the cone closest to the MC of the data set
	—	a, b, c	—	—	DC of the axis of the cone with vector axis direction pointing towards the apex of the cone
	—	—	r	—	Radius of the cone at $\mathbf{X}_o$ measured normal to the axis of the cone
	—	—	—	ψ	Vertex angle of the cone (equal to twice of the angle between the generator of the cone and the axis)
Torus	X_o, Y_o, Z_o	—	—	—	Center of the torus
	—	a, b, c	—	—	DC of the axis of the torus
	—	—	r_1	—	Radius of the circular section of the tube of the torus
	—	—	r_2	—	Mean radius of the ring of the torus

MC: mass center, DC: direction cosines.

both forms with the same set of parameters. More essential to an accurate and reliable model fitting is the choice of the set of parameters (parameterization). In this section, the importance of competent parameterization [14] to an accurate and reliable model fitting is stressed by comparing the alternative ways of parameterizing a 2-D line in XY frame (Fig. B.1).

A line can be drawn by starting from a point $\mathbf{X}_0$ and continuing in a direction $\mathbf{r}$ ($\|\mathbf{r}\| = 1$)

$$\mathbf{X}_0 + u\mathbf{r} \qquad \text{with} \qquad -\infty \leq u \leq \infty\,. \tag{B.1}$$

The direction cosine vector $\mathbf{r}$ can be interpreted as the connecting vector between the two points $\mathbf{X}_0$ and $\mathbf{X}_0 + \mathbf{r}$ (the selection of two points is a further possibility of defining a line). The line direction can be represented also by a normal vector $\mathbf{n}$ ($\|\mathbf{n}\| = 1$)

$$\mathbf{X}^{\mathrm{T}}\mathbf{n} - d = 0 \tag{B.2}$$

with the implied starting point $d\mathbf{n}$. Nevertheless, the most familiar description form of a 2-D line is

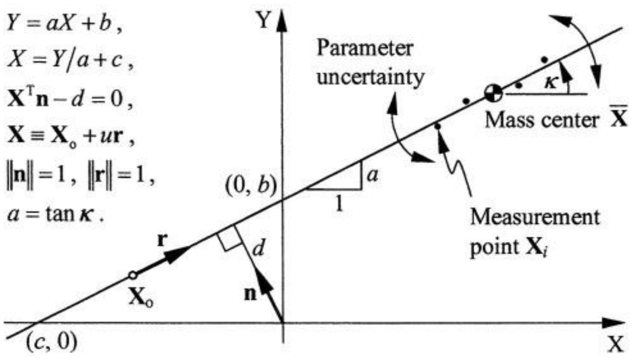
$$Y = aX + b \tag{B.3}$$

with the line slope a and the starting point $(0, b)$, or,

$$X = Y/a + c \tag{B.4}$$

with the starting point $(c, 0)$. Unfortunately, the line descriptions (B.3) and (B.4) cannot represent a vertical or a horizontal line. Thus, their use in line fitting is to be avoided in applications generating an arbitrary set of points.

The line descriptions (B.1)–(B.4) are then critically reviewed from the viewpoint of coordinate metrology aiming to accurately determine object model parameters from a set of measurement points $\{\mathbf{X}'_i\}_{i=1}^{m}$. Note that any measurement is subject to error and that any model feature only represents the measurement object



$Y = aX + b\,,$
$X = Y/a + c\,,$
$\mathbf{X}^{\mathrm{T}}\mathbf{n} - d = 0\,,$
$\mathbf{X} \equiv \mathbf{X}_0 + u\mathbf{r}\,,$
$\|\mathbf{n}\| = 1\,,\ \|\mathbf{r}\| = 1\,,$
$a = \tan\kappa\,.$
Y
Parameter uncertainty
κ
Mass center $\bar{\mathbf{X}}$
$(0, b)$
a
1
Measurement point $\mathbf{X}_i$
$\mathbf{r}$
d
$\mathbf{X}_0$
$\mathbf{n}$
$(c, 0)$
X

Fig. B.1. Mathematical description of a 2-D line

approximately. Consequently, the model parameters, which are estimated from the measurement points and represent the measurement object, are subject to contain uncertainties (variances).

Section 2.1.1 demonstrates that the ODF line passes through the mass center $\bar{\mathbf{X}}$ of the measurement points $\{\mathbf{X}_i\}_{i=1}^m$

$$\bar{\mathbf{X}} + u\mathbf{r} \tag{B.5}$$

with the direction cosine vector $\mathbf{r}$ minimizing the principal centroidal moment of $\{\mathbf{X}_i\}_{i=1}^m$. The *uncertainties* of the estimated line parameters $\bar{\mathbf{X}}$ and $\mathbf{r}$ in (B.5) are dependent on the propriety of the model selection and on the accuracy and distribution of $\{\mathbf{X}_i\}_{i=1}^m$, but not on the position and rotation of the set of points $\{\mathbf{X}_i\}_{i=1}^m$ (i.e., invariant to coordinate transformation). On the other hand, the uncertainties of the estimated position parameters d, b, c in (B.2)–(B.4) are dependent on the coordinate transformation of the set of points $\{\mathbf{X}_i\}_{i=1}^m$. For example, if the set of points lies remote from the origin of the coordinate frame, the variances of the estimated position parameters d, b, c become unnecessarily large (i.e., unreliable). Moreover, the line parameters in (B.2)–(B.4) are strongly correlated. A variational change of the line slope a should be compensated through an immediate change of the position parameters b and c because the ODF line is, independently on its slope, commanded to pass through $\bar{\mathbf{X}}$ to minimize the square sum of the distance errors (Sect. 2.1.1).

Although the line descriptions (B. 1)–(B.4) apparently represent the same line as illustrated in Fig. B. 1, the line description (B.1) with the constraint of $\|\mathbf{X}_o - \bar{\mathbf{X}}\| = 0$ is the most competent one for the purpose of line fitting to a set of measurement points. The line parameters $\bar{\mathbf{X}}$ and $\mathbf{r}$ are not correlated and their variances are invariant to coordinate transformation. Similar explanations can be given for the competent parameterization of other model features such as a 3-D line, plane, cylinder and cone as defined in ISO 10360-6 (Table B.1 and [14]).

B.1.2 Role of the Mass Center

In measuring practice, a measurement point $\mathbf{X}_i$ implies that there is a real object point. Also, the mass center $\bar{\mathbf{X}}$ of the measurement points $\{\mathbf{X}_i\}_{i=1}^m$ hints at least roughly at the position of the measurement object in space. The mass center $\bar{\mathbf{X}}$ can be directly calculated from $\{\mathbf{X}_i\}_{i=1}^m$ and is the primary information about the measurement object and should thus be fully utilized. Note that the ODF model feature of a measurement object is required to represent the measurement points as best as possible by minimizing the square sum of the shortest distances between the model feature and the measurement points. This means that the measurement object and its ODF model feature estimated from $\{\mathbf{X}_i\}_{i=1}^m$ are usually found near $\bar{\mathbf{X}}$ which is the averaging location of $\{\mathbf{X}_i\}_{i=1}^m$.

With line/plane fitting, the mass center $\bar{\mathbf{X}}$ serves as the position $\mathbf{X}_o$ of ODF line/plane, i.e. $\mathbf{X}_o = \bar{\mathbf{X}}$. Moreover, $\bar{\mathbf{X}}$ is itself a membership point of the ODF line/plane (Sect. 2.1 and Sect. B.1.1). With circle/sphere fitting, $\bar{\mathbf{X}}$ can be used as reasonable initial values for the center position $\mathbf{X}_o$ of ODF circle/sphere which is to be estimated through iteration (Sect. 2.3.5 and Sect. 3.3).

The competent parameterization of a cylinder can be accomplished in an analogous manner in the case of a 3-D line (Fig. B.1) (a cylinder is a generalization of a 3-D line with a thickness radius r (Fig. 1.3a)). A parallel motion of an ODF cylinder along the cylinder axis does not alter the spatial instance of the ODF cylinder and therefore does not cause a change in the square sum of the shortest distances of $\{\mathbf{X}_i\}_{i=1}^{m}$ to the ODF cylinder. Consequently, the position $\mathbf{X}_o$ of the ODF cylinder can be arbitrarily placed on the cylinder axis. As far as the variances and correlations of the estimated model parameters (Fig. B.1) are concerned, for similar reasons as in the case of a 3-D line, the competent parameterization of a cylinder is performed by placing $\mathbf{X}_o$ at the closest point on the cylinder axis from $\bar{\mathbf{X}}$

$$(\mathbf{X}_o - \bar{\mathbf{X}})^{\mathrm{T}}\mathbf{r} = 0 \,, \tag{B.6}$$

where $\mathbf{r}$ is the direction cosine vector of the cylinder axis (Sect. 2.3.3, Sect. 3.3.2, and Sect. B.2.7).

With cone fitting, the position $\mathbf{X}_o$ of the ODF cone could be defined at the vertex of cone (Sect. 3.3.2). However, the vertex position estimated from the measurement points is generally very unreliable and strongly correlated with the vertex angle and with the rotation parameters. For example, particularly in the case of an ODF cone with a small vertex angle, a variational change of the vertex angle causes an immediate and considerable change in the vertex position. Thus, in the same way (B.6) as with the case of an ODF cylinder, the position $\mathbf{X}_o$ of ODF cone is constrained to be the closest point on the cone axis from $\bar{\mathbf{X}}$ (Sect. 2.3.3, Sect. 3.3.2, and Sect. B.2.8). Although this definition for $\mathbf{X}_o$ of an ODF cone demands more skilled mathematical handling (Sect. B.2.8), it provides model compatibility between a cone and a cylinder which simplifies model selection and supplies the initial parameter values for the iterative cone fitting (a cylinder can be regarded as a special instance of a cone with a zero vertex angle (Fig. 1.3a)).

In summary, the mass center $\bar{\mathbf{X}}$ of the measurement points $\{\mathbf{X}_i\}_{i=1}^{m}$ acts as a quasi-reference point and implies that the *measurement object* is nearby. Through constraining the position $\mathbf{X}_o$ of ODF model feature just to be $\bar{\mathbf{X}}$ (the case of a line/plane) or the closest point on the axis of a ODF model feature from $\bar{\mathbf{X}}$ (the case of cylinder/cone), the unnecessary and undesirable increase in variances and correlations of the estimated ODF model parameters can be avoided.

B.1.3 Rotation Matrix

Any linear transformation from one coordinate frame XYZ to another frame xyz can be expressed below:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r_{\mathrm{xX}} & r_{\mathrm{xY}} & r_{\mathrm{xZ}} \\ r_{\mathrm{yX}} & r_{\mathrm{yY}} & r_{\mathrm{yZ}} \\ r_{\mathrm{zX}} & r_{\mathrm{zY}} & r_{\mathrm{zZ}} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} ,$$

or

$$\mathbf{x} = \mathbf{R}\mathbf{X} \tag{B.7}$$

with

$$\begin{aligned}
\mathbf{R} &= \begin{pmatrix}\mathbf{r}_\mathrm{x} & \mathbf{r}_\mathrm{y} & \mathbf{r}_\mathrm{z}\end{pmatrix}^\mathrm{T} ,\\
\mathbf{r}_\mathrm{x} &= \begin{pmatrix}r_{\mathrm{xX}} & r_{\mathrm{xY}} & r_{\mathrm{xZ}}\end{pmatrix}^\mathrm{T} ,\\
\mathbf{r}_\mathrm{y} &= \begin{pmatrix}r_{\mathrm{yX}} & r_{\mathrm{yY}} & r_{\mathrm{yZ}}\end{pmatrix}^\mathrm{T} ,\\
\mathbf{r}_\mathrm{z} &= \begin{pmatrix}r_{\mathrm{zX}} & r_{\mathrm{zY}} & r_{\mathrm{zZ}}\end{pmatrix}^\mathrm{T} .
\end{aligned}$$

If the two coordinate frames xyz and XYZ are rectangular (Cartesian coordinate system) and the linear transformation (B.7) preserves lengths of vectors and angles between vectors, the matrix **R** is an orthogonal matrix

$$\mathbf{R}^{-1} = \mathbf{R}^\mathrm{T} , \qquad \mathbf{R}^\mathrm{T}\mathbf{R} = \mathbf{I} ,$$

satisfying the orthogonality conditions

$$\begin{aligned}
\mathbf{r}_\mathrm{x} \cdot \mathbf{r}_\mathrm{x} &= \mathbf{r}_\mathrm{y} \cdot \mathbf{r}_\mathrm{y} = \mathbf{r}_\mathrm{z} \cdot \mathbf{r}_\mathrm{z} = 1 ,\\
\mathbf{r}_\mathrm{x} \cdot \mathbf{r}_\mathrm{y} &= \mathbf{r}_\mathrm{y} \cdot \mathbf{r}_\mathrm{z} = \mathbf{r}_\mathrm{z} \cdot \mathbf{r}_\mathrm{x} = 0 .
\end{aligned} \tag{B.8}$$

Then, the column vectors $\mathbf{r}_\mathrm{x}, \mathbf{r}_\mathrm{y}, \mathbf{r}_\mathrm{z}$ can be interpreted as the direction cosine vectors of the x-, y-, z-axes of the rotated xyz frame referenced to XYZ frame. The resulting coordinates x, y, z from (B.7) are the projections of the coordinate vector **X** in XYZ frame onto the x-, y-, z-axes

$$x = \mathbf{r}_\mathrm{x} \cdot \mathbf{X} , \qquad y = \mathbf{r}_\mathrm{y} \cdot \mathbf{X} , \qquad z = \mathbf{r}_\mathrm{z} \cdot \mathbf{X} .$$

In addition, if the coordinate frame xyz has a parallel motion (translation) $\mathbf{X}_\mathrm{o}$ referenced to XYZ frame, the coordinate transformation (B.7) becomes

$$\mathbf{x} = \mathbf{R}(\mathbf{X} - \mathbf{X}_\mathrm{o}) \qquad \text{or} \qquad \mathbf{X} = \mathbf{R}^{-1}\mathbf{x} + \mathbf{X}_\mathrm{o} .$$

The nine elements $\{r_{\mathrm{xX}}, \ldots, r_{\mathrm{zZ}}\}$ of the orthogonal rotation matrix **R** have only three degrees of freedom due to the six constraints (B.8). This means that any arbitrary rotation can be described by only three parameters (rotation parameters, Euler angles) (Euler's rotation theorem)

$$\mathbf{R} = \mathbf{R}_\kappa \mathbf{R}_\varphi \mathbf{R}_\omega .$$

Depending on the axes about which the rotations ω, φ, and κ are carried out, there are a variety of conventions for Euler angles. In this work, we have chosen the x-y-z convention

$$\begin{aligned}\mathbf{R} &\triangleq \begin{pmatrix} C_\kappa & S_\kappa & 0 \\ -S_\kappa & C_\kappa & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} C_\varphi & 0 & -S_\varphi \\ 0 & 1 & 0 \\ S_\varphi & 0 & C_\varphi \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & C_\omega & S_\omega \\ 0 & -S_\omega & C_\omega \end{pmatrix} \\ &= \begin{pmatrix} C_\varphi C_\kappa & C_\omega S_\kappa + S_\omega S_\varphi C_\kappa & S_\omega S_\kappa - C_\omega S_\varphi C_\kappa \\ -C_\varphi S_\kappa & C_\omega C_\kappa - S_\omega S_\varphi S_\kappa & S_\omega C_\kappa + C_\omega S_\varphi S_\kappa \\ S_\varphi & -S_\omega C_\varphi & C_\omega C_\varphi \end{pmatrix} ,\end{aligned} \tag{B.9}$$

with which the rotations ω, φ, and κ are carried out sequentially about the x-, y-, z-axes, i.e., about the moving axes. And, recovering the set of Euler angles $\{\omega, \varphi, \kappa\}$ from a given rotation matrix $\mathbf{R}$ is of great interest for applications

$$\omega = \tan^{-1}\frac{-r_{\mathrm{zY}}}{r_{\mathrm{zZ}}} , \quad \varphi = \tan^{-1}\frac{r_{\mathrm{zX}}}{\sqrt{r_{\mathrm{xX}}^2 + r_{\mathrm{yX}}^2}} , \quad \kappa = \tan^{-1}\frac{-r_{\mathrm{yX}}}{r_{\mathrm{xX}}} . \tag{B.10}$$

Although there is a second set of solutions $\{\omega, \varphi, \kappa\}$ due to the duality of Euler angles, the angle range of $-\pi/2 \le \varphi \le \pi/2$ has been selected using the positive square root in the formula for φ in (B. 10). The two sets of solutions form the same rotation matrix $\mathbf{R}$ according to the convention (B.9). Also, if $\varphi = \pm\pi/2$ given by the zero denominator in the formula for φ, then the formulas for ω and κ are degenerative and, according to (B.9), only $\omega + \kappa$ or $\omega - \kappa$ can be recovered from the given rotation matrix. In other words, provided with $\varphi = \pm\pi/2$, every set $\{\omega, \kappa\}$ generating the same value $\omega \pm \kappa$ forms the same rotation matrix $\mathbf{R}$ (B.9). Thus, if we obtain $\varphi = \pm\pi/2$ from (B. 10), we arbitrarily put $\kappa = 0$ and take

$$\omega = \tan^{-1}\frac{r_{\mathrm{yZ}}}{r_{\mathrm{yY}}} .$$

We would like to note the applied parameterization of the rotation of a model feature by ISO 10360-6. Not only because of the variety of conventions for Euler angles but also because of the duality and degenerative cases of Euler angles, it is not technically sound to compare and test the accuracy of the three Euler angles estimated by a model fitting algorithm. Therefore, ISO 10360-6 (see Table B.1) compares and tests the accuracy of the very facts of rotation matrix $\mathbf{R}$, namely, the direction cosine vectors $\mathbf{r}_\mathrm{x}, \mathbf{r}_\mathrm{y}, \mathbf{r}_\mathrm{z}$ that represent the pose of a model feature in space. Once the three Euler angles $\{\omega, \varphi, \kappa\}$ of a model feature have been estimated using an ODF algorithm, the three direction cosine vectors $\mathbf{r}_\mathrm{x}, \mathbf{r}_\mathrm{y}, \mathbf{r}_\mathrm{z}$ can be extracted from the resulting rotation matrix $\mathbf{R}$.

Finally, the definition is also given of the partial derivatives $\partial\mathbf{R}/\partial\mathbf{a}_\mathrm{r}$ of the rotation matrix $\mathbf{R}$ respectively to the Euler angles $\mathbf{a}_\mathrm{r} = (\omega, \varphi, \kappa)^\mathrm{T}$ which are often used in this work (see Sect. 3.2.1, Sect. 3.2.2, and Sect. 4.2.1–4.2.3),

$$\begin{aligned}\frac{\partial\mathbf{R}}{\partial\mathbf{a}_\mathrm{r}} &\triangleq \begin{pmatrix} \frac{\partial\mathbf{R}}{\partial\omega} & \frac{\partial\mathbf{R}}{\partial\varphi} & \frac{\partial\mathbf{R}}{\partial\kappa} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{R}_\kappa\mathbf{R}_\varphi\frac{\partial\mathbf{R}_\omega}{\partial\omega} & \mathbf{R}_\kappa\frac{\partial\mathbf{R}_\varphi}{\partial\varphi}\mathbf{R}_\omega & \frac{\partial\mathbf{R}_\kappa}{\partial\kappa}\mathbf{R}_\varphi\mathbf{R}_\omega \end{pmatrix} .\end{aligned}$$

In addition, together with [X] and [x], the following is defined

$$\frac{\partial \mathbf{R}}{\partial \mathbf{a}_\mathrm{r}}[\mathbf{X}] \triangleq \left(\frac{\partial \mathbf{R}}{\partial \omega}\mathbf{X} \quad \frac{\partial \mathbf{R}}{\partial \varphi}\mathbf{X} \quad \frac{\partial \mathbf{R}}{\partial \kappa}\mathbf{X} \right)$$

and

$$\frac{\partial \mathbf{R}^\mathrm{T}}{\partial \mathbf{a}_\mathrm{r}}[\mathbf{x}] \triangleq \left(\frac{\partial \mathbf{R}^\mathrm{T}}{\partial \omega}\mathbf{x} \quad \frac{\partial \mathbf{R}^\mathrm{T}}{\partial \varphi}\mathbf{x} \quad \frac{\partial \mathbf{R}^\mathrm{T}}{\partial \kappa}\mathbf{x} \right) .$$

B.1.4 Parameter Range

The competent parameterization discussed in the previous subsections preconditions an accurate and reliable model fitting as far as the low variances and correlations of the estimated model parameters are concerned. However, there are still ambiguities in model parameters which require removal during or after the iterations (2.27) and (2.31). For example, without altering its spatial instance, a cylinder can be given either of the two opposing axis directions, being recognized as two different cylinders, and vice versa. There are many similar situations with other model features which could lead to serious problems in applications. In order to remove parameter ambiguities, the following three criteria for conditioning the model parameter values during or after the iterations (2.27) and (2.31) are considered. Not forgetting that, whenever the axes of xyz frame are exchanged or inverted through any conditioning operation listed below, the minimum distance points $\{\mathbf{x}'_i\}_{i=1}^m$ in xyz frame and their location parameters $\{\mathbf{u}'_i\}_{i=1}^m$ require refreshing (see Sect. 3.1 and Sect. 4.1).

- Preventive condition: Certain model parameters *must* lie within realistic parameter ranges. For example, the size parameters of a model feature (radius r of a circle/sphere/cylinder, axis-lengths a, b, c of an ellipsoid, etc.) must be positive. Otherwise the model feature is imaginary or degenerative. If any updated size parameter of a model feature is not positive as $a_i + \alpha\Delta a_i \leq 0$ during the iterations (2.27) or (2.31), the step size factor α is appropriately decreased to keep $a_i + \alpha\Delta a_i$ positive.
- Alternative condition: Certain model parameters *should* lie within the pre-chosen parameter range among the multiple permitted ranges.
  - Many model features such as a line/plane/cylinder/torus/3-D circle having axis or plane symmetry can get one of the two opposing axis vectors or of the two opposing normal vectors without altering its spatial instance. If necessary, the axis vector or the normal vector are inverted

    $$\mathbf{r}_\mathrm{z} \leftarrow -\mathbf{r}_\mathrm{z} \qquad \text{and} \qquad \mathbf{r}_\mathrm{x} \leftrightarrow \mathbf{r}_\mathrm{y} .$$

    Depending on applications, they must have a positive or negative projection onto the X- or Y- or Z-axis of XYZ frame. This condition is checked and parameter ambiguities removed during the software test procedures by ISO 10360-6.

- If the vertex angle ψ of a cone (see Sect. B.2.8) is negative during or after iterations (2.27) or (2.31), the sign of the vertex angle is inverted together with the direction cosine vector of the cone axis

$$\psi \leftarrow -\psi \qquad \text{and} \qquad \mathbf{r}_\mathrm{z} \leftarrow -\mathbf{r}_\mathrm{z} \qquad \text{and} \qquad \mathbf{r}_\mathrm{x} \leftrightarrow \mathbf{r}_\mathrm{y} \, .$$

- The rotation parameters ω, φ, and κ of *any* model feature should lie within the range of

$$0 \le (\omega, \varphi, \kappa) < 2\pi \qquad \text{or} \qquad -\pi < (\omega, \varphi, \kappa) \le \pi \, .$$

  If necessary, ω, φ, κ are relocated

$$\omega \leftarrow \omega \pm 2\pi \qquad \text{and/or} \qquad \varphi \leftarrow \varphi \pm 2\pi \qquad \text{and/or} \qquad \kappa \leftarrow \kappa \pm 2\pi \, .$$

- With certain model features such as an ellipse/ellipsoid, the rotation parameters ω, φ, and κ should lie within the range of

$$0 \le (\omega, \varphi, \kappa) < \pi \qquad \text{or} \qquad -\pi/2 < (\omega, \varphi, \kappa) \le \pi/2 \, .$$

  If necessary, ω, φ, κ are relocated

$$\omega \leftarrow \omega \pm \pi \qquad \text{and/or} \qquad \varphi \leftarrow \varphi \pm \pi \qquad \text{and/or} \qquad \kappa \leftarrow \kappa \pm \pi \, .$$

- Relative condition: Certain size parameters *should* be arranged in the given order of size. For example, the axis-lengths a and b of an ellipsoid are exchanged to maintain the given size order of $a \ge b$ or $a \le b$

$$a \leftrightarrow b \qquad \text{and} \qquad \mathbf{r}_\mathrm{x} \leftrightarrow \mathbf{r}_\mathrm{y} \qquad \text{and} \qquad (\mathbf{r}_\mathrm{x} \leftarrow -\mathbf{r}_\mathrm{x} \quad \text{or} \quad \mathbf{r}_\mathrm{y} \leftarrow -\mathbf{r}_\mathrm{y}) \, .$$

  In this way, without altering the spatial instance of the ellipsoid, any two axis-lengths of an ellipsoid can be exchanged. This kind of operation is necessary for identification and pose determination of an object in space.

B.2 Minimum Distance Point and FHG/XHG Matrix

To apply the general ODF algorithms described in this work to a specific model feature, the FHG matrix (3.21) of the standard implicit feature equation $f(\mathbf{a}_\mathrm{g}, \mathbf{x}) = 0$ (1.1) or the XHG matrix (4.25) of the standard parametric feature description $\mathbf{x}(\mathbf{a}_\mathrm{g}, \mathbf{u})$ (1.2) of the specific model feature needs to be supplied. The following sections detail the formulas necessary for implementing the ODF algorithms on the nine model features defined in ISO 10360-6. Also, the closed form solution is given for the shortest distance point on each model feature from a given point.

B.2.1 2-D Line

A 2-D line can be described as an *implicit* curve in an xy-plane.

Standard feature equation (1.1):

$$f(\mathbf{x}) \triangleq y = 0 \, .$$

Parameters ($p = l + n + s = 0 + 2 + 1 = 3,\ q = 1$):

$$\mathbf{a}_\mathrm{g} = \varnothing \, , \qquad \mathbf{a}_\mathrm{p} = (X_\mathrm{o}, Y_\mathrm{o})^\mathrm{T} \, , \qquad a_\mathrm{r} = \kappa \, .$$

Parameter constraint (2.33):

$$f_\mathrm{c}(\mathbf{a}_\mathrm{p}) \triangleq \|\mathbf{X}_\mathrm{o} - \bar{\mathbf{X}}\| = \sqrt{(X_\mathrm{o} - \bar{X})^2 + (Y_\mathrm{o} - \bar{Y})^2} = 0 \, .$$

FHG matrix (3.21):

$$\nabla f = \begin{pmatrix} 0 \\ 1 \end{pmatrix} , \qquad \mathbf{H} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} , \qquad \mathbf{G} = \varnothing \, .$$

Minimum distance point $\mathbf{x}'_i$ from the given point $\mathbf{x}_i = \mathbf{R}_\kappa(\mathbf{X}_i - \mathbf{X}_\mathrm{o})$:

$$\mathbf{x}'_i = \begin{pmatrix} x_i \\ 0 \end{pmatrix} \, .$$

B.2.2 3-D Line

A 3-D line can be described as a *parametric* 3-D curve.

Standard feature description (1.2):

$$\mathbf{x}(u) \triangleq \begin{pmatrix} 0 \\ 0 \\ u \end{pmatrix} \qquad \text{with} \qquad -\infty \le u \le \infty \, .$$

Parameters ($p = l + n + s = 0 + 3 + 2 = 5,\ q = 1$):

$$\mathbf{a}_\mathrm{g} = \varnothing \, , \qquad \mathbf{a}_\mathrm{p} = (X_\mathrm{o}, Y_\mathrm{o}, Z_\mathrm{o})^\mathrm{T} \, , \qquad \mathbf{a}_\mathrm{r} = (\omega, \varphi)^\mathrm{T} \, .$$

Parameter constraint (2.33):

$$f_\mathrm{c}(\mathbf{a}_\mathrm{p}) \triangleq \|\mathbf{X}_\mathrm{o} - \bar{\mathbf{X}}\| = \sqrt{(X_\mathrm{o} - \bar{X})^2 + (Y_\mathrm{o} - \bar{Y})^2 + (Z_\mathrm{o} - \bar{Z})^2} = 0 \, .$$

XHG matrix (4.25):

$$\frac{\partial \mathbf{x}}{\partial u} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} , \qquad \mathbf{H} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} , \qquad \mathbf{G} = \varnothing \, .$$

Location parameter value u'_i of the minimum distance point $\mathbf{x}'_i$ from the given point $\mathbf{x}_i = \mathbf{R}_{\omega,\varphi}(\mathbf{X}_i - \mathbf{X}_\mathrm{o})$:

$$u'_i = z_i \, .$$

B.2.3 Plane

A plane can be described as an *implicit* surface.

Standard feature equation (1.1):

$$f(\mathbf{x}) \triangleq z = 0 \, .$$

Parameters ($p = l + n + s = 0 + 3 + 2 = 5, \ q = 1$):

$$\mathbf{a}_\mathrm{g} = \varnothing \, , \qquad \mathbf{a}_\mathrm{p} = (X_\mathrm{o}, Y_\mathrm{o}, Z_\mathrm{o})^\mathrm{T} \, , \qquad \mathbf{a}_\mathrm{r} = (\omega, \varphi)^\mathrm{T} \, .$$

Parameter constraint (2.33):

$$f_\mathrm{c}(\mathbf{a}_\mathrm{p}) \triangleq \|\mathbf{X}_\mathrm{o} - \bar{\mathbf{X}}\| = \sqrt{(X_\mathrm{o} - \bar{X})^2 + (Y_\mathrm{o} - \bar{Y})^2 + (Z_\mathrm{o} - \bar{Z})^2} = 0 \, .$$

FHG matrix (3.21):

$$\nabla f = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} , \qquad \mathbf{H} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} , \qquad \mathbf{G} = \varnothing \, .$$

Minimum distance point $\mathbf{x}'_i$ from the given point $\mathbf{x}_i = \mathbf{R}_{\omega,\varphi}(\mathbf{X}_i - \mathbf{X}_\mathrm{o})$:

$$\mathbf{x}'_i = \begin{pmatrix} x_i \\ y_i \\ 0 \end{pmatrix} .$$

B.2.4 2-D Circle

A 2-D circle can be described as an *implicit* curve in xy-plane.

Standard feature equation (1.1):

$$f(r, \mathbf{x}) \triangleq \frac{1}{2}(x^2 + y^2 - r^2) = 0 \, .$$

Parameters ($p = l + n + s = 1 + 2 + 0 = 3, \ q = 0$):

$$a_\mathrm{g} = r \, , \qquad \mathbf{a}_\mathrm{p} = (X_\mathrm{o}, Y_\mathrm{o})^\mathrm{T} \, , \qquad \mathbf{a}_\mathrm{r} = \varnothing \, .$$

FHG matrix (3.21):

$$\nabla f = \begin{pmatrix} x \\ y \end{pmatrix} , \qquad \mathbf{H} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} , \qquad \mathbf{G} = \begin{pmatrix} -r \\ 0 \\ 0 \end{pmatrix} .$$

Minimum distance point $\mathbf{x}'_i$ from the given point $\mathbf{x}_i = \mathbf{X}_i - \mathbf{X}_\mathrm{o}$:

$$\mathbf{x}'_i = \frac{r}{\|\mathbf{x}_i\|}\mathbf{x}_i = \frac{r}{\sqrt{x_i^2 + y_i^2}} \begin{pmatrix} x_i \\ y_i \end{pmatrix} .$$

B.2.5 3-D Circle

A 3-D circle can be described as a *parametric* 3-D curve.

Standard feature description (1.2):

$$\mathbf{x}(r,u) \triangleq \begin{pmatrix} r\cos u \\ r\sin u \\ 0 \end{pmatrix} \qquad \text{with} \qquad -\pi < u \le \pi \,.$$

Parameters ($p = l + n + s = 1 + 3 + 2 = 6,\ q = 0$):

$$a_{\mathrm{g}} = r\,, \qquad \mathbf{a}_{\mathrm{p}} = (X_{\mathrm{o}}, Y_{\mathrm{o}}, Z_{\mathrm{o}})^{\mathrm{T}}\,, \qquad \mathbf{a}_{\mathrm{r}} = (\omega, \varphi)^{\mathrm{T}}\,.$$

XHG matrix (4.25):

$$\frac{\partial \mathbf{x}}{\partial u} = \begin{pmatrix} -r\sin u \\ r\cos u \\ 0 \end{pmatrix}, \qquad \mathbf{H} = \begin{pmatrix} -r\cos u \\ -r\sin u \\ 0 \end{pmatrix},$$

$$\mathbf{G}_0 = \begin{pmatrix} \cos u \\ \sin u \\ 0 \end{pmatrix}, \qquad \mathbf{G}_1 = \begin{pmatrix} -\sin u \\ \cos u \\ 0 \end{pmatrix}.$$

Location parameter value u_i' of the minimum distance point $\mathbf{x}_i'$ from the given point $\mathbf{x}_i = \mathbf{R}_{\omega,\varphi}(\mathbf{X}_i - \mathbf{X}_{\mathrm{o}})$:

$$u_i' = \tan^{-1}\frac{y_i}{x_i}\,.$$

B.2.6 Sphere

A sphere can be described as an *implicit* surface.

Standard feature equation (1.1):

$$f(r,\mathbf{x}) \triangleq \frac{1}{2}(x^2 + y^2 + z^2 - r^2) = 0\,.$$

Parameters ($p = l + n + s = 1 + 3 + 0 = 4,\ q = 0$):

$$a_{\mathrm{g}} = r\,, \qquad \mathbf{a}_{\mathrm{p}} = (X_{\mathrm{o}}, Y_{\mathrm{o}}, Z_{\mathrm{o}})^{\mathrm{T}}\,, \qquad \mathbf{a}_{\mathrm{r}} = \varnothing\,.$$

FHG matrix (3.21):

$$\nabla f = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \qquad \mathbf{H} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{G} = \begin{pmatrix} -r \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Minimum distance point $\mathbf{x}_i'$ from the given point $\mathbf{x}_i = \mathbf{X}_i - \mathbf{X}_{\mathrm{o}}$:

$$\mathbf{x}_i' = \frac{r}{\|\mathbf{x}_i\|}\mathbf{x}_i = \frac{r}{\sqrt{x_i^2 + y_i^2 + z_i^2}}\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}.$$

B.2.7 Cylinder

A cylinder can be described as an *implicit* surface.

Standard feature equation (1.1):

$$f(r, \mathbf{x}) \triangleq \frac{1}{2}(x^2 + y^2 - r^2) = 0 .$$

Parameters ($p = l + n + s = 1 + 3 + 2 = 6,\ q = 1$):

$$a_\mathrm{g} = r , \qquad \mathbf{a}_\mathrm{p} = (X_\mathrm{o}, Y_\mathrm{o}, Z_\mathrm{o})^\mathrm{T} , \qquad \mathbf{a}_\mathrm{r} = (\omega, \varphi)^\mathrm{T} .$$

Parameter constraint (2.33):

$$\begin{aligned} f_\mathrm{c}(\mathbf{a}_\mathrm{p}, \mathbf{a}_\mathrm{r}) &\triangleq (\mathbf{X}_\mathrm{o} - \bar{\mathbf{X}})^\mathrm{T} \mathbf{r}_\mathrm{z}(\omega, \varphi) \\ &= (X_\mathrm{o} - \bar{X}) S_\varphi - (Y_\mathrm{o} - \bar{Y}) S_\omega C_\varphi + (Z_\mathrm{o} - \bar{Z}) C_\omega C_\varphi = 0 . \end{aligned}$$

FHG matrix (3.21):

$$\nabla f = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} , \qquad \mathbf{H} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} , \qquad \mathbf{G} = \begin{pmatrix} -r \\ 0 \\ 0 \\ 0 \end{pmatrix} .$$

Minimum distance point $\mathbf{x}'_i$ from the given point $\mathbf{x}_i = \mathbf{R}_{\omega,\varphi}(\mathbf{X}_i - \mathbf{X}_\mathrm{o})$:

$$\mathbf{x}'_i = \begin{pmatrix} r \dfrac{x_i}{\sqrt{x_i^2 + y_i^2}} \\ r \dfrac{y_i}{\sqrt{x_i^2 + y_i^2}} \\ z_i \end{pmatrix} .$$

B.2.8 Cone

A cone can be described as an *implicit* surface.

Standard feature equation (1.1):

$$f(\psi, r, \mathbf{x}) \triangleq \frac{1}{2} \left[x^2 + y^2 - \left(r - z \tan \frac{\psi}{2} \right)^2 \right] = 0 \quad \text{with} \quad -\pi < \psi < \pi .$$

Parameters ($p = l + n + s = 2 + 3 + 2 = 7,\ q = 1$):

$$\mathbf{a}_\mathrm{g} = (\psi, r)^\mathrm{T} , \qquad \mathbf{a}_\mathrm{p} = (X_\mathrm{o}, Y_\mathrm{o}, Z_\mathrm{o})^\mathrm{T} , \qquad \mathbf{a}_\mathrm{r} = (\omega, \varphi)^\mathrm{T} .$$

Parameter constraint (2.33):

$$\begin{aligned} f_\mathrm{c}(\mathbf{a}_\mathrm{p}, \mathbf{a}_\mathrm{r}) &\triangleq (\mathbf{X}_\mathrm{o} - \bar{\mathbf{X}})^\mathrm{T} \mathbf{r}_\mathrm{z}(\omega, \varphi) \\ &= (X_\mathrm{o} - \bar{X}) S_\varphi - (Y_\mathrm{o} - \bar{Y}) S_\omega C_\varphi + (Z_\mathrm{o} - \bar{Z}) C_\omega C_\varphi = 0 . \end{aligned}$$

FHG matrix (3.21):

$$\nabla f = \begin{pmatrix} x \\ y \\ \left(r - z\tan\frac{\psi}{2}\right)\tan\frac{\psi}{2} \end{pmatrix}, \qquad \mathbf{H} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -\tan^2\frac{\psi}{2} \end{pmatrix},$$

$$\mathbf{G} = \begin{pmatrix} \frac{z}{2}\left(r - z\tan\frac{\psi}{2}\right)\sec^2\frac{\psi}{2} & -r + z\tan\frac{\psi}{2} \\ 0 & 0 \\ 0 & 0 \\ \left(\frac{r}{2} - z\tan\frac{\psi}{2}\right)\sec^2\frac{\psi}{2} & \tan\frac{\psi}{2} \end{pmatrix}.$$

Minimum distance point $\mathbf{x}_i'$ from the given point $\mathbf{x}_i = \mathbf{R}_{\omega,\varphi}(\mathbf{X}_i - \mathbf{X}_o)$:

$$\mathbf{x}_i' = \begin{cases} \begin{pmatrix} r\frac{x_i}{\sqrt{x_i^2 + y_i^2}} \\ r\frac{y_i}{\sqrt{x_i^2 + y_i^2}} \\ z_i \end{pmatrix} & : \ \psi = 0, \ \text{i.e., a cylinder} \\ \begin{pmatrix} 0 \\ 0 \\ r\cot\frac{\psi}{2} \end{pmatrix} \ (= \text{vertex}) & : \ \begin{matrix} \psi \neq 0 \quad \text{and} \\ \operatorname{sign}(\psi)\cdot z_i \geq \operatorname{sign}(\psi)\cdot z_{i,\text{bnd}} \end{matrix} \\ \begin{pmatrix} \frac{1}{2}\left|(z_{i,\text{bnd}} - z_i)\sin\psi\right|\frac{x_i}{\sqrt{x_i^2 + y_i^2}} \\ \frac{1}{2}\left|(z_{i,\text{bnd}} - z_i)\sin\psi\right|\frac{y_i}{\sqrt{x_i^2 + y_i^2}} \\ r\cot\frac{\psi}{2} - (z_{i,\text{bnd}} - z_i)\cos^2\frac{\psi}{2} \end{pmatrix} & : \ \begin{matrix} \psi \neq 0 \quad \text{and} \\ \operatorname{sign}(\psi)\cdot z_i < \operatorname{sign}(\psi)\cdot z_{i,\text{bnd}} \end{matrix}, \end{cases}$$

with

$$z_{i,\text{bnd}} = r\cot\frac{\psi}{2} + \sqrt{x_i^2 + y_i^2}\cdot\tan\frac{\psi}{2}.$$

B.2.9 Torus

A torus can be described as an *implicit* surface.

Standard feature equation (1.1):

$$f(r_1, r_2, \mathbf{x}) \triangleq \frac{1}{4}\begin{pmatrix} x^4 + y^4 + z^4 + 2(x^2y^2 + y^2z^2 + z^2x^2) \\ -2(r_2^2 + r_1^2)(x^2 + y^2) \\ +2(r_2^2 - r_1^2)z^2 + (r_2^2 - r_1^2)^2 \end{pmatrix} = 0.$$

Parameters ($p = l + n + s = 2 + 3 + 2 = 7,\ q = 0$):

$$\mathbf{a}_\mathrm{g} = (r_1, r_2)^\mathrm{T}\ , \qquad \mathbf{a}_\mathrm{p} = (X_\mathrm{o}, Y_\mathrm{o}, Z_\mathrm{o})^\mathrm{T}\ , \qquad \mathbf{a}_\mathrm{r} = (\omega, \varphi)^\mathrm{T}\ .$$

FHG matrix (3.21):

$$\nabla f = \begin{pmatrix} \left(D - r_2^2\right) x \\ \left(D - r_2^2\right) y \\ \left(D + r_2^2\right) z \end{pmatrix}\ ,$$

$$\mathbf{H} = \begin{pmatrix} 2x^2 + D - r_2^2 & 2xy & 2xz \\ 2yx & 2y^2 + D - r_2^2 & 2yz \\ 2zx & 2zy & 2z^2 + D + r_2^2 \end{pmatrix}\ ,$$

$$\mathbf{G} = \begin{pmatrix} -r_1\left(D + r_2^2\right) & -r_2\left(x^2 + y^2 - z^2 - r_2^2 + r_1^2\right) \\ -2r_1 x & -2r_2 x \\ -2r_1 y & -2r_2 y \\ -2r_1 z & 2r_2 z \end{pmatrix}\ ,$$

with

$$D = x^2 + y^2 + z^2 - r_1^2\ .$$

Minimum distance point $\mathbf{x}_i'$ from the given point $\mathbf{x}_i = \mathbf{R}_{\omega,\varphi}(\mathbf{X}_i - \mathbf{X}_\mathrm{o})$:

$$\mathbf{x}_i' = \mathbf{x}_{i,\mathrm{circle}}' + \frac{r_1}{\left\|\mathbf{x}_i - \mathbf{x}_{i,\mathrm{circle}}'\right\|}\left(\mathbf{x}_i - \mathbf{x}_{i,\mathrm{circle}}'\right)\ ,$$

with

$$\mathbf{x}_{i,\mathrm{circle}}' = \frac{r_2}{\sqrt{x_i^2 + y_i^2}} \begin{pmatrix} x_i \\ y_i \\ 0 \end{pmatrix}\ .$$

C. FHG Matrix of Superellipse and Superellipsoid

Except for a superellipse and superellipsoid, the derivation of the FHG matrix (3.21) is straightforward with the implicit curves and surfaces shown in this work. Because the power function and logarithm function may cause a data overflow and domain error, careful coding of the program source for the FHG matrix is essential for superellipse and superellipsoid fitting.

C.1 Superellipse

To give a compact description and efficient source coding of the FHG matrix, the following superellipse function is used:

$$f(a,b,\varepsilon,\mathbf{x}) \triangleq \frac{1}{2}\left(\left(\frac{|x|}{a}\right)^{2/\varepsilon} + \left(\frac{|y|}{b}\right)^{2/\varepsilon} - 1\right) .$$

The FHG matrix of this superellipse function is

$$\nabla f = \frac{1}{\varepsilon}\begin{pmatrix} A/x \\ B/y \end{pmatrix} , \qquad \mathbf{H} = \frac{2-\varepsilon}{\varepsilon^2}\begin{pmatrix} A/x^2 & 0 \\ 0 & B/y^2 \end{pmatrix} ,$$

$$\mathbf{G} = \frac{1}{\varepsilon}\begin{pmatrix} -A/a & -B/b & -(A\ln A + B\ln B)/2 \\ -2A/\varepsilon a x & 0 & -A(1+\ln A)/\varepsilon x \\ 0 & -2B/\varepsilon b y & -B(1+\ln B)/\varepsilon y \end{pmatrix} ,$$

with

$$A = (|x|/a)^{2/\varepsilon} \qquad \text{and} \qquad B = (|y|/b)^{2/\varepsilon} .$$

From the fact $\lim_{x\to 0} x\ln x = 0$, the occurrence of the domain error ln(0) in run-time program is prevented by treating [0 ln(0)] = 0 as a group.

C.2 Superellipsoid

For superellipsoid, the following function is used:

$$f(a,b,c,\varepsilon_1,\varepsilon_2,\mathbf{x}) \triangleq \frac{1}{2}\left(\left(\left(\frac{|x|}{a}\right)^{2/\varepsilon_1}+\left(\frac{|y|}{b}\right)^{2/\varepsilon_1}\right)^{\varepsilon_1/\varepsilon_2}+\left(\frac{|z|}{c}\right)^{2/\varepsilon_2}-1\right).$$

In order to reduce the danger of a data overflow by the power function, some intermediate variables canceling similarly sized values are introduced as below:

$$d=(|x|/a)^{2/\varepsilon_1}+(|y|/b)^{2/\varepsilon_1}\,, \quad \text{and}$$
$$A=(|x|/a)^{2/\varepsilon_1}/d\,,\quad B=(|y|/b)^{2/\varepsilon_1}/d\,,\quad C=(|z|/c)^{2/\varepsilon_2}\,,\quad D=d^{\varepsilon_1/\varepsilon_2}\,.$$

Then, the FHG matrix of the above superellipsoid function will be

$$\nabla f=\frac{1}{\varepsilon_2}\begin{pmatrix}AD/x\\BD/y\\C/z\end{pmatrix}, \qquad \begin{aligned} H_{xx}&=\frac{AD}{\varepsilon_1\varepsilon_2x^2}\left(2A\frac{\varepsilon_1-\varepsilon_2}{\varepsilon_2}+2-\varepsilon_1\right),\\ H_{yy}&=\frac{BD}{\varepsilon_1\varepsilon_2y^2}\left(2B\frac{\varepsilon_1-\varepsilon_2}{\varepsilon_2}+2-\varepsilon_1\right),\\ H_{zz}&=\frac{C}{\varepsilon_2^2z^2}(2-\varepsilon_2)\,,\\ H_{xy}=H_{yx}&=\frac{2ABD}{\varepsilon_1\varepsilon_2xy}\left(\frac{\varepsilon_1-\varepsilon_2}{\varepsilon_2}\right),\\ H_{yz}=H_{zy}&=H_{zx}=H_{xz}=0\,,\end{aligned}$$

$$\left(\frac{\partial f}{\partial \mathbf{a}_g}\right)^{\mathrm{T}}=\frac{-1}{\varepsilon_2}\begin{pmatrix}AD/a\\BD/b\\C/c\\D(A\ln A+B\ln B)/2\\(C\ln C+D\ln D)/2\end{pmatrix},$$

$$\left(\frac{\partial}{\partial \mathbf{a}_g}\frac{\partial f}{\partial x}\right)^{\mathrm{T}}=\frac{-AD}{\varepsilon_1\varepsilon_2x}\begin{pmatrix}\frac{2}{a}\left(A\frac{\varepsilon_1-\varepsilon_2}{\varepsilon_2}+1\right)\\ \frac{2}{b}\left(B\frac{\varepsilon_1-\varepsilon_2}{\varepsilon_2}\right)\\0\\(A\ln A+B\ln B)\left(\frac{\varepsilon_1-\varepsilon_2}{\varepsilon_2}\right)+\ln A\\ \frac{\varepsilon_1}{\varepsilon_2}(1+\ln D)\end{pmatrix},$$

$$\left(\frac{\partial}{\partial \mathbf{a}_\mathrm{g}} \frac{\partial f}{\partial y}\right)^\mathrm{T} = \frac{-BD}{\varepsilon_1 \varepsilon_2 y} \begin{pmatrix} \frac{2}{a}\left(A\frac{\varepsilon_1 - \varepsilon_2}{\varepsilon_2}\right) \\ \frac{2}{b}\left(B\frac{\varepsilon_1 - \varepsilon_2}{\varepsilon_2} + 1\right) \\ 0 \\ (A \ln A + B \ln B)\left(\frac{\varepsilon_1 - \varepsilon_2}{\varepsilon_2}\right) + \ln B \\ \frac{\varepsilon_1}{\varepsilon_2}(1 + \ln D) \end{pmatrix},$$

$$\left(\frac{\partial}{\partial \mathbf{a}_\mathrm{g}} \frac{\partial f}{\partial z}\right)^\mathrm{T} = \frac{-C}{\varepsilon_2^2 z} \begin{pmatrix} 0 \\ 0 \\ 2/c \\ 0 \\ 1 + \ln C \end{pmatrix}.$$

This page intentionally left blank