DISSERTATION

Variational Motion Design in the Presence of Obstacles

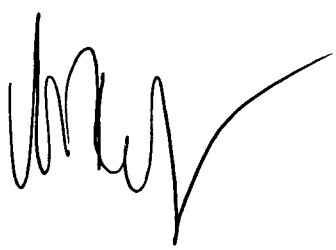
ausgeführt zum Zwecke der Erlangung des akademischen
Grades eines Doktors der technischen Wissenschaften
unter der Leitung von

o. Univ. Prof. Dr. Helmut Pottmann
Institutsnummer E104
Institut für Diskrete Mathematik und Geometrie

eingereicht an der Technischen Universität Wien
Fakultät für Mathematik und Geoinformation
von

Mag. Michael Hofer
Matrikelnummer 9314003
Bachgasse 23/15
A-1160 Wien

Wien, im November 2004



To Manuela

2000 AMS Mathematics Subject Classification: 49M07, 65D05, 65D07, 65D17, 65K10, 68U07, 70B10.

Abstract. In this thesis we study the design of energy-minimizing rigid body motions that interpolate given input positions and avoid given fixed obstacles. Our motion design problem is a curve interpolation problem in an appropriate manifold which in the unconstrained case is the group of Euclidean congruence transformations. We introduce a new metric for motion design where the mass distribution of the moving body enters the calculation, use a variational approach, derive interesting theoretic results concerning motion design and energy minimizing splines in manifolds, employ geometric optimization algorithms for the numerical solution of the posed problems and include the avoidance of obstacles in the design process via the use of appropriate barrier manifolds.

Keywords. Geometric modeling, geometric optimization, motion planning, obstacle avoidance, splines in manifolds, variational curve and motion design.

Reviewers. Prof. Dr. Helmut Pottmann (Vienna University of Technology)
Prof. Dr. Manfred Husty (Leopold Franzens Univ. Innsbruck)

Typesetting by the author using LaTeX.

Kurzfassung

Das Design von Bewegungen eines starren Körpers wird in verschiedenen wissenschaftlichen Disziplinen untersucht. Viele Beiträge zur Bewegungsplanung stammen aus der Robotik, der algorithmischen Geometrie und dem computerunterstützten geometrischen Entwerfen (CAGD). Die Planung von Bewegungen bei gegebenen Hindernissen war bis dato vor allem ein Thema in der Robotik und in der algorithmischen Geometrie. Dabei wurden meist ebene, stückweise lineare Bewegungen verwendet. Algorithmen zur Berechnung von energie-minimierenden Bewegungen mit Hilfe von Quaternionen — ohne Hindernisse zu berücksichtigen — wurden in der Computergrafik untersucht. In der vorliegenden Dissertation

- führen wir eine neue Metrik zum Design von Bewegungen eines starren Körpers ein, welche die Massenverteilung desselbigen berücksichtigt,
- studieren wir Bewegungsplanung als ein Kurvendesign-Problem auf der Fläche der Euklidischen Kongruenztransformationen, welche wir in den Raum der affinen Transformationen einbetten,
- leiten wir interessante theoretische Ergebnisse betreffend Bewegungsplanung und energie-minimierende Kurven auf Flächen her,
- formulieren wir die Bewegungsplanung als ein Variationsproblem und minimieren aus der geometrischen Modellierung bekannte Energiefunktionale auf Flächen,
- präsentieren wir einen sehr universell einsetzbaren geometrischen Optimierungsalgorithmus für die numerische Lösung der Aufgabenstellungen und diskutieren diesen für die speziellen Fälle der Bewegungsplanung und der Bewegungsplanung bei gegebenen Hindernissen,
- berücksichtigen wir die Vermeidung von Hindernissen bei der Planung von energie-minimierenden Bewegungen eines starren Körpers.

Wir studieren das folgenden Problem der Bewegungsplanung: Gegeben sind ein starrer Köper in einer Anzahl von Eingabepositionen und eine Anzahl von Hindernissen, wobei die Eingabepositionen nicht mit den Hindernissen kollidieren. Das Ziel ist die Berechnung einer energie-minimierenden glatten Bewegung des starren Körpers, welche die gegebenen Positionen interpoliert und die vorhandenen Hindernisse vermeidet. Im Gegensatz zu existierenden Beiträgen zum

Bewegungsdesign arbeiten wir nicht mit der Einheitskugel des vierdimensionalen Raumes zur Darstellung des Drehanteils der Bewegung mittels Quaternionen, sondern wir planen die gesamte Bewegung in der Gruppe der euklidischen Kongruenztransformationen eingebettet in die affine Gruppe.

In Kapitel 2 führen wir den kinematischen Bildraum und die verwendete Metrik ein. Wir formulieren die orthogonale Projektion in dieser Metrik und zeigen den Zusammenhang dieser zu einer bekannten Computer Vision Aufgabenstellung, nämlich der Registrierung mit bekannten Korrespondenzen. Wir diskutieren zwei der view bekannten expliziten Lösungsmethoden.

In Kapitel 3 präsentieren wir eine Methode, mit Hilfe derer sich ein beliebiger Algorithmus zum Design einer Kurve auf das Design einer Bewegung übertragen lässt. Wir beweisen, dass sich dabei die Glattheit des verwendeten Kurvenschemas auf die Glattheit der berechneten Bewegung überträgt. Der Algorithmus ist einfach zu implementieren und besonders schnell, wenn das verwendete Kurvenschema linear ist, was für die meisten bekannten Algorithmen zum Design von Kurven gilt. Energie-minimierende Eigenschaften übertragen sich nicht vom Kurvenschema auf die Bewegung. Trotzdem ist der Algorithmus hervorragend geeignet, um schnell eine gute Startlage für die Planung von energie-minimierenden Bewegungen (in Kapitel 5) zu berechnen.

Kapitel 4 dient dem Studium von energie-minimierenden Kurven auf gekrümmten Flächen. Wir erweitern die Definition von kubischen Splines und Splines in Tension auf gekrümmte Flächen und leiten eine geometrische Kennzeichnung her. Für die numerische Lösung präsentieren wir einen geometrischen Optimierungsalgorithmus, welcher iterativ die Energie von Kurven auf Flächen beliebiger Dimension und Kodimension minimiert. Diser Algorithmus arbeitet mit einer Diskretisierung der Kurve und einer Diskretisierung des verwendeten Energiefunktionals. Der sehr allgemeine Ansatz erlaubt die Berechnung von energie-minimierenden Bewegungen in Kapitel 5 und die Planung von energie-minimierenden Bewegungen bei gegebenen Hindernissen in Kapitel 6.

In Kapitel 5 wenden wir die Ergebnisse aus Kapitel 4 auf die Bewegungsplanung an. Zuerst übertragen wir die theoretischen Ergebnisse und dann diskutieren wir einen numerischen Algorithmus zum Design von Bewegungen, welche die Summe der Energien von Punktbahnen minimieren.

In Kapitel 6 diskutieren wir zuerst das Design von energie-minimierenden Kurven bei gegebenen Hindernissen. Kombiniert mit den Ergebnissen der vorigen Kapitel führt dies zu einem ersten konservativen Algorithmus zur Berechnung energie-minimierender Bewegungen bei gegebenen Hindernissen. Dazu wird der starre Körper durch eine ihn umschliessende Kugel ersetzt und die Vermeidung der Hindernisse erreicht, indem die Schwerpunktsbahn so berechnet wird, dass sie von den Hindernissen einen genügend großen Abstand hat. Ein zweiter Algorithmus berücksichtigt die Form des starren Körpers viel genauer und nützt alle für die Berechnung vorhandenen Freiheitsgrade. Wir verwenden wieder den geometrischen Optimierungsalgorithmus aus Kapitel 4 mit einer geeigneten Mannigfaltigkeit.

In Kapitel 7 fassen wir die im Rahmen dieser Dissertation gewonnenen Ergebnisse zusammen und geben einen Ausblick auf zukünftige Forschungsziele.

Acknowledgements

First and foremost I have to express my gratitude to my supervisor Professor Helmut Pottmann. He is a really excellent and visionary researcher in applied geometry and related fields with an extremely motivating, very supportive and overall enthusiastic nature. I acknowledge the uncountable impulses he gave me for the research performed during my Ph.D. studies and for continuously supporting my work in the best possible way. It has been a great experience to mature in such a stimulating environment as a member of his research group 'Geometric Modeling and Industrial Geometry' at TU Vienna. I am looking forward to our cooperation in the future.

My sincerest thanks also belong to Johannes Wallner who shared his enormous mathematical and geometric knowledge with me. I have really benefited from his fast thinking, the clarity of his explanations and the many fruitful discussions I had with him. Furthermore he supported me whenever I needed help with either LaTeX, Linux or Postscript.

I would also like to thank Andreas Asperl, Stefan Leopoldseder, Martin Peternell, Tibor Steiner and my other colleagues. I am grateful to Professor Bahram Ravani for supporting my research at The University of California at Davis from August till November 2002. Special thanks to Professor Herbert Edelsbrunner at Duke University, Professor Leonidas Guibas at Stanford University, Professor Bert Jüttler at Linz University, Professor Lyle Noakes at The University of Western Australia, and Professor Peter Schröder at Caltech, whose research labs I could visit during my Ph.D. studies and where I presented and discussed my research in stimulating environments.

I gratefully acknowledge the financial support by the Austrian Science Fund (FWF) under grant P16002-N05 *Geometric Optimization with Moving and Deformable Objects* that allowed me to perform research from December 2002 on.

Thanks go to my family and friends, especially my parents Hildegard and Franz Hofer, my parents in law Elli and Ewald Haingartner, my brother Markus, my brother Klemens, his wife Heather and their recently born baby son Lukas and my grandmothers Hildegard Steiner and Josefa Hofer. This dissertation is also dedicated to my grandfather Peter Steiner who always supported his grandchildren in the best possible way.

What would life be without you, my beloved wife Manuela? My sincerest loving 'thank you' belongs to you sweetheart for your tender never ending embracing love, your sunny-happy-joyful nature, your kisses, hugs and laughs and all the wonderful love you give me liberally every second of my life. Thanks for being my partner for already half of the life I spend on this great planet.

3D Model Sources

The teapot model (Figs. 3.12, 5.6, 5.11, 6.5) has been created by Martin Newell. The golf club model (Figs. 5.1, 5.2, 5.7, 5.9, 5.10, 6.8, 6.11, 6.12), and the Isis model (Figs. 6.6, 6.7) are courtesy of Cyberware. The bunny model (Figs. 3.4, 3.5, 3.6, 6.5) is courtesy of the Stanford Computer Graphics Laboratory.

All other models have been created by the author. Some of them (Figs. 2.3, 4.3, 4.10, 4.11, 5.4, 6.5, 6.6, 6.7) were obtained via 3D scanning of real objects with a Minolta VI-900 3D laserscanner. For that I acknowledge the support of the innovative project '3D Technology' at Vienna University of Technology. Special thanks go to my brother Markus whose face appears in Fig. 4.10 as a triangle mesh.

Contents

List of Figures

Chapter 1

Introduction

Motion design is a well-studied problem in various areas of scientific research including Robotics, Computational Geometry, and Computer Aided Geometric Design. Motion design in the presence of obstacles has mostly been studied in the areas of Robotics and Computational Geometry. Variational motion design has received some attention for example in the Computer Graphics community. In this PhD thesis we

- introduce a new metric for motion design where the mass distribution of the moving body enters the calculation via a set of sample points,
- derive interesting theoretic results concerning motion design and energy-minimizing splines in manifolds,
- use a variational approach to motion design and minimize the counterparts on surfaces of well-known energy functionals of geometric modeling,
- present a very general geometric optimization algorithm for the computation of energy-minimizing spline curves in manifolds, and show how to employ it for the solution of the motion design problems we consider,
- include the avoidance of obstacles in the design of energy-minimizing rigid body motions via appropriate barrier manifolds.

The motion design problem we study is the following: given input positions of a moving body and obstacles to be avoided, compute a smooth energy-minimizing rigid body motion which interpolates or approximates the given key or control positions and avoids the given obstacles. Unlike most contributions to motion design, we do not use the quaternion unit sphere as a model of the special orthogonal group, but consider the group of Euclidean congruence transformations as a manifold embedded in the affine group. Our motion design problem is a curve interpolation or approximation problem in this manifold.

In Chapter 2 we explain how the group of Euclidean motions is embedded in the affine group and introduce a new metric in the kinematic image space. Furthermore, we study orthogonal projection in this metric in the kinematic image space and show its relation to a well-known problem in Computer Vision, namely registration with known correspondences.

In Chapter 3 we show how to transfer *any* curve design algorithm to motion design and prove that the smoothness of the generated rigid body motion is the same as the smoothness of the employed curve scheme. The presented motion design algorithm — coined the CMD algorithm — is simple to implement and especially fast if a linear curve design algorithm is used (this is not a drawback since most known curve schemes are linear anyway). However, the CMD algorithm does not transfer energy-minimizing properties from the curve scheme to the computed motion. Nevertheless, it will provide us with a good initial motion for variational motion design discussed in Chapter 5.

Variational interpolation in curved geometries is introduced in Chapter 4. We extend the definition of the familiar cubic spline curves and splines in tension to their counterparts in manifolds and derive interesting theoretic results. For the numerical solution we propose a geometric optimization algorithm, which minimizes the chosen energy of curves on surfaces of arbitrary dimension and codimension. This general approach allows not only variational motion design — discussed in Chapter 5 — but also the treatment of obstacles via barrier surfaces presented in Chapter 6.

In Chapter 5 we apply the general results on splines in manifolds to present a characterization of motions which are analogous to known energy-minimizing spline curves, such as cubic splines or splines in tension. We use the numerical algorithm discussed in Chapter 4 for variational motion design, i.e., to compute rigid body motions that minimize a certain energy expressed with help of the energies of trajectories of points on the moving body.

Finally, in Chapter 6 we discuss variational motion design in the presence of obstacles. We begin with variational curve design in the presence of obstacles. This leads to a first algorithm where the moving body avoids the given fixed obstacles via a single enclosing ball. In a second algorithm we capture the shape of the moving body more precisely and use all available degrees of freedom for variational motion design in the presence of obstacles.

We conclude this PhD thesis with an outlook towards future research in Chapter 7.

1.1 Previous Work

Good starting points for the tremendous amount of literature dealing with motion design are the book [Lat01] by Latombe (robot motion planning), the article [Sha97] of Sharir (motion planning in Computational Geometry), and the survey article [JW02] of Jüttler and Wagner (motion design rooted in Kinematics and Computer Aided Geometric Design). In the following overview we only cite literature from the field of motion design (motion planning) that is related to the present work.

Spatial rotations and rigid body motions in Euclidean three-space have been formulated in the literature as time-parameterized loci in non-Euclidean spaces by using (unit) quaternions [FHK+98, GR94b, GR94a, JW96, KN95, Sho85] and (unit) dual quaternions [Jüt94]; see also the references in [Rös98]. Kinematic mappings proved to be a useful tool for these approaches. Invariant

interpolation of rotations has been studied by [PR97], and interpolation methods for rigid body motions by [ZK98, BK00, BK02]. Smooth motions that make use of NURBS techniques were investigated by e.g. [JW96]. To describe rigid body motions it is necessary to use *rational* rather than polynomial representations. However, rational representations are much less suitable for variational design and efficient optimization techniques than polynomial ones. Therefore, [HJK01] investigated the construction of affine spline motions with minimal distortion. Motion design based on the quaternion representation of the spherical component and nonlinear extensions of spline constructions in affine spaces to the sphere (see the references in [Rös98]), are also difficult to deal with for optimization purposes. Algorithms for motion fairing using the quaternion representation have been proposed by [FHK+98]. The design of smooth interpolating motions has also been formulated as a variational problem on the Lie group SE(3) of spatial rigid body displacements [ZK98, BK00]. In these papers mainly the generation of a rigid body motion between two positions with specific boundary conditions has been studied.

While energy-minimizing curves in Euclidean spaces have been studied extensively in the literature, variational design of curves on surfaces has received much less attention. A substantial amount of research focuses on the quaternion 3-sphere because of its well-known relationship with rigid body motions [BCGH92, BC94, JW02, PR97, RB97]. A series of contributions addresses intrinsically cubic splines in manifolds, i.e., the minimizers of the covariant acceleration [CLC01, GK85, NHP89, Noa03]. An *extrinsic* formulation of energy-minimizing curves in parametric surfaces in 3-dimensional Euclidean space is the topic of H. Bohl's thesis [Boh99], who proves the existence of a solution and computes it by quasi-Newton iteration.

Obstacles are well studied for a certain class of motion design problems, cf. [dBvKOS00, Lat01]. There, the main focus is on efficient free motions between two positions, rather than collision free spline-type motions which interpolate or approximate given intermediate positions and minimize an energy functional. One basic approach is the configuration space method [Per83]. As Canny [Can86, Can88] has shown, the fundamental restrictions imposed by moving polyhedral objects and polyhedral obstacles become especially simple in configuration space when using a quaternion representation of motions. The configuration space approach is also used in the fast marching method for efficient collision free motions between two positions according to Kimmel and Sethian (see [Set99]). Obstacle avoidance can also be performed by building a potential field around an obstacle, which assumes high values close to it. Incorporating this into a problem formulation as an optimization problem avoids that the moving object gets too close to the given obstacles (see e.g. [Lat01]). The *singularities* of motions constrained by a contacting surface pair were studied by [PR00]. [HPR03] presented an algorithm for the *design* of a gliding motion for the entire motion cycle rather than around singularities. Near Euclidean near gliding motions were studied by [Wal04b].

Chapter 2

The Kinematic Image Space

Motion design of rigid bodies has received a considerable amount of attention over the last two decades. In the literature, the moving body is usually replaced by a reference coordinate frame and then the motion of that coordinate frame is designed as follows: First compute the path of the barycenter, and then separately compute the linear part of the motion, e.g. using quaternions. The replacement of the moving body by a coordinate frame has the disadvantage, that the designed motion is independent of the shape and mass distribution of the moving body: If two very differently shaped rigid bodies are replaced by the same coordinate frame, the computed motion will be the same.

We propose a new approach that includes the mass distribution of the moving rigid body in the design process. Furthermore, we do not separate the computation of position and orientation of the moving body. For that purpose we use as kinematic image space the 12-dimensional space of affine maps, in which the 6-dimensional manifold M^6 of Euclidean maps is embedded. To apply our concepts we define an appropriate metric in $\mathbb{R}^{12}$. This metric depends on the mass distribution of the rigid body in consideration. It turns out that the orthogonal projection from a point in $\mathbb{R}^{12}$ onto M^6 is related to a well-studied problem in Computer Vision – the registration of two points sets with known correspondences. Let us begin with the embedding of the group of Euclidean displacements in the group of affine displacements.

2.1 The Group of Euclidean Maps Embedded in the Affine Group

Consider a rigid body $\mathcal{B}$ moving in Euclidean three-space E^3. We use Cartesian coordinates and denote points of the moving system Σ^0 by $\mathbf{x}^0, \mathbf{y}^0, \ldots$, and points of the fixed system Σ by $\mathbf{x}, \mathbf{y}$, and so on.

A *one-parameter motion* Σ^0/Σ is a smooth family of Euclidean congruence transformations depending on a parameter u which can be thought of as time. A point $\mathbf{x}^0$ of Σ^0 is, at time u, mapped to the point

$$\mathbf{x}(u) = A(u) \cdot \mathbf{x}^0 + \mathbf{a}_0(u) \tag{2.1}$$

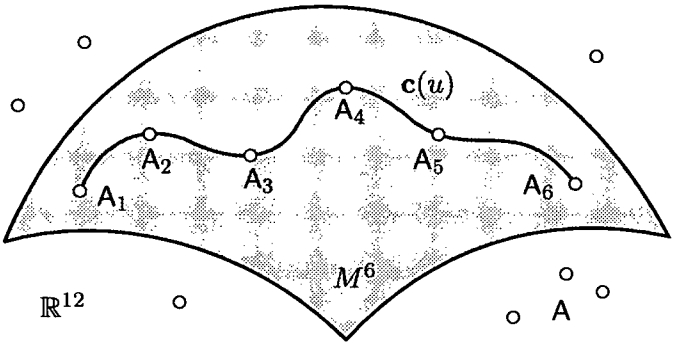


Figure 2.1: The kinematic image space: affine maps A are points in $\mathbb{R}^{12}$; Euclidean maps $\mathsf{A}_1, \ldots, \mathsf{A}_6$ are points of $M^6 \subset \mathbb{R}^{12}$; a rigid body motion $\mathcal{B}(u)$ in $\mathbb{R}^3$ is a curve $\mathbf{c}(u) \subset M^6$.

of Σ, where $A(u) \in SO(3)$ and $\mathbf{a}_0(u) \in \mathbb{R}^3$. If we do not impose any restriction to the matrix A in (2.1), we get, for each u, an *affine map.*

In the following, we will use a kinematic mapping that views affine maps as points in 12-dimensional space $\mathbb{R}^{12}$. For that, consider the affine map $\mathbf{x} = \alpha(\mathbf{x}^0) = \mathbf{a}_0 + A \cdot \mathbf{x}^0$. Let us denote the three column vectors of A as $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$. They describe the images of the basis vectors of Σ^0 in Σ. Of course, we have $\mathbf{x} = \mathbf{a}_0 + x_1^0\mathbf{a}_1 + x_2^0\mathbf{a}_2 + x_3^0\mathbf{a}_3$. Now we associate with the affine map α in $\mathbb{R}^3$ a *point in 12-dimensional space* $\mathbb{R}^{12}$, represented by the vector $\mathsf{A} = (\mathbf{a}_0, \ldots, \mathbf{a}_3)$. The images of Euclidean maps $\alpha \in SE(3)$ form a 6-dimensional manifold $M^6 \subset \mathbb{R}^{12}$. Its 6 equations are given by the orthogonality conditions of the matrix A in (2.1), i.e., $\mathbf{a}_i \cdot \mathbf{a}_j = \delta_{ij}$, $i = 1, 2, 3$.

A one-parameter motion (rigid body motion) $\mathcal{B}(u)$ in $\mathbb{R}^3$ can be seen as a curve $\mathbf{c}(u) \subset M^6$ in $\mathbb{R}^{12}$. If the rigid body motion interpolates given input positions, defined by Euclidean maps α_i, then the curve $\mathbf{c}(u)$ interpolates the points $\mathsf{A}_i \in M^6$ that correspond to α_i, see Fig. 2.1. Whenever we speak of a *position* of the moving body $\mathcal{B}$ this means that we consider all parameters of the affine (Euclidean) map applied to it. In some literature, *position* only describes the translational part of the applied map while the linear part is referred to as *orientation.*

2.2 Metric in the Kinematic Image Space

We introduce a meaningful *metric* in $\mathbb{R}^{12}$ with help of a collection X of points $\mathbf{x}_1^0, \mathbf{x}_2^0, \ldots, \mathbf{x}_K^0$ in the moving system, sampled from the moving body $\mathcal{B}$, see Fig. 2.3. The squared distance between two affine maps α and β is now defined as sum of squared distances of sample point positions after application of α and β, respectively, (see Fig. 2.2)

$$d^2(\alpha, \beta) = \|\mathsf{A} - \mathsf{B}\|^2 := \sum_i [\alpha(\mathbf{x}_i^0) - \beta(\mathbf{x}_i^0)]^2. \tag{2.2}$$

With $\mathsf{A} = (\mathbf{a}_0, \ldots, \mathbf{a}_3)$, $\mathsf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_3)$, $\mathsf{C} := \mathsf{A} - \mathsf{B} = (\mathbf{c}_0, \ldots, \mathbf{c}_3)$, and

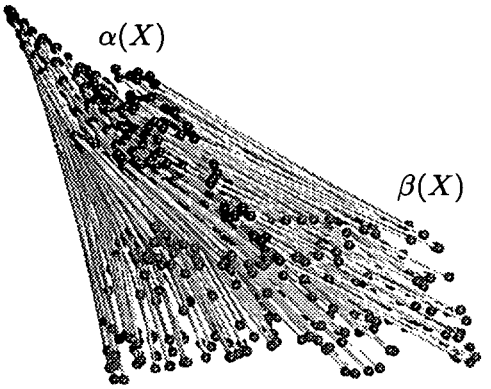


Figure 2.2: Visualization of distance between two affine maps α and β.

$\mathbf{x}_i^0 = (x_{i,1}^0, x_{i,2}^0, x_{i,3}^0)$ the distance (2.2) becomes

$$\|\mathsf{A} - \mathsf{B}\|^2 = \|\mathsf{C}\|^2 = \sum_i [\mathbf{c}_0 + x_{i,1}^0 \mathbf{c}_1 + x_{i,2}^0 \mathbf{c}_2 + x_{i,3}^0 \mathbf{c}_3]^2 = \mathsf{C}^T \cdot M \cdot \mathsf{C}, \quad (2.3)$$

where the symmetric positive definite 12 by 12 matrix M is given as

$$M = \sum_i \begin{pmatrix} I & x_{i,1}^0 I & x_{i,2}^0 I & x_{i,3}^0 I \\ \cdot & (x_{i,1}^0)^2 I & x_{i,1}^0 x_{i,2}^0 I & x_{i,1}^0 x_{i,3}^0 I \\ \cdot & \cdot & (x_{i,2}^0)^2 I & x_{i,2}^0 x_{i,3}^0 I \\ \cdot & \cdot & \cdot & (x_{i,3}^0)^2 I \end{pmatrix}. \quad (2.4)$$

Here I denotes the 3 by 3 unit matrix. Taking a look at the entries of the matrix M reveals the following: First, the upper left 3 by 3 diagonal matrix KI only depends on the number K of sample points. Second, the entries of the 3 by 3 diagonal matrices $\sum_i x_{i,j}^0 I$, $j = 1, 2, 3$ only depend on the barycenter

$$\mathbf{s}_x = \frac{1}{K} \sum_{i=1}^{K} (x_{i,1}^0, x_{i,2}^0, x_{i,3}^0) \quad (2.5)$$

of the sample points. Third, the lower right 9 by 9 submatrix of M only contains entries of the inertia tensor J,

$$J := \sum_i \mathbf{x}_i^0 \cdot \mathbf{x}_i^{0T} = \sum_i \begin{pmatrix} (x_{i,1}^0)^2 & x_{i,1}^0 x_{i,2}^0 & x_{i,1}^0 x_{i,3}^0 \\ \cdot & (x_{i,2}^0)^2 & x_{i,2}^0 x_{i,3}^0 \\ \cdot & \cdot & (x_{i,3}^0)^2 \end{pmatrix}, \quad (2.6)$$

which is itself a symmetric positive definite 3 by 3 matrix. We summarize the derived facts about the metric in the following lemma.

Lemma 1. *The metric (2.2) in the space of affine maps only depends on the number K, on the barycenter $\mathbf{s}_x$, and on the inertia tensor J of the set of sample points $\mathbf{x}_1^0, \ldots, \mathbf{x}_K^0$ of the moving body. $\mathbb{R}^{12}$ equipped with this metric is a Euclidean space E^{12}.*

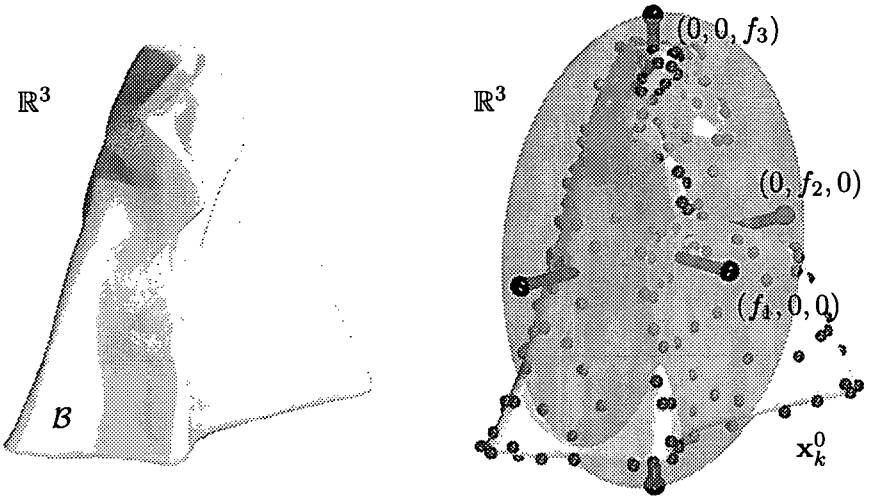


Figure 2.3: (Left) Example moving body $\mathcal{B}$. (Right) Sample points $\mathbf{x}_k^0$, inertia ellipsoid and the principal axis of inertia.

We also see that we need not use unit point masses at a discrete number of sample points. We could instead work with another positive measure on a domain of interest D (the moving body) in Σ^0, e.g. the Lebesgue measure times the characteristic function χ_d of D. Of course, we then replace summation in (2.2) by integration. By a well-known result from mechanics, we can replace the points $\mathbf{x}_1^0, \ldots, \mathbf{x}_K^0$ by the six special points

$$\mathbf{s}_x \pm \sqrt{\frac{\lambda_j}{2}}\mathbf{e}^j, \qquad j = 1, 2, 3, \tag{2.7}$$

without changing the barycenter and the inertia tensor of X. There, $\lambda_1, \lambda_2, \lambda_3$ and $\mathbf{e}^1, \mathbf{e}^2, \mathbf{e}^3$ are the eigenvalues and corresponding unit eigenvectors of the matrix J described in (2.6). Let us choose the barycenter as origin in the moving system and the eigenvectors of J as coordinate axes. Then the six points (2.7) have coordinates $(\pm f_1, 0, 0), (0, \pm f_2, 0), (0, 0, \pm f_3)$ with $f_j := \sqrt{\lambda_j/2}, j = 1, 2, 3$, and the norm in E^{12} becomes

$$\|\mathsf{C}\|^2 = K\mathbf{c}_0^2 + 2\sum_{i=1}^{3} f_i^2 \mathbf{c}_i^2 = \mathsf{C}^T \cdot M \cdot \mathsf{C}, \tag{2.8}$$

with the positive definite 12 by 12 diagonal matrix M,

$$M = \begin{pmatrix} KI & & & \\ & 2f_1^2 I & & \\ & & 2f_2^2 I & \\ & & & 2f_3^2 I \end{pmatrix}. \tag{2.9}$$

Remark 2.1. Analogously we can embed the three dimensional manifold M^3 of orthogonal maps into the group of linear maps A^9. Then the matrix of the corresponding metric is given by the lower right 9 by 9 block of the matrix M defined in (2.9), and thus only depends on the inertia tensor of the point cloud.

2.3 Tangent Spaces of M^6 and Orthogonality to M^6

Later on we will need tangent spaces at points $\mathsf{A} \subset M^6$ and a characterization of orthogonality to these tangent spaces. A tangent vector at an arbitrary point $\mathsf{A} \in E^{12}$ can be interpreted in $\mathbb{R}^3$ as a velocity vector field of an affine motion. In particular, a tangent vector of M^6 belongs to a velocity vector field of a Euclidean motion, which is of the form $\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}$. The coordinate representation of this tangent vector in E^{12}, attached to $\mathsf{A} = (\mathbf{a}_0, \ldots, \mathbf{a}_3) \in M^6$, reads

$$\mathsf{T} = (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{a}_0, \mathbf{c} \times \mathbf{a}_1, \mathbf{c} \times \mathbf{a}_2, \mathbf{c} \times \mathbf{a}_3). \tag{2.10}$$

We would like to express orthogonality of an arbitrary vector $\mathsf{D} = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3) \in E^{12}$ to the tangent space at A. If we align origin and axes of the coordinate system in Σ^0 with center and axes of the inertia ellipsoid of the moving body $\mathcal{B}$, then the inner product is expressed in view of (2.8) as

$$\langle \mathsf{D}, \mathsf{T} \rangle = K\mathbf{d}_0 \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{a}_0) + 2\sum_{i=1}^{3} f_i^2 \mathbf{d}_i \cdot (\mathbf{c} \times \mathbf{a}_i).$$

This equation may also be written as

$$\langle \mathsf{D}, \mathsf{T} \rangle = K\mathbf{d}_0 \cdot \bar{\mathbf{c}} + K\mathbf{c} \cdot (\mathbf{a}_0 \times \mathbf{d}_0) + 2\mathbf{c}\sum_{i=1}^{3} f_i^2 (\mathbf{a}_i \times \mathbf{d}_i). \tag{2.11}$$

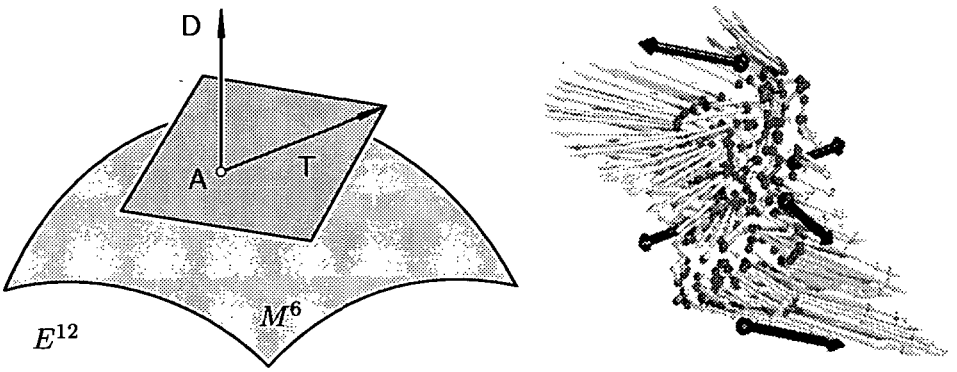


Figure 2.4: (Left) Tangent space of and orthogonality to M^6. (Right) Force system acting on the moving body.

Let us attach to the six point positions $\mathbf{a}_0 \pm f_i\mathbf{a}_i$, $i = 1, 2, 3$, the vectors of the linear vector field determined by $\mathsf{D} \in \mathbb{R}^{12}$. These vectors are $\mathbf{d}_0 \pm f_i\mathbf{d}_i$, and shall be interpreted as forces (for such concepts from statics, see e.g. [PW01], pp. 191–194). The moments of these forces are

$$(\mathbf{a}_0 \pm f_i\mathbf{a}_i) \times (\mathbf{d}_0 \pm f_i\mathbf{d}_i).$$

The sums of force vectors and moment vectors is the screw

$$(\mathbf{s}_d, \bar{\mathbf{s}}_d) = (K\mathbf{d}_0, K\mathbf{a}_0 \times \mathbf{d}_0 + 2\sum_{i=1}^{3} f_i^2 (\mathbf{a}_i \times \mathbf{d}_i)). \tag{2.12}$$

We call the screw resulting from the action of an instantaneous affine motion (linear vector field) on the sample points the *screw or force system* S_d *induced by the linear vector field* D. The inner product (2.11) between T and D is now expressed as

$$\langle \mathsf{D}, \mathsf{T} \rangle = \bar{\mathbf{c}} \cdot \mathbf{s}_d + \mathbf{c} \cdot \bar{\mathbf{s}}_d. \tag{2.13}$$

This is the *virtual work done by the force system* S_d *on the body which moves instantaneously with the velocity field determined by* T. Orthogonality between D and all T requires $(\mathbf{s}_d, \bar{\mathbf{s}}_d) = 0$, i.e., balance of the induced force system S_d. Thus, we have proved the following result.

Theorem 1. *A vector* $\mathsf{D} \in E^{12}$*, attached to a point* $\mathsf{A} \in M^6$*, is orthogonal to* $M^6 \subset E^{12}$ *if and only if the force system* S_d *induced by* D *is in balance.*

2.4 Orthogonal Projection onto M^6

The orthogonal projection of a point $\mathsf{A} \in E^{12}$ onto a point $\mathsf{M} \in M^6$ in the metric defined in Sect. 2.2 can be formulated in $\mathbb{R}^3$ as follows. Given an affine map α (with kinematic image A) we seek a Euclidean map m (with kinematic image M),

$$m : \mathbf{x}_i' = \mathbf{t} + R \cdot \mathbf{x}_i, \tag{2.14}$$

such that the squared distance between m and α,

$$d^2(m, \alpha) = \sum_{i=1}^{K} \|m(\mathbf{x}_i) - \alpha(\mathbf{x}_i)\|^2, \tag{2.15}$$

is minimized in the metric (2.2).

Registration with known correspondences. The orthogonal projection of a point $\mathsf{A} \in E^{12}$ onto M^6 in the metric (2.2) is related to a well-known problem in Computer Vision, coined *registration with known correspondences* or *the absolute orientation problem.* It is a more general problem where one seeks the Euclidean map m that aligns a point cloud $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$ to an arbitrarily deformed copy $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_K\}$ of X such that the sum of squared distances between corresponding points $\mathbf{x}_i$ and $\mathbf{y}_i$, $i = 1, \ldots, K$ is minimized,

$$\sum_{i=1}^{K} \|m(\mathbf{x}_i) - \mathbf{y}_i\|^2 \rightarrow \min. \tag{2.16}$$

We will see that the optimal translation vector $\mathbf{t}$ and the optimal rotation matrix R of the Euclidean map m can be computed separately. The optimal translation is easy to solve as we will see in Sect. 2.4.1. The optimal rotation is a little bit more involved, however four distinct *closed form solutions* have been found by researchers of the Computer Vision community between 1983 and 1991. Some of them were discovered several times independently. Before that, *iterative solutions* can be found in the photogrammetry and psychology literature of the 1950s and 1960s. An iterative solution for the simultaneous

registration of more than two systems has been described by [PLH02, PHYH04]. The closed form solution methods for the optimal rotation R of the Euclidean map minimizing (2.16) are based on

- unit quaternions (UQ), see [FH83, Hor87]
- singular value decomposition (SVD), see [AHB87, Wal02]
- orthonormal matrices, see [SS87, HHN88, Ume91]
- dual quaternions, see [WSV91].

A comparison of these four algorithms with respect to numerical accuracy, stability, and efficiency has been performed by [ELF97]. They conclude that the differences are small, with SVD and UQ being slightly more stable than the other two methods. In this thesis we only discuss the UQ and SVD method and refer the reader to the above cited literature for the other two approaches. We first discuss the more general problem (2.16), and then we present a computationally efficient solution for (2.15).

Remark 2.2. Registration with known correspondences is a problem that has to be solved repeatedly in the Iterative Closest Point (ICP) algorithm, cf. [BM92, CM92, RL01].

2.4.1 Optimal Translation

Let us begin with the derivation of the optimal translation vector $\mathbf{t}$.

Lemma 2. *The Euclidean map m minimizing (2.16) maps the barycenter $\mathbf{s}_x := \frac{1}{K}\sum_{i=1}^{K}\mathbf{x}_i$ of X to the barycenter $\mathbf{s}_y := \frac{1}{K}\sum_{i=1}^{K}\mathbf{y}_i$ of Y.*

Proof. We seek the minimizer of the quadratic function

$$F(\mathbf{t}, R) = \sum_i (\mathbf{x}'_i - \mathbf{y}_i)^2 = \sum_i (\mathbf{t} + R \cdot \mathbf{x}_i - \mathbf{y}_i)^2. \tag{2.17}$$

The necessary conditions for a minimum of (2.17) include the three scalar equations

$$\frac{\partial}{\partial \mathbf{t}}: \quad 2\sum_i (\mathbf{t} + R \cdot \mathbf{x}_i - \mathbf{y}_i) = 0. \tag{2.18}$$

If the Euclidean displacement maps $\mathbf{s}_x$ to $\mathbf{s}_y$, then $\mathbf{s}_y = \mathbf{t} + R \cdot \mathbf{s}_x$, i.e.,

$$\mathbf{t} = \mathbf{s}_y - R \cdot \mathbf{s}_x. \tag{2.19}$$

If we plug (2.19) into (2.18) we get

$$\sum_i (\mathbf{s}_y - R \cdot \mathbf{s}_x + R \cdot \mathbf{x}_i - \mathbf{y}_i) = \sum_i (\mathbf{s}_y - \mathbf{y}_i) + R \cdot \sum_i (\mathbf{x}_i - \mathbf{s}_x) = 0, \tag{2.20}$$

since $\mathbf{s}_x$ and $\mathbf{s}_y$ satisfy

$$\sum_i (\mathbf{s}_y - \mathbf{y}_i) = 0 \quad \text{and} \quad \sum_i (\mathbf{x}_i - \mathbf{s}_x) = 0. \tag{2.21}$$

This proves that the optimal transformation maps the barycenter $\mathbf{s}_x$ of X to the barycenter $\mathbf{s}_y$ of Y. □

2.4.2 Optimal Rotation — Preliminaries

Finding the optimal rotation R using unit quaternions or singular value decomposition involves a first manipulation step which is identical for both cases: To the point cloud X we apply the translation $\mathbf{s}_x \mapsto \mathbf{s}_y$, see Lemma 2. Then we choose $\mathbf{s}_y$ as new origin and denote the such translated point cloud X' of points $\mathbf{x}'_i$ again by X and $\mathbf{x}_i$, respectively. To compute the rotation around the origin O, $\mathbf{x}' = R \cdot \mathbf{x}$ (with an orthogonal matrix R), such that the quadratic objective function F is minimized, we rearrange the terms in the following way:

$$\begin{aligned} F &= \sum_i \|\mathbf{x}'_i - \mathbf{y}_i\|^2 = \sum_i (R \cdot \mathbf{x}_i - \mathbf{y}_i)^2 \\ &= \sum_i (\mathbf{x}_i^T \cdot R^T - \mathbf{y}_i^T) \cdot (R \cdot \mathbf{x}_i - \mathbf{y}_i) \\ &= \sum_i (\mathbf{x}_i^T \cdot R^T \cdot R \cdot \mathbf{x}_i - \mathbf{y}_i^T \cdot R \cdot \mathbf{x}_i - \mathbf{x}_i^T \cdot R^T \cdot \mathbf{y}_i + \mathbf{y}_i^T \cdot \mathbf{y}_i) \\ &= \sum_i (\mathbf{x}_i^T \cdot \mathbf{x}_i + \mathbf{y}_i^T \cdot \mathbf{y}_i - 2\mathbf{y}_i^T \cdot R \cdot \mathbf{x}_i) \rightarrow \min . \end{aligned} \tag{2.22}$$

The first two terms in the sum (2.22) are given by the input points and are thus fixed. Thus, in order to minimize (2.22) we have to find the orthogonal matrix R maximizing

$$\sum_i \mathbf{y}_i^T \cdot R \cdot \mathbf{x}_i \rightarrow \max . \tag{2.23}$$

The difference of the unit quaternion method and the singular value decomposition method is in the maximization of (2.23). We derive these methods in Sect. 2.4.3 and Sect. 2.4.4 respectively.

2.4.3 Optimal Rotation Using Unit Quaternions

Computing the optimal rotation using unit quaternions leads to the solution of a general eigenvalue problem. The most cited article for this solution is [Hor87], while the first solution is due to [FH83] in 1983.

Lemma 3. *The optimal rotation R maximizing (2.23) is computed via the unit eigenvector $\mathbf{q} \in \mathbb{R}^4$ corresponding to the largest eigenvalue of the symmetric 4 by 4 matrix*

$$N := \begin{pmatrix} S_{11}+S_{22}+S_{33} & S_{23}-S_{32} & S_{31}-S_{13} & S_{12}-S_{23} \\ \cdot & S_{11}-S_{22}-S_{33} & S_{12}+S_{21} & S_{31}+S_{13} \\ \cdot & \cdot & -S_{11}+S_{22}-S_{33} & S_{23}+S_{32} \\ \cdot & \cdot & \cdot & -S_{11}-S_{22}+S_{33} \end{pmatrix}, \tag{2.24}$$

where $S_{jk}, j,k = 1,\ldots,3$ are the nine entries of the matrix

$$H := \sum_i \mathbf{x}_i \cdot \mathbf{y}_i^T =: \begin{pmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{pmatrix}. \tag{2.25}$$

From the unit vector $\mathbf{q} = (q_0, q_1, q_2, q_3)^T$ we compute the rotation matrix R via (A.22) of Prop. 1.

Proof. By Prop. 1 of Appendix A we can use a unit quaternion $\mathbf{q}$ to rewrite the rotation $\mathbf{x}_i \mapsto R \cdot \mathbf{x}_i$ to $\mathbf{x}_i \mapsto \mathbf{q} \circ \mathbf{x}_i \circ \overline{\mathbf{q}}$. Furthermore we use properties (A.14), (A.19) and (A.20) of Appendix A to rewrite

$$\begin{aligned} \mathbf{y}_i^T \cdot R \cdot \mathbf{x}_i &= \langle \mathbf{y}_i, R \cdot \mathbf{x}_i \rangle \\ &= \langle \mathbf{y}_i, \mathbf{q} \circ \mathbf{x}_i \circ \overline{\mathbf{q}} \rangle \\ &= \langle \mathbf{y}_i \circ \mathbf{q}, \mathbf{q} \circ \mathbf{x}_i \rangle \\ &= \langle Y_i \cdot \mathbf{q}, \tilde{X}_i \cdot \mathbf{q} \rangle \\ &= \mathbf{q}^T \cdot Y_i^T \cdot \tilde{X}_i \cdot \mathbf{q}. \end{aligned} \tag{2.26}$$

Using (2.26) we rewrite (2.23) to get a quadratic form we have to maximize,

$$\sum_i \mathbf{y}_i^T \cdot R \cdot \mathbf{x}_i = \sum_i \mathbf{q}^T \cdot Y_i^T \cdot \tilde{X}_i \cdot \mathbf{q} = \mathbf{q}^T \cdot N \cdot \mathbf{q} \to \max. \tag{2.27}$$

Note that $\mathbf{x}_i$ and $\mathbf{y}_i$ are vectors and thus embedded into quaternion space as $(0, x_{i,1}, x_{i,2}, x_{i,3})$ and $(0, y_{i,1}, y_{i,2}, y_{i,3})$ respectively. Using (A.19) and (A.20) we see that the symmetric 4 by 4 matrix $\sum_i Y_i^T \cdot \tilde{X}_i$ is exactly the matrix N of (2.24):

$$\begin{aligned} Y^T \cdot \tilde{X} &= \begin{pmatrix} 0 & y_1 & y_2 & y_3 \\ -y_1 & 0 & y_3 & -y_2 \\ -y_2 & -y_3 & 0 & y_1 \\ -y_3 & y_2 & -y_1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & -x_1 & -x_2 & -x_3 \\ x_1 & 0 & x_3 & -x_2 \\ x_2 & -x_3 & 0 & x_1 \\ x_3 & x_2 & -x_1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} x_1y_1 + x_2y_2 + x_3y_3 & x_2y_3 - x_3y_2 & x_3y_1 - x_1y_3 & x_1y_2 - x_2y_1 \\ \cdot & x_1y_1 - x_2y_2 - x_3y_3 & x_1y_2 + x_2y_1 & x_1y_3 + x_3y_1 \\ \cdot & \cdot & -x_1y_1 + x_2y_2 - x_3y_3 & x_2y_3 + x_3y_2 \\ \cdot & \cdot & \cdot & -x_1y_1 - x_2y_2 + x_3y_3 \end{pmatrix}. \end{aligned} \tag{2.28}$$

The quadratic side condition for the maximization of the quadratic form $\mathbf{q}^T \cdot N \cdot \mathbf{q}$ in (2.27) is

$$\|\mathbf{q}\|^2 = \mathbf{q}^T \cdot \mathbf{q} = \mathbf{q}^T \cdot I \cdot \mathbf{q} = 1. \tag{2.29}$$

It is well-known that the minimization of a quadratic form with a quadratic side condition leads to the solution of a general eigenvalue problem. However,

in the present problem the matrix of the quadratic side condition is the 4 by 4 unit matrix I — thus we only have to solve a standard eigenvalue problem $\det(N - \lambda I) = 0$. The largest eigenvector $\mathbf{q} \in \mathbb{R}^4$ of the matrix N, normalized according to (2.29), gives the desired solution. □

2.4.4 Optimal Rotation Using Singular Value Decomposition

Computing the optimal rotation R using a singular value decomposition has been introduced by [AHB87], whose derivation we follow here. Again we want to maximize the quadratic objective function (2.23). For that purpose we rewrite it as

$$\sum_i \mathbf{y}_i^T \cdot R \cdot \mathbf{x}_i = \operatorname{tr}\left(\sum_i R \cdot \mathbf{x}_i \cdot \mathbf{y}_i^T\right) = \operatorname{tr}(R \cdot H), \tag{2.30}$$

where $\operatorname{tr} A := \sum_j a_{jj}$ denotes the *trace* of a square matrix A and $H = \sum_i \mathbf{x}_i \cdot \mathbf{y}_i^T$ is the matrix defined in (2.25).

Lemma 4. *For any positive definite matrix $A \cdot A^T$ and any orthogonal matrix Q,*

$$\operatorname{tr}(A \cdot A^T) \geq \operatorname{tr}(Q \cdot A \cdot A^T). \tag{2.31}$$

Proof. Let $\mathbf{a}_i$ be the i-th column of A. Using the fact that the trace of a product of two square matrices is independent of the order of the multiplication we derive

$$\operatorname{tr}(Q \cdot A \cdot A^T) = \operatorname{tr}(A^T \cdot Q \cdot A) = \sum_i \mathbf{a}_i^T \cdot (Q \cdot \mathbf{a}_i). \tag{2.32}$$

Using the Cauchy-Schwarz inequality and the orthogonality of Q we get

$$\mathbf{a}_i^T \cdot (Q \cdot \mathbf{a}_i) \leq \sqrt{(\mathbf{a}_i^T \cdot \mathbf{a}_i)(\mathbf{a}_i^T \cdot Q^T \cdot Q \cdot \mathbf{a}_i)} = \mathbf{a}_i^T \cdot \mathbf{a}_i. \tag{2.33}$$

Thus, $\operatorname{tr}(Q \cdot A \cdot A^T) \leq \sum_i \mathbf{a}_i^T \cdot \mathbf{a}_i = \operatorname{tr}(A \cdot A^T)$, which completes the proof. □

Lemma 5. *Let H be the matrix defined in (2.25) with the singular value decomposition (SVD)*

$$H = U \cdot \Sigma \cdot V^T. \tag{2.34}$$

Then, among all orthogonal matrices, the matrix

$$R = V \cdot U^T \tag{2.35}$$

maximizes (2.23). If $\det(R) = +1$, then R describes the optimal rotation.

Proof. The 3 by 3 matrices U and V are orthogonal and the 3 by 3 diagonal matrix Σ contains the nonnegative singular values of H. As a product of two orthogonal matrices R is itself orthogonal. Using the orthogonality of U we derive

$$R \cdot H = V \cdot U^T \cdot U \cdot \Sigma \cdot V^T = V \cdot \Sigma \cdot V^T, \tag{2.36}$$

which is a symmetric and positive definite matrix. Therefore, it follows from Lemma 4 that for any 3 by 3 orthogonal matrix Q,

$$\operatorname{tr}(R \cdot H) \geq \operatorname{tr}(Q \cdot R \cdot H). \tag{2.37}$$

Thus R maximizes the objective function (2.23), and if $\det(R) = +1$, then the orthogonal matrix R describes the optimal rotation. □

Remark 2.3. If $\det(R) = -1$, then R is a reflection and not a rotation. Following [AHB87] we can deal with it as follows. Geometrically, this case corresponds to Y being a set of coplanar points. Then there is a unique reflection and a unique rotation, and one of the singular values of the matrix H is zero, say σ_3. As explained in [AHB87], one can change the sign of the third column of the matrix $V = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ to get a matrix $\tilde{V} = [\mathbf{v}_1, \mathbf{v}_2, -\mathbf{v}_3]$. Then $\tilde{R} = \tilde{V} \cdot U^T$ is the desired rotation.

2.4.5 Optimal Rotation in the Affine Case

Now that we have solved the more general problem (2.16) of finding the optimal rigid transformation that aligns a Euclidean copy of a point cloud to an arbitrarily deformed version of it, we return to the special case (2.15), where the deformation is described by a nonsingular affine map.

We have seen that the matrix H defined in (2.25) contains the essential information that is used to compute the best-fit Euclidean transformation. This holds for both, the unit quaternion and the singular value decomposition solution. Later on in our motion design algorithms we will have the case that the point cloud $\mathbf{y}_i$ is an affine copy of $\mathbf{x}_i$, and that we need to compute the matrix H repeatedly. Therefore it is advantageous to know that in the affine case H can be computed efficiently with just 9 multiplications, instead of the $9K$ multiplications involved in the definition (2.25) of H (where K could be quite large).

Lemma 6. *If the point cloud $\mathbf{y}_i$ results from $\mathbf{x}_i$ by applying a linear map with matrix A, $\mathbf{y}_i = A \cdot \mathbf{x}_i$ for $i = 1, \ldots, K$, then the matrix $H = \sum_i \mathbf{x}_i \cdot \mathbf{y}_i^T$ defined in (2.25) is given by*

$$H = J \cdot A^T, \tag{2.38}$$

where $J = \sum_i \mathbf{x}_i \cdot \mathbf{x}_i{}^T$ is the symmetric inertia tensor defined in (2.6).

Proof. Transposing $\mathbf{y}_i = A \cdot \mathbf{x}_i$ gives $\mathbf{y}_i^T = \mathbf{x}_i^T \cdot A^T$ and plugging this into (2.25) results in

$$H = \sum_i \mathbf{x}_i \cdot \mathbf{y}_i^T = \sum_i \mathbf{x}_i \cdot (\mathbf{x}_i^T \cdot A^T) = (\sum_i \mathbf{x}_i \cdot \mathbf{x}_i^T) \cdot A^T = J \cdot A^T. \tag{2.39}$$

□

Since the point cloud $\mathbf{x}_i$ is such that its barycenter is in the origin and the principal axes of inertia are aligned with the coordinate axes, the inertia tensor J is actually a diagonal matrix. In our applications, J only needs to be computed once in a preprocessing step. The affine maps will arise from the motion design algorithm. Since J is a diagonal matrix the matrix $H = J \cdot A^T$ can be computed with only 9 multiplications.

We summarize the main result of Sect. 2.4 in the following algorithm for the orthogonal projection of a point $\mathsf{A} \in E^{12}$ onto a point $\mathsf{M} \in M^6$:

Algorithm 1 (Orthogonal projection onto M^6). *Let* A *be a given point in* E^{12}*, corresponding to a nonsingular affine map in* $\mathbb{R}^3$*. Then the orthogonal projection of* A *onto* M^6 *corresponds to the Euclidean map* m *of (2.14) that minimizes (2.15) in the metric of Sect. 2.2. We compute* m *as follows:*

1. *The optimal translational part* $\mathbf{t}$ *maps barycenter to barycenter.*
2. *For the optimal rotation* R *we first compute the matrix* H *via (2.38), and then we compute* R *from* H *either using the unit quaternion approach of Sect. 2.4.3 or the singular value decomposition approach of Sect. 2.4.4.*

Remark 2.4. If $\det(A) = 0$, then there is no unique best-fit orthogonal matrix. For a derivation of the medial axis of the orthogonal group $O(3)$ and the special orthogonal group $SO(3)$ in the metric of Sect. 2.2 we refer the reader to [Wal02].

Chapter 3

From Curve Design Algorithms to the Design of Rigid Body Motions

In this chapter we present a motion design algorithm that can be considered as a transfer principle from curve design algorithms to the design of rigid body motions. We abbreviate this algorithm, which has been introduced by Hofer, Pottmann and Ravani [HPR02, HPR04], as the CMD algorithm (from Curve design algorithms to Motion Design). At the same time completely independent, a similar method has been published by Belta and Kumar [BK02]. The difference to their method is the derivation of the results and that the CMD algorithm is more general in the sense that it can be used to compute the best-fit Euclidean motion to arbitrarily deformed motions, and not only to affine ones.

The CMD algorithm is used in Chapter 5 to find an initial value for the iterative optimization algorithm used to compute energy-minimizing motions. In the present chapter we prove that the smoothness of the generated motion is the same as the smoothness of the used curve design algorithm, and we show examples of interpolating and approximating rigid body motions using several different curve design algorithms.

The motion design problem Given N positions $\mathcal{B}_i := \mathcal{B}(u_i)$ of a moving body $\mathcal{B} \subset \mathbb{R}^3$ at time instances u_i, compute a smooth rigid body motion $\mathcal{B}(u)$ that interpolates (or approximates) the given N positions $\mathcal{B}(u_i)$, such that chosen sample points of the moving system run on smooth paths.

Outline of the CMD algorithm

1. Use the curve design algorithm to create a *deformed* motion, which is an *affine* motion in the case of a *linear* curve design scheme.

2. Compute the best-fit Euclidean motion to the deformed motion. This is done via registration with known correspondences discussed in Sect. 2.4.

We first present the CMD algorithm in Sect. 3.1 in full generality, and afterwards, in Sect. 3.2 we show how the algorithm simplifies if a linear curve design algorithm is employed.

3.1 The CMD Algorithm

As introduced in Sect. 2.2 we represent the moving body $\mathcal{B}$ by $K > 4$ sample points $\mathbf{x}_1^0, \ldots, \mathbf{x}_K^0$. Given are Euclidean maps $m_i = (R_i, \mathbf{t}_i)$ of N input positions $\mathcal{B}_i := \mathcal{B}(u_i)$ of the moving body $\mathcal{B}$ at time instances u_i.

Homologous points are the different locations of a single sample point as the moving body takes different positions. By applying the Euclidean maps m_i to the sample points $\mathbf{x}_1^0, \ldots, \mathbf{x}_K^0$ we get their locations $\mathbf{x}_k^i$ at the given time instances u_i, which results in K sequences of homologous points.

Step 1: Deformed motion via the curve design algorithm. In the first step of the CMD algorithm we apply to each of the K sequences of homologous points the chosen C^k curve design algorithm. This results in K curves $\mathbf{x}_1(u), \ldots, \mathbf{x}_K(u)$ which we refer to as *sample curves*, see Fig 3.1. For each u the K points $\mathbf{x}_1(u), \ldots, \mathbf{x}_K(u)$ may be considered as image points of $\mathbf{x}_1^0, \ldots, \mathbf{x}_K^0$ under some deformation $F_u(\mathbf{x}_i^0) = \mathbf{x}_i(u)$.

In fact, the deformation F_u is not determined on all points of the moving body $\mathcal{B}$, but just on the sample points. However, it will be convenient to speak of a *deformed copy* $\mathcal{B}'(u)$ of the moving body. By applying the same curve design algorithm to all sample points we obtain a time dependent family of deformed copies $\mathcal{B}'(u)$ of the body $\mathcal{B}$, a so-called *deformed motion.*

Step 2: Best-fit Euclidean motion. In the second step of the algorithm we use the methods of Sect. 2.4 to compute to the deformed motion derived in step one, the closest Euclidean motion in the least squares sense using the metric of Sect. 2.2. In other words, at every time step of the motion, we perform registration with known correspondences of a Euclidean copy $\mathcal{B}^0$ of the moving body to the deformed copy $\mathcal{B}'(u)$.

3.1.1 Smoothness of Motion for General Curve Schemes

Although the registration usually only results in small corrections, we get the following theoretical problem. If we use a curve design algorithm that produces a C^k curve for each sample point, does our motion design algorithm produce a C^k motion? We formulate the following theorem.

Theorem 2. *If we use a C^k curve design algorithm, then our motion design algorithm generates a C^k motion, provided that the maximal eigenvalue $\lambda_m(u)$ of the matrix $N(u)$ in Equ. (2.24) has multiplicity one.*

Proof. The first step of our algorithm generates for all u a deformed copy $\mathcal{B}'(u)$ of the moving body $\mathcal{B}$. For the deformed motion every sample point runs on a C^k path by construction. Then, for all u we register the rigid moving body

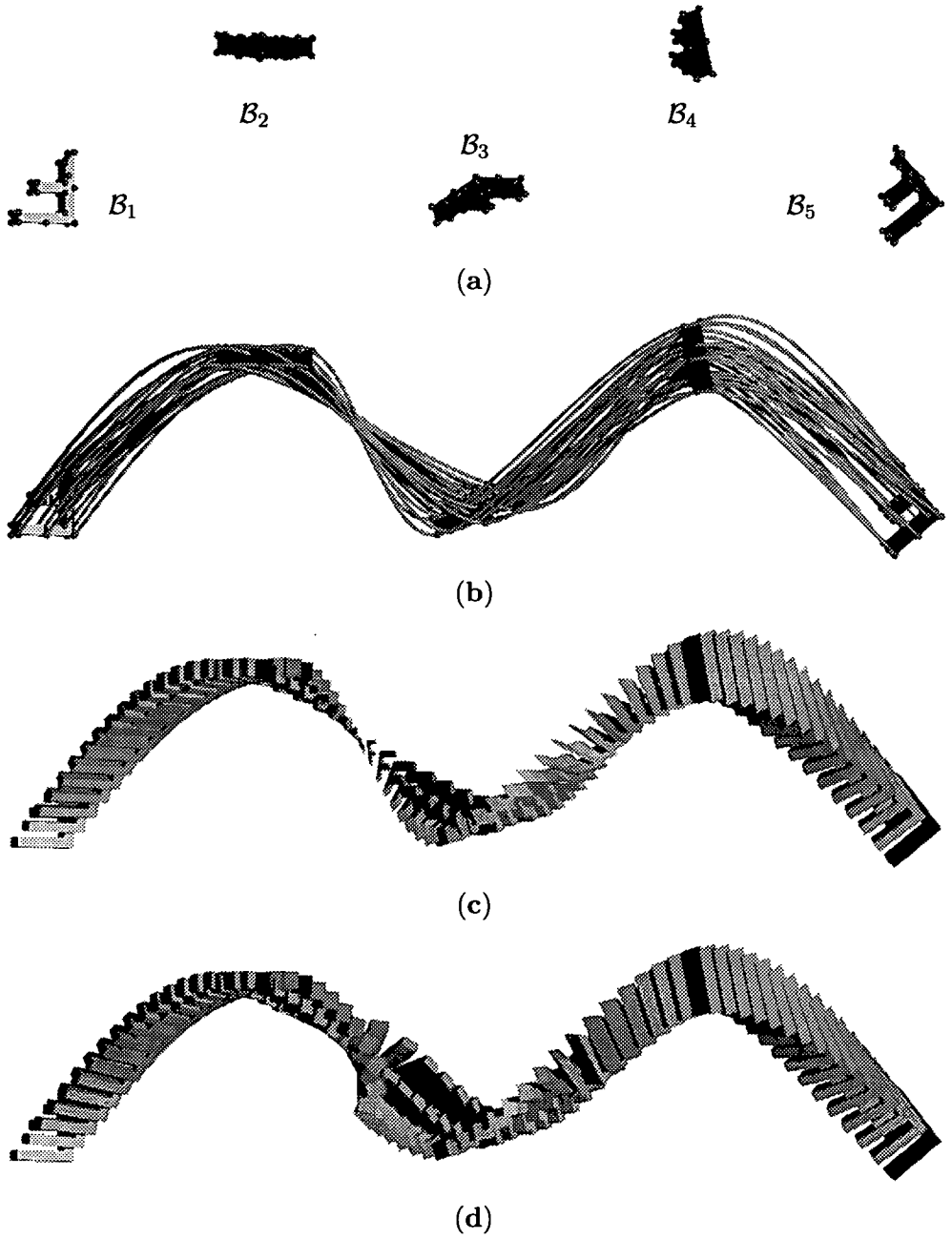


Figure 3.1: Steps of the CMD algorithm: (**a**) Sample points in their homologous input positions; (**b**) Sample curves computed by the curve design algorithm; (**c**) Deformed motion by curve design algorithm; (**d**) Best-fit Euclidean motion.

$\mathcal{B}^0$ to $\mathcal{B}'(u)$. We have to prove that the registration, described in Sect. 2.4, is a C^k operation. In order to find the Euclidean motion that performs the registration we have to compute the eigenvector $\mathbf{x}_m(u)$ corresponding to the maximal eigenvalue $\lambda_m(u)$ of the matrix $N(u)$ described by (2.24).

The eigenvalues of $N(u)$ are the zeros of the quartic polynomial $\det(N(u) - \lambda(u) \cdot I)$, with the identity matrix I. Eigenvectors $\mathbf{x}_m(u)$ corresponding to the maximal positive eigenvalue $\lambda_m(u)$ are found by solving the homogeneous linear system of equations $(N(u) - \lambda_m(u) \cdot I) \cdot \mathbf{x}_m(u) = 0$. From a unit eigenvector $\mathbf{x}_m(u)$ we find the rotation matrix $R(u)$, see (A.22). We have to show that all these operations are C^k, from which we can then conclude that we get a C^k motion.

The symmetric matrix $N(u)$ consists of entries that are polynomial in the coordinates of the points $\mathbf{x}_i$ and $\mathbf{y}_i$. Thus the function $N(u)$ is clearly C^k. Similarly, the computation of R from (a_0, a_1, a_2, a_3) is polynomial. The maximal eigenvalue of $N(u)$ is smoothly dependent on $N(u)$ if we can solve the equation

$$\det(N(u) - \lambda_m(u) \cdot I) = 0 \tag{3.1}$$

locally for λ_m. By the implicit function theorem this is possible if

$$\frac{\partial}{\partial \lambda} \det(N - \lambda \cdot I) \neq 0 \quad \text{for} \quad \lambda = \lambda_m, \tag{3.2}$$

i.e., if λ_m is a single zero of the polynomial $p(\lambda) = \det(N(u) - \lambda \cdot I)$. Next we have to show the following lemma:

Lemma 7. *Suppose that $A(u)$ and $\lambda(u)$ are C^k functions. If for all u in some interval $[a, b]$, $\lambda(u)$ is a single eigenvalue of the matrix $A(u)$, then there is a C^k function $\mathbf{x}(u)$, defined in the same interval, which is a unit eigenvector of $A(u)$ to the eigenvalue $\lambda(u)$.*

Proof. (of Lemma 7) Note that differentiability is a local property. We consider u in a neighborhood of some u_0. The rank of the matrix $(A - \lambda \cdot I)$ equals $n-1$. Thus, by removing one row and one column of this matrix we get a $(n-1)$ by $(n-1)$ matrix B with $\det(B) \neq 0$. Since $\det(B(u))$ is continuous it does not vanish in a neighborhood of u_0. Hence, for all u_0 there is a neighborhood where the same submatrix B is regular. Now we use this matrix B to solve the linear system of equations $(A - \lambda \cdot I) \cdot \mathbf{x} = 0$. Without loss of generality let B consist of the first $n-1$ rows and the first $n-1$ columns of $(A - \lambda \cdot I) =: (c_{ij})$. We first solve for $(x_1, \ldots, x_n)$ with $x_n = 1$ which means that we have to solve

$$B(x_1, \ldots, x_{n-1})^T = (-c_{1n}, \ldots, -c_{n-1,n})^T. \tag{3.3}$$

Hence the solution vector is found to be

$$(x_1, \ldots, x_{n-1})^T = B^{-1}(-c_{1n}, \ldots, -c_{n-1,n})^T. \tag{3.4}$$

This computation did not use the n-th equation of the linear system, but that one is automatically fulfilled, since we know that $(A - \lambda \cdot I)$ has rank $n-1$. The cofactor formula for computation of B^{-1} shows that B^{-1} is C^k if B is C^k. Thus, the eigenvector $(x_1, \ldots, x_{n-1}, 1)$ is C^k. By normalizing we get a C^k

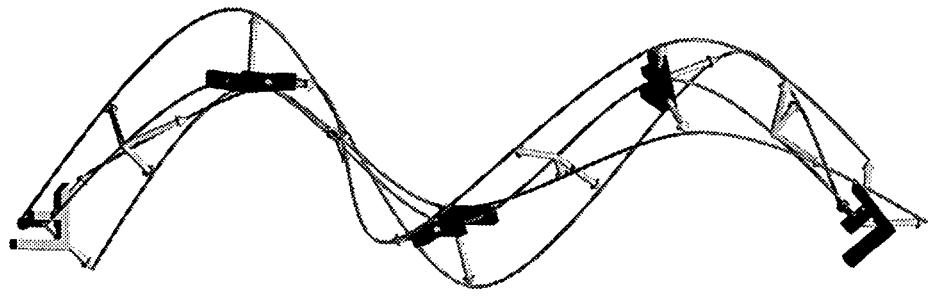


Figure 3.2: The CMD algorithm with a linear curve scheme generates in step 1 an affine motion, which is determined by the paths of four non-coplanar points.

unit eigenvector. This was a local construction, and it could happen that the ambiguity in the normalization $\mathbf{x} \mapsto \pm\mathbf{x}/\|\mathbf{x}\|$ yields locally defined unit vectors $\mathbf{x}(u)$ which do not fit together. This is easily repaired by replacing a finite number of locally defined $\mathbf{x}(u)'s$ by their opposites $-\mathbf{x}(u)$. □

This completes the proof of Theorem 2, i.e., that by using a C^k curve design algorithm, the CMD algorithm generates a C^k motion. □

3.2 The CMD Algorithm for Linear Curve Schemes

If we use the same *linear* curve design algorithm for all sample points, then it is not necessary to apply it to *all* sequences of homologous points, but only to *four* non-coplanar ones. This is so, since we obtain *affine copies* of the moving body as intermediate positions, and thus the deformed motion is actually an *affine motion.*

Lemma 8. *If we use one linear curve design algorithm to generate the paths of all sample points, then the resulting deformed motion is an* affine motion.

Proof. A proof of this property relies on the *linearity* of the used curve design algorithm. The j-th homologous input position $\mathbf{x}_k^j$ results from an initial position $\mathbf{x}_k^0$ by applying an affine map with matrix A_j and translational part $\mathbf{a}_j$,

$$\mathbf{x}_k^j = \mathbf{a}_j + A_j \cdot \mathbf{x}_k^0. \tag{3.5}$$

Using a linear curve design algorithm means that each intermediate position $\mathbf{x}_k(u)$ is a linear combination of the N homologous positions $\mathbf{x}_k^1, \ldots, \mathbf{x}_k^N$ of the k-th sample point $\mathbf{x}_k^0$,

$$\mathbf{x}_k(u) = \sum_{j=1}^{N} \mu_j(\mathbf{a}_j + A_j \cdot \mathbf{x}_k^0), \quad \mu_j \in \mathbb{R}. \tag{3.6}$$

After reordering, we obtain

$$\mathbf{x}_k(u) = \sum_j \mu_j \mathbf{a}_j + (\sum_j \mu_j A_j) \cdot \mathbf{x}_k^0 =: \mathbf{a} + A \cdot \mathbf{x}_k^0. \tag{3.7}$$

We see that the inserted position results from the initial one by application of an affine map with matrix A and translational part $\mathbf{a}$. □

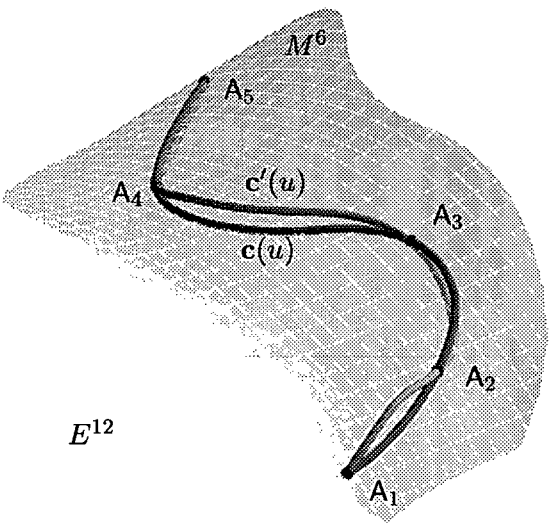


Figure 3.3: Motions as curves in the kinematic image space: an affine motion corresponds to a curve $\mathbf{c}'(u)$ in E^{12} and the best-fit Euclidean motion is a curve $\mathbf{c}(u) \subset M^6$ obtained as the orthogonal projection of $\mathbf{c}'(u)$ onto M^6.

Step 1: Affine motion via the linear curve scheme. Lemma 8 allows us to apply the curve design algorithm to only four sequences of non-coplanar homologous points. From these four sequences we can then derive the affine maps. Since we can choose any four non-coplanar homologous points to determine the affine maps, we use the homologous input positions of the four points $\mathbf{a}_0 = (0,0,0)^T, \mathbf{a}_1 = (1,0,0)^T, \mathbf{a}_2 = (0,1,0)^T, \mathbf{a}_3 = (0,0,1)^T$. Then we apply the curve design algorithm and we immediately get the affine maps at every time instant u: $\mathbf{a}_0(u)$ gives the translational part (this is the path of the barycenter of the moving body) and the columns of the matrix A describing the linear part are $\mathbf{a}_1(u) - \mathbf{a}_0(u)$, $\mathbf{a}_2(u) - \mathbf{a}_0(u)$, and $\mathbf{a}_3(u) - \mathbf{a}_0(u)$.

Step 2: Best-fit Euclidean motion. The computation of the closest Euclidean maps depends on all sample points. However, by using Lemma 2 and Lemma 6 of Sect. 2.4 there is no need to compute the intermediate positions of all the remaining sample points using the affine maps. We can directly compute the closest Euclidean motion to the affine motion in the metric of our kinematic image space by using the affine maps, see Algorithm 1.

3.2.1 Smoothness of Motion for Linear Curve Schemes

By applying the same *linear* curve design algorithm to all sample points we obtain a time dependent family of affine copies $\mathcal{B}'(u)$ of the body $\mathcal{B}$, a so-called affine motion. This special case allows a nice geometric interpretation and an even simpler derivation of our smoothness result.

To each affine map in 3-space, or equivalently to each affine image of the body $\mathcal{B}$, we associate a point in 12-dimensional space E^{12} as introduced in

Sect. 2.1. The image points of Euclidean maps form a 6-dimensional submanifold $M^6 \subset E^{12}$. The input positions $\mathcal{B}(u_i)$ correspond to points A_i on M^6. The affine motion $\mathcal{B}'(u)$ corresponds to a curve $\mathbf{c}'(u)$ which interpolates or approximates the points A_i, but does not lie on M^6 (see Fig. 3.3).

We can measure the distance between two affine maps in the Euclidean metric in E^{12}, introduced in Sect. 2.1. Thus, we also have an orthogonality in E^{12}. The way in which we compute a rigid body motion $\mathcal{B}(u)$ from the affine motion $\mathcal{B}'(u)$ via registration corresponds to an orthogonal projection of the curve $\mathbf{c}'$ to a curve $\mathbf{c} \subset M^6$. Depending on the multiplicity of the eigenvalues of $N(u)$ (and therefore the dimension of the eigenspaces) there are four different footpoints from a point $\mathbf{c}'(u)$ on the manifold M^6 or infinitely many.

J. Wallner [Wal02] has proved that the medial axis of M^6 is given by the singular affine maps. Thus, if the matrix A of the affine map which generates $\mathcal{B}'(u)$ and has $\mathbf{c}'(u)$ as image point in E^{12} has a positive determinant $\det A > 0$, then we can be sure that the conditions of Theorem 2 are fulfilled. Geometrically $\det A > 0$ means that $\mathbf{c}'(u)$ is not on the medial axis of M^6 and thus it possesses a unique closest point $\mathbf{c}(u)$ on M^6. We have proven again that by using a C^k curve design algorithm we generate a C^k rigid body motion.

3.3 Examples and Limitations

The implementation of the CMD algorithm is straightforward. For linear curve design algorithms, i.e., where the deformed motion is an affine motion, one proceeds as summarized in the following algorithm.

Algorithm 2 (CMD). *The algorithm consists of one preprocessing and two main steps:*

1. **Preprocessing:** *Choose sample points representing the rigid body $\mathcal{B}$. Then align the rigid body such that the barycenter is in the origin of the coordinate system and that the three principal moment axes are in direction of the coordinates axes.*

2. **Affine motion via curve design algorithm:** *Apply to the N homologous positions of the four points $(0,0,0),(1,0,0),(0,1,0),(0,0,1)$ the same linear curve scheme to get the affine maps $\mathbf{x}(u) = \mathbf{a}(u) + A(u) \cdot \mathbf{x}^0$.*

3. **Best-fit Euclidean motion:** *The path of the origin is $\mathbf{a}(u)$ and the best-fit rotation matrices $R(u)$ to $A(u)$ are computed using Algorithm 1.*

Thus we only need an implementation of the curve design algorithm we want to use, and either an eigenvalue solver or a SVD solver. The remaining code is then a mere 100 lines. We have implemented the CMD algorithm using Matlab 6.5. For the computation of the best-fit Euclidean motion we have tested both Matlab's 'eig' routine for computing the eigenvalues and eigenvectors of the matrix H, and the 'svd' routine of Matlab for the computation of the singular value decomposition of the matrix H. Our experience is that the 'eig' routine seems to be a bit more stable for almost degenerate cases ($\det A \approx 0$).

The total computation time (with the Matlab implementation on a Pentium 4 with 1.8 GHz), is for each motion shown in Figs. 3.4, 3.6–3.11 less than 0.1s. Since we only compute the motion at a discrete number of positions, the CMD algorithm is very well suited for the use of a subdivision scheme as the curve design algorithm.

Remark 3.1. We use an approximation of the rigid body by a triangular mesh with uniformly distributed vertices on the surface of the rigid body. Then these vertices are taken as sample points that we use for our computations. An algorithm for the computation of polyhedral mass properties such as barycenter and inertia tensor can be found in a paper by B. Mirtich [Mir96].

In the following we present examples of interpolating and approximating motions and show how to achieve motion modifications — such as tension effects — using the CMD algorithm. Figure 3.4 shows an open and closed interpolating rigid body motion of the Stanford bunny computed using the variational subdivision algorithm of [Kob96].

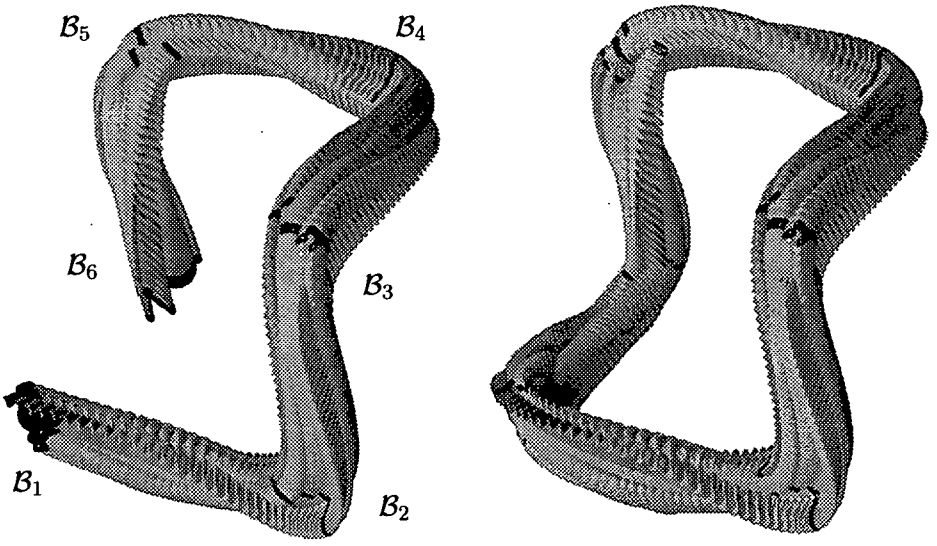


Figure 3.4: The CMD algorithm with interpolating variational subdivision: (left) motion from start position $\mathcal{B}_1$ to end position $\mathcal{B}_6$; (right) cyclic motion.

In Fig. 3.6 we compare Euclidean motions computed with the CMD algorithm to the same input positions (i.e., the same Euclidean maps $\mathsf{A}_i \in M^6$), using three different rigid bodies — the bunny, the window, and the gripper. In a preprocessing step all of them are translated such that their barycenter is in the origin, scaled such that they fit into a unit cube, and aligned such that their principal axes of inertia coincide with the coordinate axes. Then we use the same linear curve design algorithm in the CMD algorithm to compute the three motions. Linearity of the curve design algorithm implies that the paths of the barycenters of all motions coincide. However, because the inertia tensors

J are not equal, see Fig. 3.5 for inertia ellipsoids scaled by a factor of 0.02,

$$\begin{aligned} J_{\text{bunny}} &= \text{diag}(34.3006, 17.0387, 11.0698), \\ J_{\text{window}} &= \text{diag}(44.7291, 30.0250, 0.8846), \\ J_{\text{gripper}} &= \text{diag}(37.4229, 7.2722, 0.7893), \end{aligned}$$

the linear parts of the best-fit Euclidean motions are different as well. We visualize the computed linear parts by converting the rotation matrices into their corresponding unit quaternions $\mathbf{q}$ and then plotting the vector parts $V(\mathbf{q})$ of $\mathbf{q}$ for the motions of the bunny, the window, and the gripper as curves $\mathbf{c}_b$, $\mathbf{c}_w$, and $\mathbf{c}_g$ in $\mathbb{R}^3$, see Fig. 3.6 (**d**).

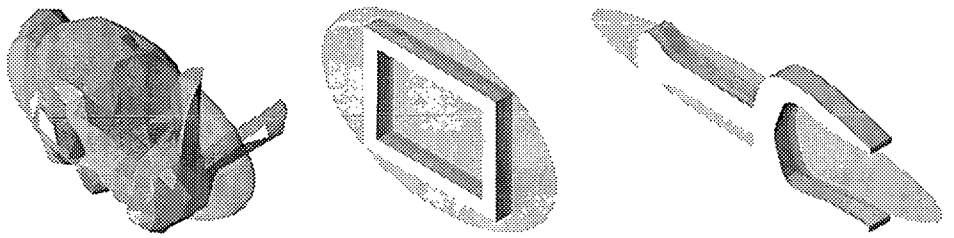


Figure 3.5: Inertia ellipsoids: (left) bunny, (middle) window, (right) gripper.

Figures 3.7,–3.10 show for the same five input positions $\mathcal{B}_1, \ldots, \mathcal{B}_5$ the affine and Euclidean motions generated by the CMD algorithm using linear, quadratic, cubic, and quartic B-spline curves. Figure 3.11 shows five Euclidean motions, interpolating the same input positions, computed with the CMD algorithm using ν-splines with $\nu = 0, 12, 25, 50, 99$. Figure 3.12 shows 1185 positions of the best-fit Euclidean motion of the Utah teapot interpolating 38 input positions, computed in 0.6s.

One limitation of the CMD algorithm is that if the affine motion — seen as a curve in E^{12} — gets too close to the medial axis, then $\det A \approx 0$ and the algorithm might fail to compute the best-fit orthogonal matrix to an almost singular affine matrix. Another limitation of the CMD algorithm is, that it does not preserve energy minimization. That means that if we use a variational curve design algorithm, then best-fit Euclidean motion is not an energy-minimizing motion. However, we generate a sufficiently good initial motion for the iterative optimization algorithm that we will use in Chapter 5 to compute energy-minimizing motions.

The focus of the CMD algorithm is the *design* of a rigid body motion with certain desired properties such as interpolating, approximating, smoothness, fairness, tension, etc. The strength of the CMD algorithm is that the properties of the employed curve design algorithm are transferred to the rigid body motion in a straightforward way. If necessary, one can use the computed dense set of discrete positions of the moving body and interpolate it with standard motion design techniques to get the desired motion representation e.g. in NURBS form.

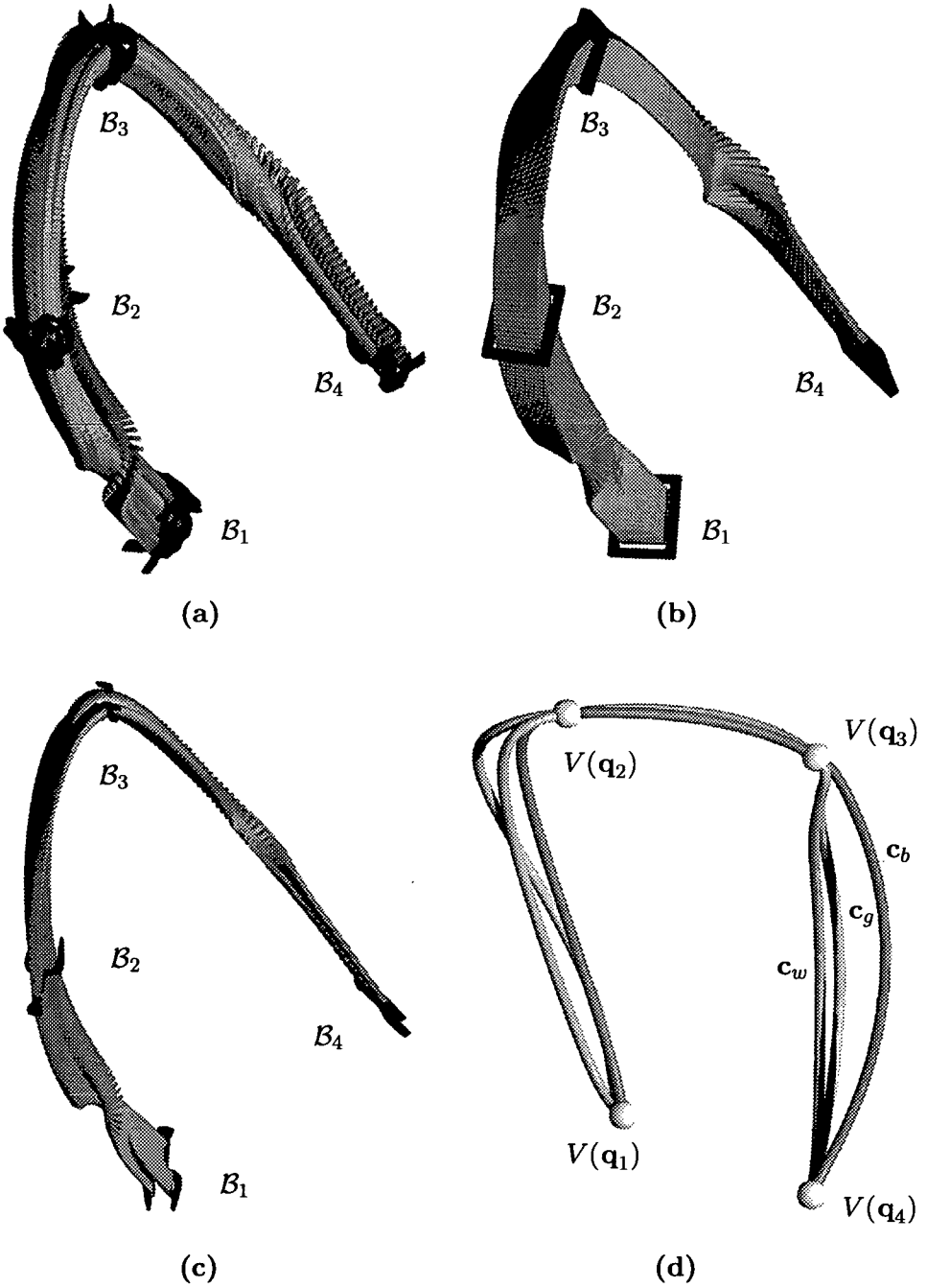
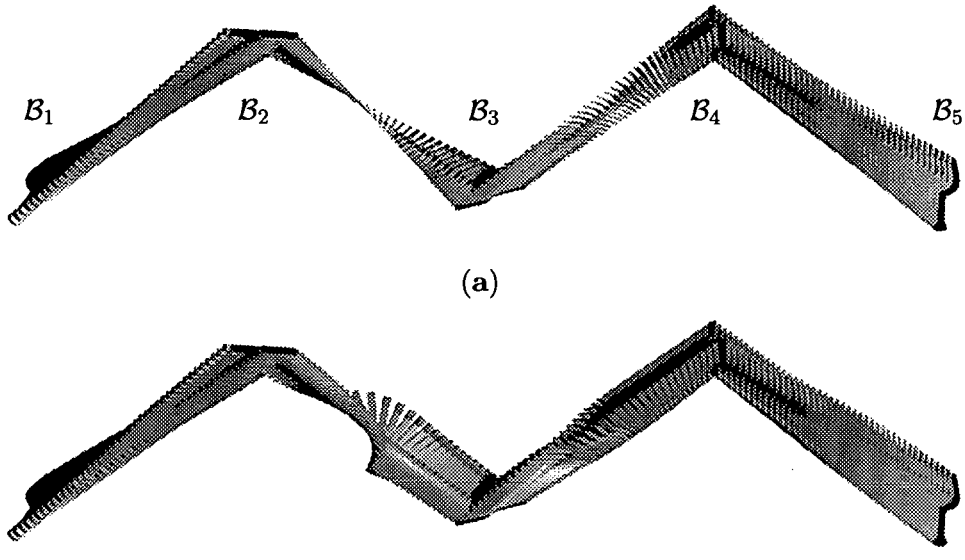


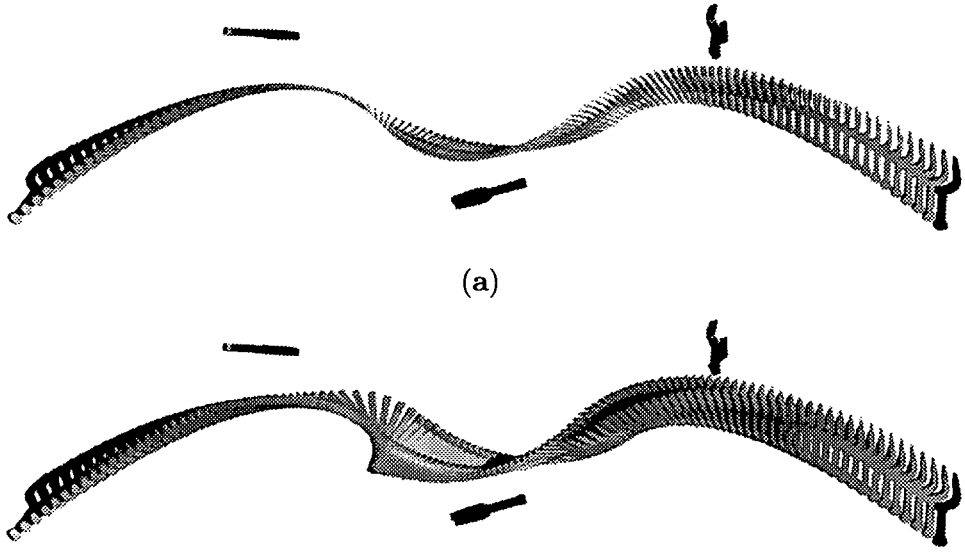
Figure 3.6: The CMD algorithm for the same input positions $\mathcal{B}_1, \ldots, \mathcal{B}_4$ (enlarged) using different moving bodies: **(a)** bunny, **(b)** window, **(c)** gripper. **(d)** visualizes the different linear parts of the motions of bunny, window, and gripper using the curves $\mathbf{c}_b$, $\mathbf{c}_w$, and $\mathbf{c}_g$ respectively.



(a)

(b)

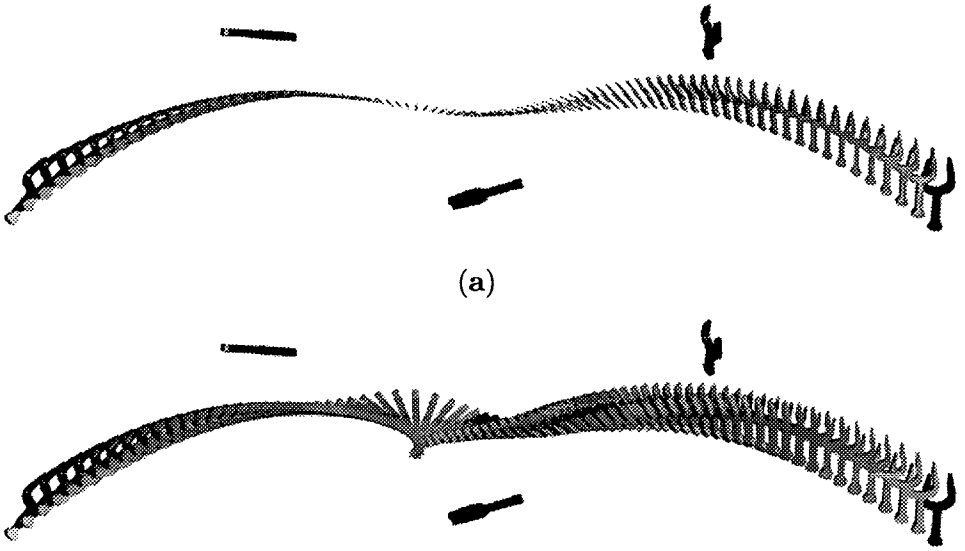
Figure 3.7: The CMD algorithm with *linear* B-splines: (**a**) affine motion; (**b**) best-fit Euclidean motion.



(a)

(b)

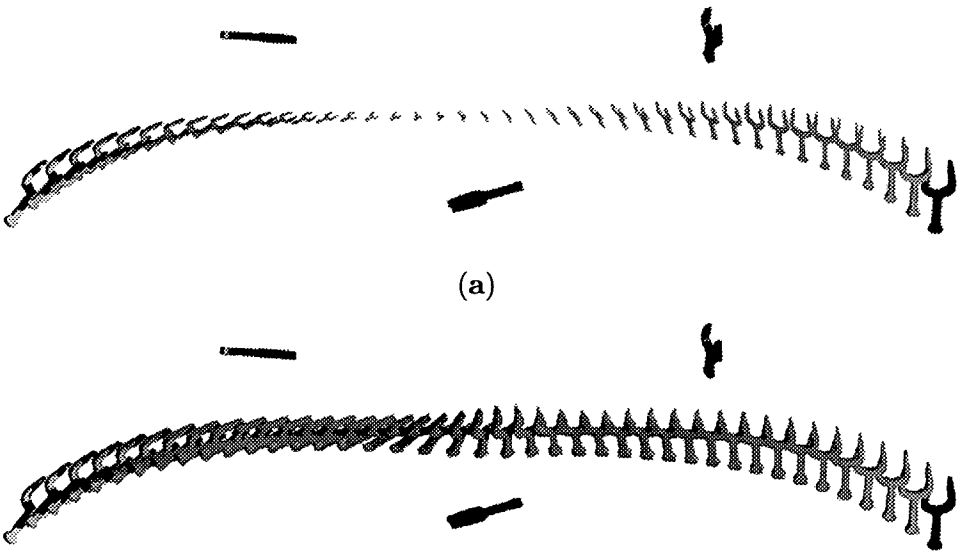
Figure 3.8: The CMD algorithm with *quadratic* B-splines: (**a**) affine motion; (**b**) best-fit Euclidean motion.



(a)

(b)

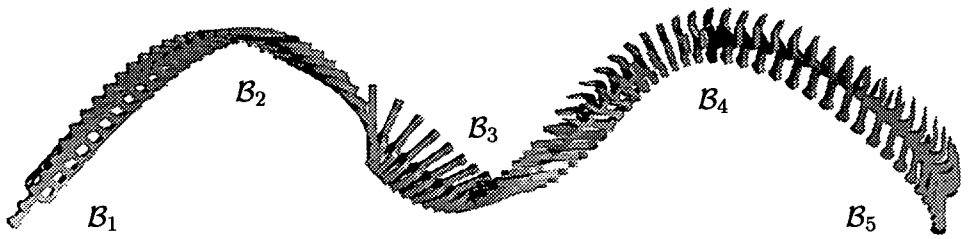
Figure 3.9: The CMD algorithm with *cubic* B-splines: (**a**) affine motion; (**b**) best-fit Euclidean motion.



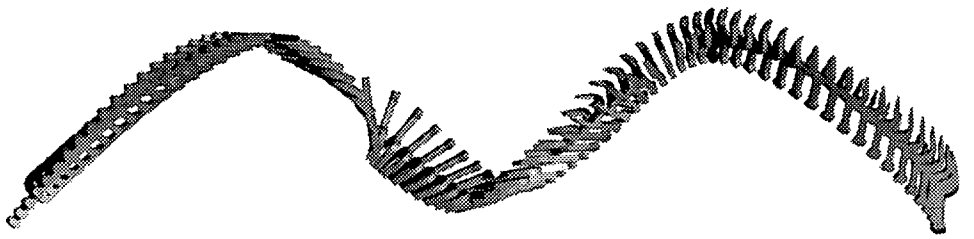
(a)

(b)

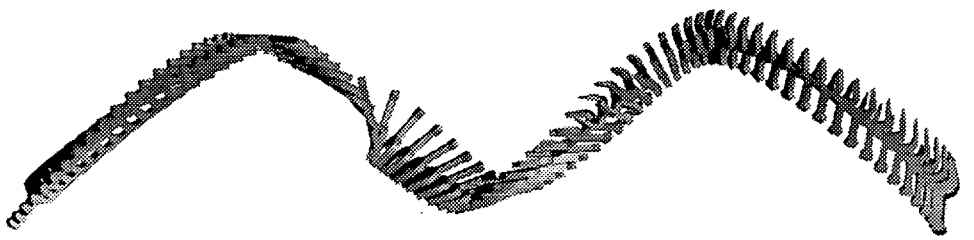
Figure 3.10: The CMD algorithm with *quartic* B-splines: (**a**) affine motion; (**b**) best-fit Euclidean motion.



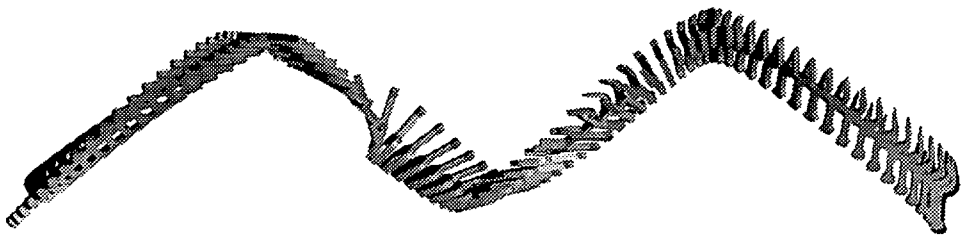
(a)



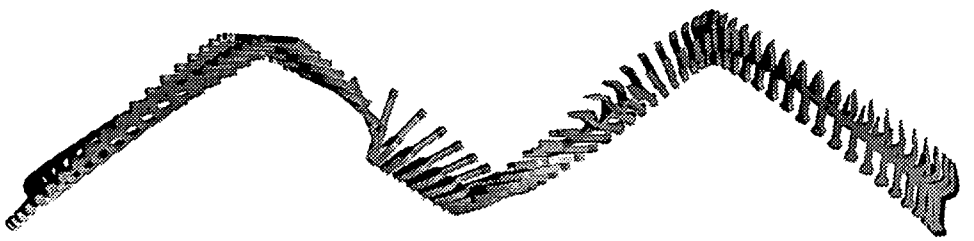
(b)



(c)



(d)



(e)

Figure 3.11: The CMD algorithm using ν-splines: best-fit Euclidean motion with (**a**) $\nu = 0$, (**b**) $\nu = 12$, (**c**) $\nu = 25$, (**d**) $\nu = 50$, (**e**) $\nu = 99$.

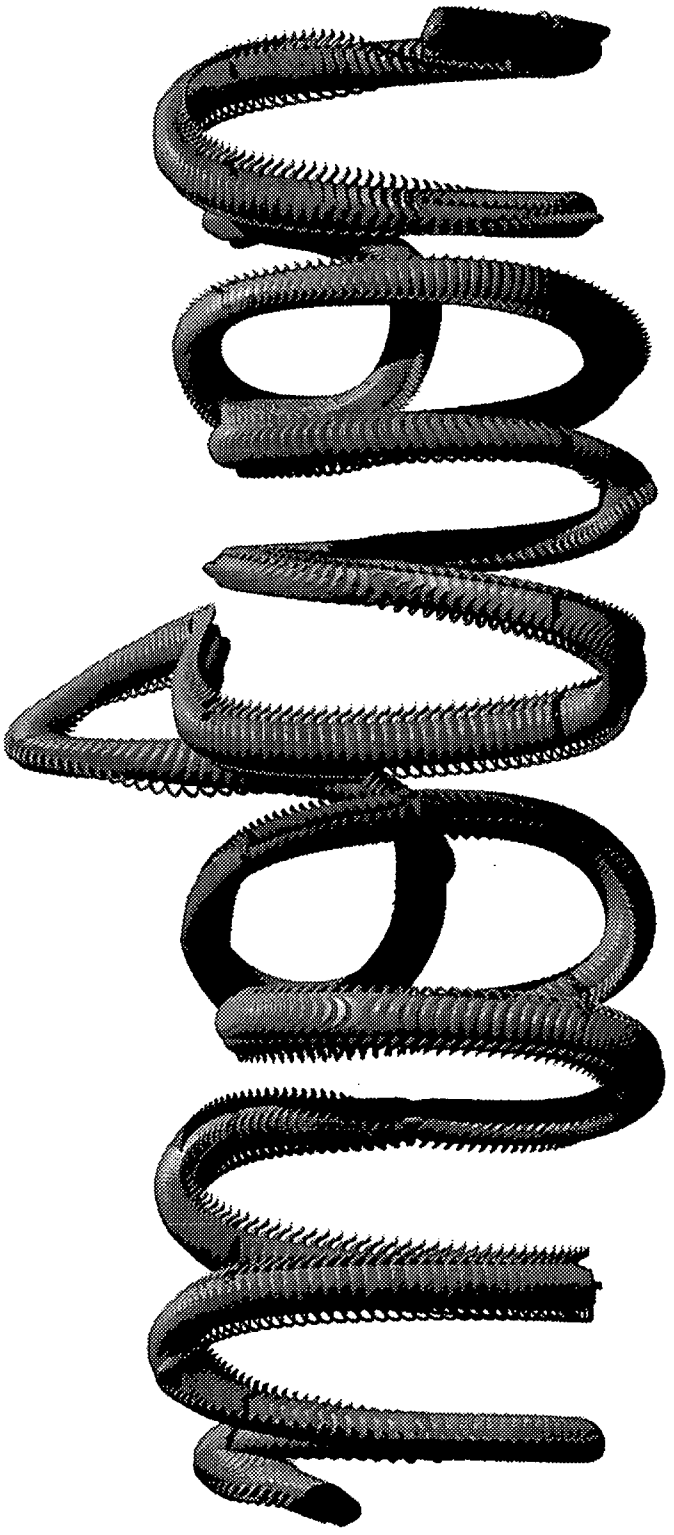


Figure 3.12: Euclidean motion computed with the CMD algorithm interpolating 38 input positions.

Chapter 4

Energy-Minimizing Splines in Manifolds

We formulate variational design of rigid body motions as variational design of a curve on the 6-dimensional manifold M^6 of Euclidean maps embedded into 12-dimensional space of affine maps. This is one special instance of the more general problem of computing an energy-minimizing spline on a m-dimensional regular surface S, embedded in Euclidean $\mathbb{R}^n$, $m < n$. Moreover, a sequence of points $\mathbf{p}_i \in S$, $i = 1, \ldots, N$ and real numbers $u_1 < \cdots < u_N$ are given. We are seeking a curve $\mathbf{c}(u)$ on S, which interpolates the given data, $\mathbf{c}(u_i) = \mathbf{p}_i$, and minimizes some energy functional.

In Section 4.1 we characterize parametric curves in S, which interpolate or approximate a sequence of given points $\mathbf{p}_i \in S$ and minimize a given energy functional. As energy functionals we study familiar functionals from spline theory, which are linear combinations of L^2 norms of certain derivatives. The characterization of the solution curves is similar to the well-known unrestricted case. The counterparts to cubic splines on a given surface, defined as interpolating curves minimizing the L^2 norm of the second derivative, are C^2; their segments possess fourth derivative vectors, which are orthogonal to S; at an end point, the second derivative is orthogonal to S. Analogously, we characterize counterparts to splines in tension. A characterization of quintic C^4 splines and smoothing splines can be found in [PH05].

Only on very special surfaces, some spline segments can be determined explicitly. In general, the computation has to be based on numerical optimization. In Section 4.2 we present a geometric optimization algorithm, which minimizes an energy of curves on surfaces of arbitrary dimension and codimension [HP04]. The concept is very general and can be applied for the computation of energy-minimizing spline curves on parametric surfaces, level sets, triangle meshes, and point set surfaces. It includes variational motion design, which we will study in Chapter 5, and the treatment of obstacles via barrier surfaces, the topic of Chapter 6.

4.1 Characterization of Energy-Minimizing Splines in Manifolds

Spline curves in Euclidean spaces which minimize the L^2 norm of the second derivative and related functionals are well understood and comprise one of the basics of Geometric Modeling. For a comprehensive discussion of the theory and computational issues we refer to [HL93]. The classical energy functionals are quadratic, which makes minimization easy. Geometric functionals which require nonlinear minimization techniques and their application to curve design have been studied e.g. by [BHS93] and [MS93]. Variational curve design has also been performed within the framework of subdivision [KS98], an approach we only touch briefly in Remark 6.

Variational design of curves on surfaces has received much less attention. A substantial amount of research focuses on the quaternion 3-sphere because of its well-known relationship with rigid body motions [BCGH92, BC94, JW02, PR97, RB97]. Minimizing the L^1 or L^2 norm of the *first* derivative of curves is a classical problem of Differential Geometry and leads to the geodesic lines of surfaces. Applications of this in Computer Vision and Image Processing are given in [CKS97, KMG98, Mal02, MS01, Sap01]. A series of contributions addresses intrinsically cubic splines in manifolds, i.e., the minimizers of the covariant acceleration [CLC01, GK85, NHP89, Noa03].

We concentrate on an *extrinsic* formulation of energy-minimizing curves in embedded manifolds; its definition uses the ambient space. We minimize classical quadratic energy functionals involving first and second derivatives, but with the nonlinear side condition that the solution curves are confined to surfaces. For *parametric* surfaces in $\mathbb{R}^3$, this is the topic of H. Bohl's thesis [Boh99], who proves the existence of a solution and computes it by quasi-Newton iteration.

In view of the applications we have in mind, it is necessary to work on surfaces of arbitrary dimension and codimension. Thus, we consider a m-dimensional surface S in Euclidean $\mathbb{R}^n$, $m < n$. Moreover, a sequence of points $\mathbf{p}_i \in S$, $i = 1, \ldots, N$ and real numbers $u_1 < \cdots < u_N$ shall be given. We are seeking interpolating splines on the surface S. Sometimes we will also call S a manifold; if not stated otherwise, we work with a manifold *without boundary*. Thus, we have closed or unbounded surfaces. As we will point out in Chapter 6, the computation of energy-minimizing splines which respect patch boundaries or holes in a surface, can be reduced to the computation of a spline on (another) surface without boundary.

4.1.1 Cubic Splines Revisited

Let us recall the situation, where we are not confined to a surface. Readers familiar with the derivation of interpolating cubic C^2 splines in $\mathbb{R}^n$ can jump to the next section.

Theorem 3. *Among all curves $\mathbf{x}(u) \subset \mathbb{R}^n$, whose first derivative is absolutely continuous, $\dot{\mathbf{x}} \in AC(I)$, and whose second derivative satisfies $\ddot{\mathbf{x}} \in L^2(I)$ on*

$I = [u_1, u_N]$, and which interpolate the given points, $\mathbf{x}(u_i) = \mathbf{p}_i$, the unique minimizer of

$$E_2(\mathbf{x}) = \int_{u_1}^{u_N} \|\ddot{\mathbf{x}}(u)\|^2 du, \tag{4.1}$$

is the interpolating C^2 cubic spline $\mathbf{c}(u)$.

Proof. Step 1: Energy minimizing property.

We assume the existence of a solution $\mathbf{c}$ and prove that for any other admissible interpolating curve $\mathbf{x}$ we have $E_2(\mathbf{x}) \geq E_2(\mathbf{c})$. We define an inner product of two curves $\mathbf{x}(u)$ and $\mathbf{y}(u)$ as

$$\langle \mathbf{x}, \mathbf{y} \rangle := \int_{u_1}^{u_N} \ddot{\mathbf{x}}(u) \cdot \ddot{\mathbf{y}}(u)\, du. \tag{4.2}$$

Then $\langle \cdot, \cdot \rangle$ is a symmetric positive semidefinite bilinear form which satisfies $\langle \mathbf{x}, \mathbf{x} \rangle = E_2(\mathbf{x})$. Bilinearity implies that

$$\langle \mathbf{x} - \mathbf{c}, \mathbf{x} - \mathbf{c} \rangle + 2\langle \mathbf{c}, \mathbf{x} - \mathbf{c} \rangle = \langle \mathbf{x}, \mathbf{x} \rangle - \langle \mathbf{c}, \mathbf{c} \rangle. \tag{4.3}$$

We want to show that $\langle \mathbf{c}, \mathbf{x} - \mathbf{c} \rangle = 0$, because then we can use $\langle \mathbf{x} - \mathbf{c}, \mathbf{x} - \mathbf{c} \rangle \geq 0$ to conclude that $\langle \mathbf{x}, \mathbf{x} \rangle - \langle \mathbf{c}, \mathbf{c} \rangle \geq 0$. We calculate

$$\langle \mathbf{c}, \mathbf{x} - \mathbf{c} \rangle = \int_{u_1}^{u_N} \ddot{\mathbf{c}} \cdot (\ddot{\mathbf{x}} - \ddot{\mathbf{c}})\, du. \tag{4.4}$$

Consider for the moment only the integral for $u \in [u_i, u_{i+1}]$. We integrate by parts twice to get

$$\begin{aligned} \int_{u_i}^{u_{i+1}} \ddot{\mathbf{c}} \cdot (\ddot{\mathbf{x}} - \ddot{\mathbf{c}})\, du &= \ddot{\mathbf{c}} \cdot (\dot{\mathbf{x}} - \dot{\mathbf{c}})\big|_{u_i}^{u_{i+1}} - \int_{u_i}^{u_{i+1}} \mathbf{c}^{(3)} \cdot (\dot{\mathbf{x}} - \dot{\mathbf{c}})\, du \\ &= \ddot{\mathbf{c}} \cdot (\dot{\mathbf{x}} - \dot{\mathbf{c}})\big|_{u_i}^{u_{i+1}} - \mathbf{c}^{(3)} \cdot (\mathbf{x} - \mathbf{c})\big|_{u_i}^{u_{i+1}} + \int_{u_i}^{u_{i+1}} \mathbf{c}^{(4)} \cdot (\mathbf{x} - \mathbf{c})\, du. \end{aligned} \tag{4.5}$$

Using the interpolation conditions $\mathbf{c}(u_i) = \mathbf{p}_i$, and the fact that for a cubic $\mathbf{c}$ the fourth derivative vector vanishes, $\mathbf{c}^{(4)} = 0$, we rewrite (4.5) as

$$\int_{u_i}^{u_{i+1}} \ddot{\mathbf{c}} \cdot (\ddot{\mathbf{x}} - \ddot{\mathbf{c}})\, du = \ddot{\mathbf{c}}_-(u_{i+1}) \cdot (\dot{\mathbf{x}}(u_{i+1}) - \dot{\mathbf{c}}(u_{i+1})) - \ddot{\mathbf{c}}_+(u_i) \cdot (\dot{\mathbf{x}}(u_i) - \dot{\mathbf{c}}(u_i)).$$

We now calculate $\langle \mathbf{c}, \mathbf{x} - \mathbf{c} \rangle$,

$$\begin{aligned} \langle \mathbf{c}, \mathbf{x} - \mathbf{c} \rangle = \; & - \; \ddot{\mathbf{c}}_+(u_1) \cdot (\dot{\mathbf{x}}(u_1) - \dot{\mathbf{c}}(u_1)) \\ & + \; \sum_{i=2}^{N-1} (\ddot{\mathbf{c}}_-(u_i) - \ddot{\mathbf{c}}_+(u_i)) \cdot (\dot{\mathbf{x}}(u_i) - \dot{\mathbf{c}}(u_i)) \\ & + \; \ddot{\mathbf{c}}_-(u_N) \cdot (\dot{\mathbf{x}}(u_N) - \dot{\mathbf{c}}(u_N)). \end{aligned} \tag{4.6}$$

Now (4.6) vanishes exactly if the following conditions hold

$$\begin{aligned} \ddot{\mathbf{c}}_+(u_1) &= 0, \\ (\ddot{\mathbf{c}}_-(u_i) - \ddot{\mathbf{c}}_+(u_i)) &= 0, \quad i = 2, \ldots, N-1, \\ \ddot{\mathbf{c}}_-(u_N) &= 0. \end{aligned} \tag{4.7}$$

These are exactly the conditions of an interpolating cubic C^2 spline with natural end conditions, if the solution exists.

Step 2: Uniqueness of the solution.

To show uniqueness of the solution of the variational problem let us assume that there are two solutions $\mathbf{c}, \mathbf{c}_1$. From the discussion above we see that $\langle \mathbf{c}, \mathbf{c} \rangle = \langle \mathbf{c}_1, \mathbf{c}_1 \rangle$. Equation (4.3) implies that $\langle \mathbf{c}_1 - \mathbf{c}, \mathbf{c}_1 - \mathbf{c} \rangle = 0$. This means that $\mathbf{c}_1 - \mathbf{c}$ is piecewise linear. Because of the interpolation conditions it follows that $(\mathbf{c}_1 - \mathbf{c})(u_i) = 0$ for $i = 1, \ldots, N$. Thus we have that $\mathbf{c}_1 - \mathbf{c} = 0$, which proves the uniqueness of the solution curve $\mathbf{c}$.

Step 3: Existence of a solution.

The existence of a solution $\mathbf{c}$ follows from the explicit computation of $\mathbf{c}$. The idea is to construct between each two adjacent points $\mathbf{p}_i = \mathbf{c}(u_i)$ and $\mathbf{p}_{i+1} = \mathbf{c}(u_{i+1})$ a cubic Bézier curve $\mathbf{c}_i$ with derivative vectors $\mathbf{m}_i$ and $\mathbf{m}_{i+1}$ at the end points such that adjacent cubic segments $\mathbf{c}_{i-1}$ and $\mathbf{c}_i$ fulfill the C^2 conditions at $\mathbf{c}(u_i)$,

$$\ddot{\mathbf{c}}_+(u_i) - \ddot{\mathbf{c}}_-(u_i) = 0 \quad \Leftrightarrow \quad \ddot{\mathbf{c}}_{i-1}(u_i) - \ddot{\mathbf{c}}_i(u_i) = 0. \tag{4.8}$$

In the following we denote by $\Delta_i := u_{i+1} - u_i$ the first forward difference. Note that the solution curve $\mathbf{c}$ also has to fulfill the C^1 condition, i.e., that the first derivatives at $\mathbf{c}(u_i)$ are equal. Using the derivative formula for Bézier curves, the C^1 condition can be written as

$$\dot{\mathbf{c}}(u_i) = \frac{3}{\Delta_{i-1}}(\mathbf{b}_{3i} - \mathbf{b}_{3i-1}) = \frac{3}{\Delta_i}(\mathbf{b}_{3i+1} - \mathbf{b}_{3i}) = \mathbf{m}_i. \tag{4.9}$$

From that we derive the Bézier points of the cubic segments $\mathbf{c}_i$,

$$\begin{aligned} \mathbf{b}_{3i} &= \mathbf{p}_i, \\ \mathbf{b}_{3i+1} &= \mathbf{p}_i + \frac{\Delta_i}{3}\mathbf{m}_i, \\ \mathbf{b}_{3i+2} &= \mathbf{p}_{i+1} - \frac{\Delta_i}{3}\mathbf{m}_{i+1}, \\ \mathbf{b}_{3i+3} &= \mathbf{p}_{i+1}. \end{aligned}$$

Note that the unknowns we have to determine are the derivative vectors $\mathbf{m}_i$. Since the $\mathbf{m}_i$ also have to fulfill the C^2 condition (4.8), we make again use of the derivative formula for Bézier curves to get

$$\begin{aligned} \ddot{\mathbf{c}}_i(u_i) &= \frac{6}{\Delta_i^2}(\mathbf{b}_{3i} - 2\mathbf{b}_{3i+1} + \mathbf{b}_{3i+2}), \\ \ddot{\mathbf{c}}_{i-1}(u_i) &= \frac{6}{\Delta_{i-1}^2}(\mathbf{b}_{3i-2} - 2\mathbf{b}_{3i-1} + \mathbf{b}_{3i}). \end{aligned}$$

The above two equations need to be equal in order for the solution curve $\mathbf{c}$ to be C^2 at $\mathbf{c}(u_i)$. Plugging in the Bézier points from the C^1 condition and rearranging the terms yields the following expression

$$\Delta_i \mathbf{m}_{i-1} + 2(\Delta_{i-1} + \Delta_i)\mathbf{m}_i + \Delta_{i-1}\mathbf{m}_{i+1} = 3\left(\frac{\Delta_i}{\Delta_{i-1}}\Delta\mathbf{p}_{i-1} + \frac{\Delta_{i-1}}{\Delta_i}\Delta\mathbf{p}_i\right). \tag{4.10}$$

The natural end conditions $\ddot{\mathbf{c}}_+(u_1) = 0$ and $\ddot{\mathbf{c}}_-(u_N) = 0$ yield the conditions

$$2\mathbf{m}_1 + \mathbf{m}_2 = \frac{3}{\Delta_1}\mathbf{p}_1 \quad \text{and} \quad \mathbf{m}_{N-1} + 2\mathbf{m}_N = \frac{3}{\Delta_N}\mathbf{p}_N. \tag{4.11}$$

Thus we have N equations (4.10) and (4.11) for the N unknown vectors $\mathbf{m}_i$ which we can write in matrix notation as

$$A \cdot (\mathbf{m}_1, \ldots, \mathbf{m}_N)^T = (\mathbf{r}_1, \ldots, \mathbf{r}_N)^T, \tag{4.12}$$

where the coefficient matrix A is

$$A = \begin{pmatrix} 2 & 1 & & & \\ \Delta_2 & 2(\Delta_1+\Delta_2) & \Delta_1 & & \\ \ddots & \ddots & \ddots & \ddots & \\ & \Delta_{N-1} & 2(\Delta_{N-2}+\Delta_{N-1}) & \Delta_{N-2} & \\ & & 1 & 2 & \end{pmatrix}, \tag{4.13}$$

and the vectors $\mathbf{r}_i$ of the right hand side are

$$\mathbf{r}_1 = \frac{3}{\Delta_1}\mathbf{p}_1, \tag{4.14}$$

$$\mathbf{r}_i = 3\left(\frac{\Delta_i}{\Delta_{i-1}}\Delta\mathbf{p}_{i-1} + \frac{\Delta_{i-1}}{\Delta_i}\Delta\mathbf{p}_i\right), \quad i = 2, \ldots, N-1, \tag{4.15}$$

$$\mathbf{r}_N = \frac{3}{\Delta_N}\mathbf{p}_N. \tag{4.16}$$

The coefficient matrix A is strictly diagonally dominant and thus it is regular. Therefore the linear system of equations (4.12) has a unique solution. This proves the existence (and again the uniqueness) of the solution of the posed variational problem. □

Remark 4.1. The explicit solution for closed curves is easily derived from the above linear system of equations.

4.1.2 The Counterparts to Cubic Splines on Surfaces

We would like to extend the well-known result of Sect. 4.1.1 to the case where the admissible curves $\mathbf{x}(u)$ are restricted to a given surface $S \subset \mathbb{R}^n$. We are not changing the functional E_2 of (4.1), whose interpretation requires an embedding space. We are considering the restriction to the surface as a constraint, rather than formulating the problem in terms of the intrinsic geometry of the manifold. As we will see later, this is a suitable formulation for the problems we would like to solve.

Theorem 4. *Consider real numbers $u_1 < \cdots < u_N$ and points $\mathbf{p}_1, \ldots, \mathbf{p}_N$ on an m-dimensional C^4 surface S in Euclidean space $\mathbb{R}^n$. We let $I = [u_1, u_N]$. Then among all C^1 curves $\mathbf{x} : I \to S \subset \mathbb{R}^n$, which interpolate the given data, i.e., $\mathbf{x}(u_i) = \mathbf{p}_i$, $i = 1, \ldots, N$, and whose restrictions to the intervals $[u_i, u_{i+1}], i = 1, \ldots, N-1$ are C^4, a curve $\mathbf{c}$ which minimizes the functional E_2 of (4.1) is C^2 and possesses segments $\mathbf{c}|[u_i, u_{i+1}]$, whose fourth derivative vectors are orthogonal to S. At the end points $\mathbf{p}_1 = \mathbf{c}(u_1)$ and $\mathbf{p}_N = \mathbf{c}(u_N)$ of the solution curve, the second derivative vector is orthogonal to S.*

Proof. If a solution curve $\mathbf{c}$ exists, the first variation of the energy functional must vanish there [GF63]. To express this condition, we consider neighboring curves $\mathbf{x}(u) \subset S$, written as

$$\mathbf{x}(u,\epsilon) = \mathbf{c}(u) + \mathbf{h}(u,\epsilon). \tag{4.17}$$

For any fixed $\tilde{u} \in I$, the curve $\mathbf{h}(\tilde{u},\epsilon)$ is a curve in S with $\mathbf{h}(\tilde{u},0) = 0$. Its Taylor expansion at $\epsilon = 0$ reads

$$\mathbf{h}(\tilde{u},\epsilon) = \epsilon\mathbf{h}_\epsilon(\tilde{u},0) + \frac{\epsilon^2}{2}\mathbf{h}_{\epsilon\epsilon}(\tilde{u},0) + \ldots,$$

where the subscripts indicate differentiation. Note that $\mathbf{h}_\epsilon(\tilde{u},0) =: \mathbf{t}(\tilde{u})$ is a tangent vector of S at $\mathbf{c}(\tilde{u})$. The displacement curves $\mathbf{h}$ to the given interpolation points vanish, $\mathbf{h}(u_i,\epsilon) = 0$, for all ϵ. In particular, we have $\mathbf{h}_\epsilon(u_i,0) = \mathbf{t}(u_i) = 0$.

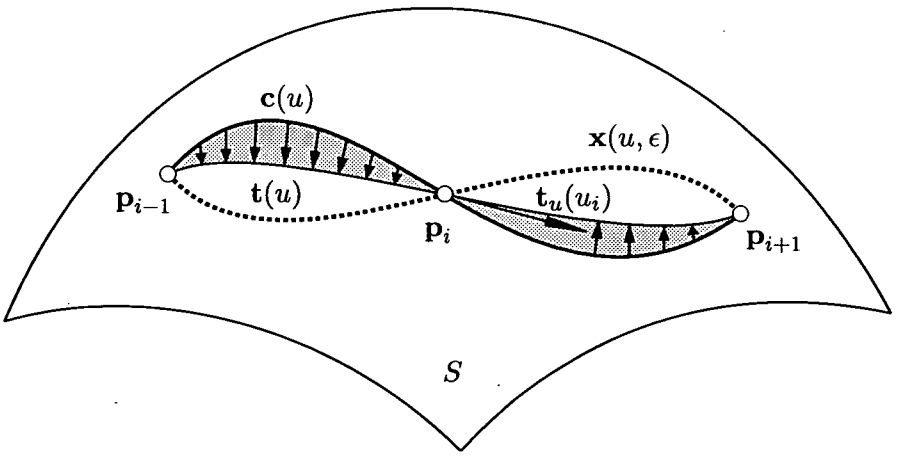


Figure 4.1: Displacement of the solution curve for deriving the first variation of the energy.

For the following, it is important to see that the mixed partial derivative vector $\mathbf{h}_{\epsilon u}(u_i,0) = \mathbf{t}_u(u_i)$ is a tangent vector of S at $\mathbf{c}(u_i) = \mathbf{p}_i$. Geometrically, this follows from the fact that the curve $\mathbf{c}(u) + \mathbf{t}(u)$ is a curve on the ruled surface $\mathbf{r}(u,v) = \mathbf{c}(u) + v\mathbf{t}(u)$, which touches S along $\mathbf{c}$. This curve passes through each interpolation point $\mathbf{p}_i = \mathbf{c}(u_i) + \mathbf{t}(u_i)$, and thus its derivative vector $\mathbf{c}_u(u_i) + \mathbf{t}_u(u_i)$ is a tangent vector of S. Together with the tangency of $\mathbf{c}_u(u_i)$ this implies tangency of $\mathbf{t}_u(u_i)$.

Whatever field of displacement curves $\mathbf{h}(u,\epsilon)$ we have chosen, the energy functional must assume a stationary value at $\epsilon = 0$,

$$\frac{d}{d\epsilon}E_2(\mathbf{x}(u,\epsilon))\big|_{\epsilon=0} = 0. \tag{4.18}$$

In view of

$$E_2(\mathbf{x}(u,\epsilon)) = \int_I [\ddot{\mathbf{c}}(u) + \mathbf{h}_{uu}(u,\epsilon)]^2 du = \int_I [\ddot{\mathbf{c}} + \epsilon\mathbf{t}_{uu} + \epsilon^2(\cdots)]^2 du,$$

this is equivalent to

$$\int_I \ddot{\mathbf{c}} \cdot \mathbf{t}_{uu}\, du = 0. \tag{4.19}$$

Integration by parts (twice) on each segment yields

$$\int_{u_i}^{u_{i+1}} \ddot{\mathbf{c}} \cdot \mathbf{t}_{uu}\, du = \ddot{\mathbf{c}} \cdot \mathbf{t}_u\Big|_{u_i}^{u_{i+1}} - \mathbf{c}^{(3)} \cdot \mathbf{t}\Big|_{u_i}^{u_{i+1}} + \int_{u_i}^{u_{i+1}} \mathbf{c}^{(4)} \cdot \mathbf{t}\, du.$$

We first note that the middle term vanishes, since $\mathbf{t}(u_i) = 0$. Summing up over all intervals and denoting left and right derivatives at the knots u_i by a subscript $-$ and $+$, respectively, the condition for a solution $\mathbf{c}$ reads,

$$\begin{aligned} 0 = -\ddot{\mathbf{c}}_+(u_1) \cdot \mathbf{t}_u(u_1) \;&+\; \sum_{i=2}^{N-1} [(\ddot{\mathbf{c}}_-(u_i) - \ddot{\mathbf{c}}_+(u_i)) \cdot \mathbf{t}_u(u_i)] \\ &+\; \ddot{\mathbf{c}}_-(u_N) \cdot \mathbf{t}_u(u_N) + \int_I \mathbf{c}^{(4)} \cdot \mathbf{t}\, du. \end{aligned} \tag{4.20}$$

Since $\mathbf{t}(u)$ is an arbitrary tangent vector field along $\mathbf{c}$, and $\mathbf{t}_u(u_i)$ are arbitrary tangent vectors at the given interpolation points, a solution curve $\mathbf{c}(u)$ must satisfy the following orthogonality conditions to the surface S. To formulate them, we denote by *tpr* the orthogonal projection of a vector at a point $\mathbf{p} \in S$ onto the tangent space of S at $\mathbf{p}$,

$$tpr\ \ddot{\mathbf{c}}_+(u_1) = tpr\ \ddot{\mathbf{c}}_-(u_N) = 0, \tag{4.21}$$

$$tpr\ (\ddot{\mathbf{c}}_-(u_i) - \ddot{\mathbf{c}}_+(u_i)) = 0, \quad i = 2, \ldots, N-1, \tag{4.22}$$

$$tpr\ \mathbf{c}^{(4)}(u) = 0 \quad \text{on each interval } [u_i, u_{i+1}]. \tag{4.23}$$

Exactly these conditions are stated in Theorem 4. Equation (4.21) means that the second derivative vector at the end points is orthogonal to the surface; this implies (but is not equivalent to) vanishing geodesic curvature at the end points. Equation (4.22) shows that the tangential component of the second derivative is continuous along $\mathbf{c}$; we call this behaviour *tangentially* C^2 or briefly TC^2. It will be shown later that in view of the C^4 manifold S this implies C^2. Finally, by Equ. (4.23), the fourth derivative vectors of the curve (which may be discontinuous at the interpolation points) are orthogonal to the manifold S. Vanishing fourth derivative characterizes a cubic curve; since here only the tangential component vanishes, we speak of a *tangentially cubic curve* in the following. To complete the proof, we show the following result:

Lemma 9. *On an m-dimensional C^k manifold $S \subset \mathbb{R}^n$, $m < n$, we consider a curve $\mathbf{c}(u)$, which is C^k ($k \geq 2$) everywhere except at a point $\mathbf{c}(u_0)$. At $\mathbf{c}(u_0)$, all derivatives up to order k have a continuous tangential component, i.e.,*

$$tpr(\mathbf{c}_-^{(i)}(u_0) - \mathbf{c}_+^{(i)}(u_0)) = 0,\ i = 1, \ldots, k. \tag{4.24}$$

Then, the curve is also C^k at $\mathbf{c}(u_0)$.

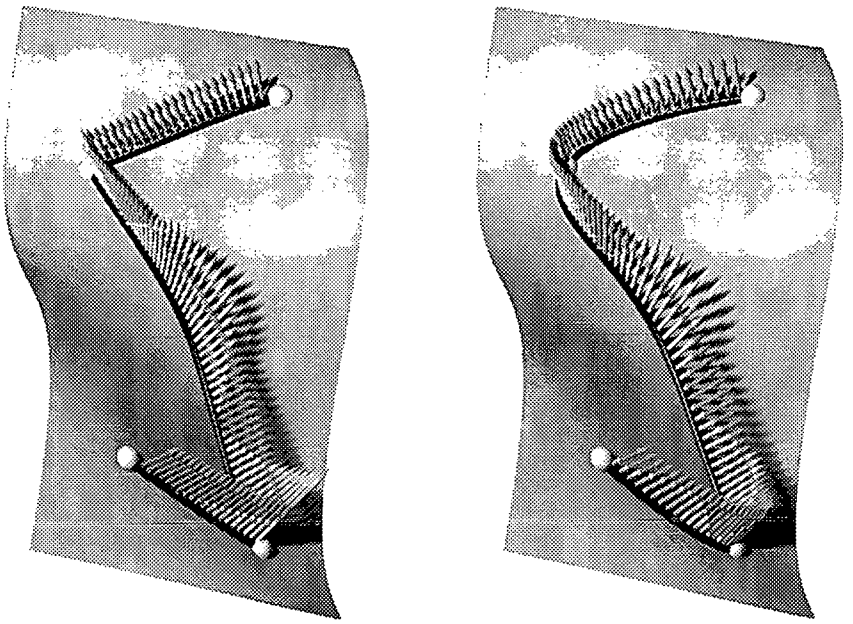


Figure 4.2: Geometric characterization of a minimizer: For a minimizer of (left) E_1 and (right) E_2, the second respectively fourth derivative vectors (yellow) are orthogonal to the surface.

Proof. We may represent the manifold at least in a neighborhood of $\mathbf{c}(u_0)$ as an intersection of $n-m$ hypersurfaces, whose implicit representations shall be

$$F_j(\mathbf{x}) = 0, \quad j = 1, \ldots, n-m.$$

Since the curve lies in all these hypersurfaces, we have the identity $F_j(\mathbf{c}(u)) = 0$ in u. By differentiation, we obtain $\nabla F_j(\mathbf{c}) \cdot \dot{\mathbf{c}} = 0$, and for the i-th derivative we find an expression of the form

$$G_{i-1}(\mathbf{c}, \ldots, \mathbf{c}^{(i-1)}) + \nabla F_j(\mathbf{c}) \cdot \mathbf{c}^{(i)} = 0. \tag{4.25}$$

Here, $G_{i-1}(\mathbf{c}, \ldots, \mathbf{c}^{(i-1)})$ abbreviates a function, which depends on F_j, up to differentiation order i, and on $\mathbf{c}$, but only up to differentiation order $i-1$. At $\mathbf{c}(u_0)$ this holds for the left and right derivatives. The first derivative is tangential, and thus $\mathbf{c}$ is C^1 at u_0. This is the basis of an induction on the differentiation order i. Assuming continuity of the derivative $\mathbf{c}^{(i-1)}$ at u_0, Equ. (4.25) expresses that the components $\nabla F_j(\mathbf{c}) \cdot \mathbf{c}^{(i)}$ of the i-th derivative are continuous at u_0, since they equal $G_{i-1}(\mathbf{c}, \ldots, \mathbf{c}^{(i-1)})$, which is continuous by the induction hypothesis. Since the vectors $\nabla F_j(\mathbf{c}(u_0))$, $j = 1, \ldots, n-m$, span the normal space of S at $\mathbf{c}(u_0)$, we have shown that the normal component of the i-th derivative is continuous. The tangential component is continuous by (4.24) and thus we have shown continuity of $\mathbf{c}^{(i)}$. This completes the proof of Lemma 9 and thus also the proof of Theorem 4. Note that the case $k = 2$ follows also immediately by Meusnier's theorem. □

The minimizers of the energy functional E_2 in (4.1) on surfaces possess a characterization which is very similar to the familiar cubic C^2 splines (revisited in Sect. 4.1.1). The minimizers on surfaces are C^2 and tangentially cubic. However, the problem is nonlinear and the minimizers can in general not be computed explicitly. In Sect. 4.2 we present a numerical optimization algorithm to compute such energy-minimizing curves. Existence of the solution follows from the work of Bohl [Boh99] and Wallner [Wal04a]. Uniqueness cannot be expected, as will be clear from the following considerations.

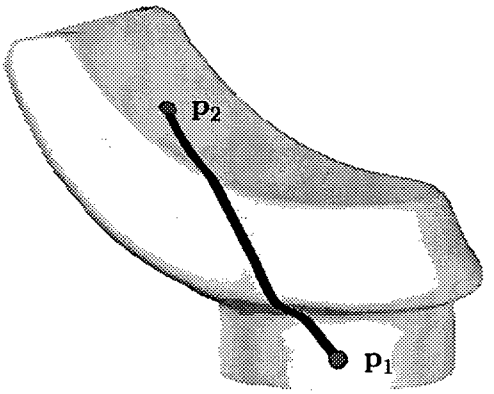


Figure 4.3: Geodesic between two points $\mathbf{p}_1$ and $\mathbf{p}_2$.

4.1.3 Geodesic Curves on Surfaces

Let us replace the energy in (4.1) by the L^2 norm of the first derivative,

$$E_1(\mathbf{x}) = \int_{u_1}^{u_2} \|\dot{\mathbf{x}}(u)\|^2 du. \tag{4.26}$$

Moreover, we just prescribe the two end points $\mathbf{p}_1$ and $\mathbf{p}_2$. Then we find with the same approach as in the proof of Theorem 4 that the curve's second derivative vectors $\ddot{\mathbf{c}}$ are orthogonal to S. In particular, $\dot{\mathbf{c}}$ and $\ddot{\mathbf{c}}$ are orthogonal, and hence $\|\dot{\mathbf{c}}\|^2 = const.$ This proves, that the curve is parameterized by a constant multiple of its arc length. Moreover, it shows that $\ddot{\mathbf{c}}$ represents the principal normal, and orthogonality of the principal normal to S characterizes a geodesic (a shortest path on S). Thus we find the known result that *the minimizers of E_1 are geodesics in a scaled arc length parameterization.* Note that the functionals we are considering are also optimizing the parameterization. There are no simple global uniqueness results for geodesics, so we cannot expect such results for the more involved case of splines.

4.1.4 The Counterparts to Splines in Tension on Surfaces

A linear combination of energies (4.26) and (4.1) leads in the unrestricted case to the well known *splines in tension* as minimizers [Sch66]. The counterpart on manifolds is characterized in the following theorem, whose proof can be omitted since it is completely analogous to that of Theorem 4.

Theorem 5. *Consider data points, parameter values and admissible curves on a surface S as in Theorem 4. Then, a minimizer of the functional*

$$E_t(\mathbf{x}) = \int_{u_1}^{u_N} (\|\ddot{\mathbf{x}}(u)\|^2 + w\|\dot{\mathbf{x}}(u)\|^2)\, du, \quad w = const > 0, \tag{4.27}$$

is a C^2 curve which satisfies

$$tpr\ (\mathbf{c}^{(4)}(u) - w\ddot{\mathbf{c}}(u)) = 0 \tag{4.28}$$

on all segments. The end conditions are $tpr\,\ddot{\mathbf{c}}_+(u_1) = tpr\,\ddot{\mathbf{c}}_-(u_N) = 0$.

Increasing w brings the solution closer to the curve which minimizes E_1, i.e., the curve composed by geodesic segments between the interpolation points. Hence, the parameter w controls the tension of the curve, see Fig. 4.4.

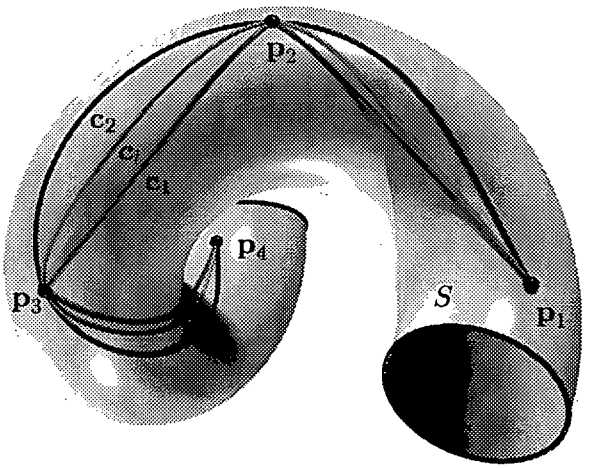


Figure 4.4: Spline curves $\mathbf{c}_1$, $\mathbf{c}_t$, $\mathbf{c}_2$ on a surface S, interpolating the same points $\mathbf{p}_1, \ldots, \mathbf{p}_4$ and minimizing energies E_1, E_t, and E_2, respectively.

Remark 4.2. One can provide analogous results (see [PH05]) for the surface counterparts of *quintic C^4 splines* (minimizers of the L^2 norm of the third derivative) and *smoothing splines*. The latter approximate the given points and are related to C^2 cubic splines in the following way. Reinsch [Rei67] relaxed the interpolation conditions $\mathbf{x}(u_i) - \mathbf{p}_i = 0$ by adding the sum of squared interpolation errors as a penalty term to E_2,

$$E_2^s(\mathbf{x}) = \lambda \int_{u_1}^{u_N} \|\ddot{\mathbf{x}}(u)\|^2 du + \mu \sum_{i=1}^{N} \|\mathbf{x}(u_i) - \mathbf{p}_i\|^2, \quad \lambda, \mu > 0. \tag{4.29}$$

The minimizers, called *smoothing splines*, found a variety of applications in the analysis of observational data [Wah90]. The choice of the smoothing parameter $\lambda : \mu$ in this method for data approximation is not a simple problem. Often one uses statistical methods for this critical task [Wah90]. We should also mention that an analogous functional is used for freeform surface fitting in reverse engineering applications [VM02].

The numerical optimization procedure (Algorithm 4) presented in Sect. 4.2 can handle the counterparts of quintic splines and smoothing splines on surfaces as well, but we concentrate on the counterparts of cubic splines and splines in tension on surfaces. In the following we will thus mainly work with the energy functionals E_2 of (4.1), E_t of (4.27), and for comparison reasons also with E_1 of (4.26).

4.2 Computation of Energy-Minimizing Splines in Manifolds

We are minimizing *quadratic functionals* such as E_2 or E_t. After discretization we obtain *quadratic functions*. The restriction of the curves to a surface yields constraints, and thus the numerical computation of splines in manifolds requires an algorithm for the constrained minimization of a quadratic function. For this purpose we propose a projected gradient algorithm, whose stepsize control is guided by a geometric interpretation of an error estimate. The optimization procedure we present is related to research on active curves [BI98, KWT87], geometric flows of curves on surfaces [CBMO02]), and snakes on surfaces [LL02].

4.2.1 The Basic Geometric Optimization Algorithm

Consider minimization of a quadratic function

$$F : \mathbb{R}^D \to \mathbb{R}, \quad F(\mathbf{x}) = \mathbf{x}^T \cdot Q \cdot \mathbf{x} + 2\mathbf{q}^T \cdot \mathbf{x} + q, \tag{4.30}$$

with a symmetric positive definite matrix Q, under the constraint that $\mathbf{x}$ lies in some surface $\Phi \subset \mathbb{R}^D$. We assume that Φ has dimension m and that it is smooth in the area we work in.

The geometric approach to this minimization problem views the matrix Q as matrix of the inner product $\langle \mathbf{x}, \mathbf{y} \rangle := \mathbf{x}^T \cdot Q \cdot \mathbf{y}$, of a Euclidean metric in $\mathbb{R}^D$. F assumes its minimum in the point $\mathbf{p} = -Q^{-1} \cdot \mathbf{q}$. Up to an unimportant additive constant, F then equals

$$F(\mathbf{x}) = (\mathbf{x} - \mathbf{p})^T \cdot Q \cdot (\mathbf{x} - \mathbf{p}). \tag{4.31}$$

This is the squared distance $\|\mathbf{x} - \mathbf{p}\|^2$ of points $\mathbf{p}$ and $\mathbf{x}$ in the Euclidean norm mentioned above. Here, and in this entire section, 'distance', 'orthogonality', and related concepts refer to the metric defined by the matrix Q. The minimum of F on the surface Φ is attained at the point $\mathbf{p}^*$, which is closest to $\mathbf{p}$, see Fig. 4.5. The point $\mathbf{p}^*$ is the normal footpoint of $\mathbf{p}$ on Φ.

We propose the following iterative procedure to compute $\mathbf{p}^*$. It consists of repeated application of the following two steps (see Fig. 4.5): $\mathbf{x}_c$ (the current point) is initialized with an initial guess $\mathbf{x}_0$ for the minimizer.

1. Compute the tangent space T^m of Φ at the current iterate $\mathbf{x}_c$ and project the point $\mathbf{p}$ orthogonally into T^m, which results in the point $\mathbf{p}_T$.

2. Compute an appropriate stepsize s and project $\mathbf{x}_s := \mathbf{x}_c + s(\mathbf{p}_T - \mathbf{x}_c)$ onto Φ; this yields the next iterate $\mathbf{x}_+$.

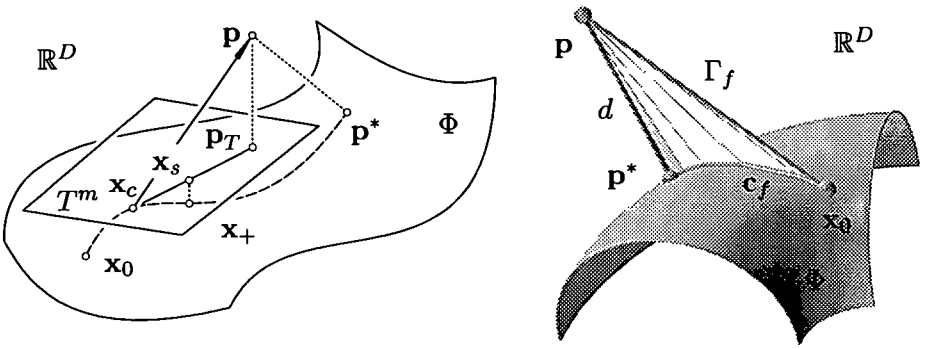


Figure 4.5: (Left) Footpoint computation with a projected gradient algorithm. (Right) Footpoint cone.

4.2.2 Convergence Analysis

The choice of the stepsize s requires a discussion of error estimates. We briefly summarize results here and omit proofs, which can be found in [PH04].

Under the assumption that Φ has derivatives up to third order, the current error $\mathbf{e}_c := \mathbf{x}_c - \mathbf{p}^*$ and the error at the next iteration step $\mathbf{e}_+ = \mathbf{x}_+ - \mathbf{p}^*$ are related via

$$\|\mathbf{e}_+\| \leq |1 - s + sd\kappa_{\max}| \, \|\mathbf{e}_c\| + C \, \|\mathbf{e}_c\|^2. \tag{4.32}$$

d equals the distance of $\mathbf{p}$ and $\mathbf{p}^*$. $\kappa_{\max}$ is the largest (in the sense of absolute values) normal curvature of the surface Φ in the point $\mathbf{p}^*$, with respect to the normal vector $\mathbf{n} := \mathbf{p} - \mathbf{p}^*$. C is a constant.

Since $\mathbf{p} - \mathbf{x}_c$ is the negative gradient of $\frac{1}{2}\|\mathbf{x} - \mathbf{p}\|^2$ (the squared distance function to $\mathbf{p}$) at $\mathbf{x}_c$, the method is a *projected gradient algorithm.*

Remark 4.3. The normal curvature κ of a curve $\mathbf{c}_f$ in Φ through $\mathbf{p}^*$, with respect to the normal vector $\mathbf{n}$, has the following interpretation: Connecting $\mathbf{p}$ with $\mathbf{c}_f$ yields a cone Γ_f (see Fig. 4.5). By developing this cone into the plane, $\mathbf{c}_f$ is transformed into a planar curve $\widetilde{\mathbf{c}}_f$, whose (ordinary) curvature is precisely κ [dC76, Spi75].

Remark 4.4. The projection $\mathbf{x}_s \mapsto \mathbf{x}_+$ is of marginal importance as regards local convergence. A sufficiently smooth projection only influences the constant factor C in Equ. (4.32). For motion design the projection is the orthogonal projection onto M^6 (discussed in Sect. 2.4) which meets this condition. Other projections discussed in [HP04] also meet this condition.

We proceed with the discussion of (4.32). Convergence of the algorithm depends on the constant

$$C_1 := |1 - s + sd\kappa_{\max}|.$$

We have linear convergence if $C_1 < 1$. The goal is a strategy for selecting the stepsize s such that C_1 becomes as small as possible.

The normal vector $\mathbf{n}$ has been chosen such that it points from $\mathbf{p}^*$ to $\mathbf{p}$. We have $d \geq 0$, but we cannot assume that $\kappa_{\max} \geq 0$. Two cases have to be discussed:

(a) $d\kappa_{\max} > 1$: Then $1 + s(d\kappa_{\max} - 1) > 1$, i.e., $C_1 > 1$, and no choice of s would guarantee convergence. Fortunately, this case does not arise in our setting: it can be shown to occur only if $\mathbf{p}^*$ is a local minimizer, but not a global one.

(b) $d\kappa_{\max} < 1$: In this case any choice of s with

$$0 < s < \frac{2}{|d\kappa_{\max} - 1|} \tag{4.33}$$

gives a constant $C_1 < 1$, i.e., *linear convergence.*

Note that $d = 0$ together with $s = 1$ yields the optimal situation $C_1 = 0$, i.e., a *quadratically convergent algorithm.*

4.2.3 Stepsize Selection

To show the idea behind our selection of stepsize, we first look at the simple case that Φ is a *curve* with curvature $\kappa = \kappa_{\max}$ (in the sense of Remark 4.3) at the footpoint $\mathbf{p}^*$. If we choose

$$s = \frac{1}{|d\kappa - 1|}, \tag{4.34}$$

we get $C_1 = 0$ in (4.32), which means *quadratic convergence.*

In practice, however, we do not know d and κ. In the following we show how to extend this idea from the curve to the surface case, and how to estimate the unknown values d and κ from data collected at previous iteration steps.

Remark 4.5. Equation (4.34) becomes intuitively clear for a planar curve, which for purposes of second order analysis is replaced by its osculating circle of radius $1/\kappa$ at the footpoint (Fig. 4.6). The point $\mathbf{x}_s$ is chosen as close as possible to $\mathbf{p}^*$, but still on the tangent T at $\mathbf{x}_c$. It is elementary to verify Equ. (4.34) for this choice of $\mathbf{x}_s$.

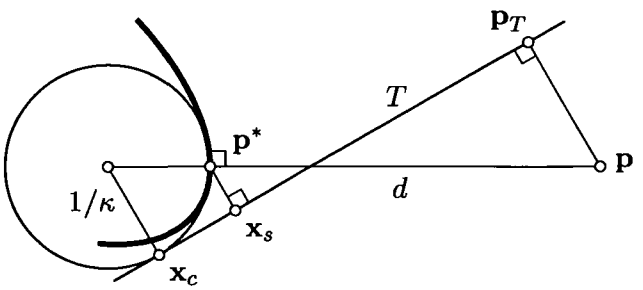


Figure 4.6: Visualization of stepsize selection.

In order to utilize the curve case for a general surface Φ, we consider a curve $\mathbf{c}_f$ on Φ, containing the individual iterates in our projected gradient algorithm. Computing $\mathbf{p}$'s footpoint in $\mathbf{c}_f$ yields the same point $\mathbf{p}^*$ as computing the footpoint on Φ. This means that we could just as well have applied the entire iteration to $\mathbf{c}_f$, which however is not known.

In view of Remark 4.3, we now estimate the stepsize s via developing the *footpoint cone* Γ_f, which connects $\mathbf{p}$ and $\mathbf{c}_f$. An approximation of this development in a neighborhood of the current iterate $\mathbf{x}_c$ can be computed from the mutual distances of the three points $\mathbf{x}_c$, the previous iterate $\mathbf{x}_-$, and $\mathbf{p}$.

Algorithm 3 (Stepsize). *Estimate the stepsize s via an approximate planar development of a part of the footpoint cone Γ_f (see Fig. 4.7):*

1. *Using the distances $\|\mathbf{p}_T - \mathbf{x}_c\|$, $\|\mathbf{p}_T - \mathbf{p}\|$, $\|\mathbf{x}_- - \mathbf{p}\|$, $\|\mathbf{x}_- - \mathbf{x}_c\|$ we develop the triangles $\mathbf{p}\mathbf{x}_c\mathbf{p}_T$ and $\mathbf{p}\mathbf{x}_c\mathbf{x}_-$ into a plane as shown by Fig. 4.7. Development is indicated by a tilde.*

2. *In the planar coordinate system of Fig. 4.7, let $\widetilde{\mathbf{x}}_-$ have coordinates (ξ, η). The circle k, which passes through $\widetilde{\mathbf{x}}_-$ and touches the line $\widetilde{\mathbf{x}}_c\widetilde{\mathbf{p}}_T$ at $\widetilde{\mathbf{x}}_c$, has center $\mathbf{m} = (0, \varrho)$ and radius $\varrho = (\xi^2 + \eta^2)/(2\xi)$. We let $\kappa := 1/\varrho$ and $d := |\ \|\widetilde{\mathbf{p}} - \mathbf{m}\| - |\varrho|\ |$. We plug these values into (4.34) to get the stepsize s.*

Note that there are two ways to attach the triangles $\widetilde{\mathbf{p}}\widetilde{\mathbf{x}}_c\widetilde{\mathbf{p}}_T$ and $\widetilde{\mathbf{p}}\widetilde{\mathbf{x}}_c\widetilde{\mathbf{x}}_-$ to each other. One way is shown by Fig. 4.7. The other possibility is obtained by reflecting the triangle $\widetilde{\mathbf{p}}\widetilde{\mathbf{x}}_c\widetilde{\mathbf{x}}_-$ in the line $\widetilde{\mathbf{p}}\widetilde{\mathbf{x}}_c$. Of course, we take the possibility which gives the smaller distortion of the spatial distance $\|\mathbf{p}_T - \mathbf{x}_-\|$.

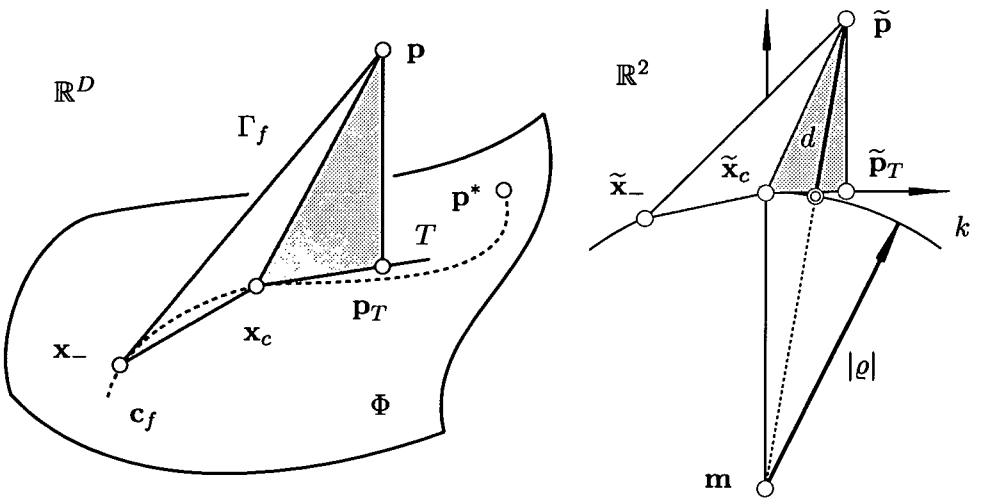


Figure 4.7: (Left) Distances in $\mathbb{R}^D$. (Right) Approximate planar development of a part of the footpoint cone.

4.2.4 Summary of the Optimization Algorithm

Algorithm 4 (Compute constrained minimizer). *Compute the minimizer* $\mathbf{p}^*$ *of a quadratic function with positive definite matrix* Q *(and unconstrained minimizer* $\mathbf{p}$*) under the constraint that* $\mathbf{p}^*$ *lies on a given* m*-dimensional surface* $\Phi \subset \mathbb{R}^D$*: Starting with an initial guess* $\mathbf{x}_0$ *for the minimizer, iteratively apply the following two steps.*

1. *Compute a basis* $\{\mathbf{c}_1, \ldots, \mathbf{c}_m\}$ *of* Φ*'s tangent space at the current iterate* $\mathbf{x}_c$*, and the Gramian matrix* $G = (g_{ij}) = (\langle \mathbf{c}_i, \mathbf{c}_j \rangle)$ *of that basis with respect to the inner product* $\langle \mathbf{x}, \mathbf{y} \rangle := \mathbf{x}^T \cdot Q \cdot \mathbf{y}$*. Further compute the vector* $\mathbf{r} = (r_1, \ldots, r_m)$ *with* $r_j = \langle \mathbf{p} - \mathbf{x}_c, \mathbf{c}_j \rangle$*. Define the tangent vector* $\mathbf{t} = \sum_i v_i \mathbf{c}_i$ *where* $\mathbf{v} = (v_1, \ldots, v_m)$ *is the solution of the linear system* $G \cdot \mathbf{v} = \mathbf{r}$*.*

2. *With distance computations based on the norm* $\|\mathbf{x} - \mathbf{y}\|^2 = \langle \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} \rangle$ *and the stepsize strategy of Algorithm 3, compute a stepsize* s *and the point* $\mathbf{x}_s = \mathbf{x}_c + s\mathbf{t}$*. Project* $\mathbf{x}_s$ *onto* Φ *to obtain the next iterate* $\mathbf{x}_+$*.*

The projection in step 2 depends on the chosen representation of Φ. In the motion design case the projection is the orthogonal projection summarized in Algorithm 1. In the very first iteration step, Algorithm 3 does not work and we must be content with a safely small stepsize s, whose validity can be tested with a standard strategy, e.g. the Armijo rule [Kel99]. Such a strategy should be added to each step anyway, in particular if large curvature changes unbalance our stepsize selection. In the applications shown later, such changes have been detected by means of the footpoint cone: we observe the change of curvature as iteration progresses. Later during iteration, points $\mathbf{x}_-$ and $\mathbf{x}_c$ will be too close to be used for curvature estimation; in such a case we revert to the last available valid estimate for κ.

The optimization algorithm presented here is very well suited for high dimensional applications, since it reduces curvature related computations (Algorithm 3) to the minimum which is necessary to achieve fast convergence.

Remark 4.6. The stepsize selection (4.33) can be seen as one step in a Newton iteration for computing $\mathbf{p}$'s footpoint on a circle, which approximates the footpoint curve $\mathbf{c}_f$ at $\mathbf{x}_c$. Other curves can be used for that purpose as well, e.g. a method proposed by Hartmann [Har99] for computing footpoints on parametric and implicit surfaces in $\mathbb{R}^3$ uses a certain parabola. The difference to our method is that the auxiliary curve used by [Har99] in each round of iteration does not use the approximate footpoints computed in previous rounds. It turns out that the performance of Hartmann's method is similar to ours, but it is stable only for small d.

4.2.5 Splines in Manifolds

Computing energy-minimizing curves on surfaces requires discretization. We describe a solution to the interpolation problem: The unknown curve $\mathbf{c}$ must pass through given points, so we assume that we are given parameter values

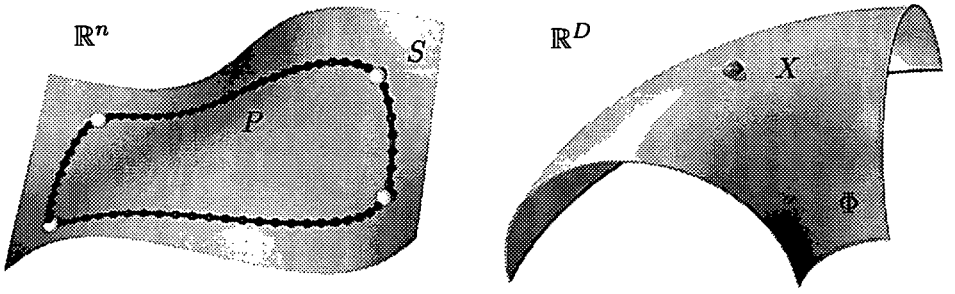


Figure 4.8: (Left) Polygon P on low-dimensional surface S. (Right) Polygon P viewed as one point X on the high-dimensional surface Φ.

$u_1, \ldots, u_N$ and points $\mathbf{p}_1, \ldots, \mathbf{p}_N$ on an m-dimensional surface S in $\mathbb{R}^n$, such that $\mathbf{c}(u_i) = \mathbf{p}_i$. The curve itself is represented by a point sequence which contains the points $\mathbf{p}_i$ (see Fig. 4.8 and Fig. 4.9):

$$\mathbf{p}_1, \mathbf{q}_{1,1}, \mathbf{q}_{1,2}, \cdots, \mathbf{q}_{1,M_1}, \mathbf{p}_2, \mathbf{q}_{2,1}, \cdots, \mathbf{p}_N,$$

with M_i new points in between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. We assume that evaluating the unknown curve $\mathbf{c}$ at parameter values $u_i + j(u_{i+1} - u_i)/(M_i + 1)$ yields $\mathbf{q}_{i,j}$. In our implementation, the parameter values u_i of the data points $\mathbf{p}_i$ are estimated such that Δu_i equals the arc length of the segment between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ on the initial curve. In case of heavy changes of the length of certain segments during the optimization, we recompute the parameters.

The new points $\mathbf{q}_{i,j}$ are the unknowns in the minimization problem. Their number equals $M := M_1 + \cdots + M_{N-1}$ for an open curve. We rename them $(\mathbf{x}_1, \ldots, \mathbf{x}_M)$ in order to establish the connection with the previous section and collect them in a new point $X \in \mathbb{R}^D$ with $D = Mn$. Thus a polygon $P = (\mathbf{x}_1, \ldots, \mathbf{x}_M) \in S$ in $\mathbb{R}^n$ corresponds to a point $X \in \Phi$ in $\mathbb{R}^D$, see Fig. 4.8. The constraint manifold Φ is the set of X's, such that the single $\mathbf{x}_k$'s are contained in the given surface S. Thus Φ is a surface whose dimension equals Mm, i.e., the number of unknowns M times the dimension m of the surface S.

Our optimization procedure (Algorithm 4 described in the previous section) employs Φ's tangent spaces and orthogonal projection onto them. It is therefore necessary to give bases of these spaces. Suppose that for each point $\mathbf{x}_k$, the original surface's S tangent space is spanned by vectors $\mathbf{c}_{k,l}$. Then the long vector $(0, \ldots, 0, \mathbf{c}_{k,l}, 0, \ldots, 0)$ with $\mathbf{c}_{k,l}$ in the place of $\mathbf{x}_k$ is tangent to Φ; and Φ's tangent space is spanned by those vectors, as k runs in $1, \ldots, M$ and l runs in $1, \ldots, n$.

The next step is to describe the quadratic function F. We use the difference of successive points as a discrete first derivative, and a difference of such differences as a discrete second derivative. By replacing integration by summation, any of the functionals E_1, E_2, E_t is thus converted into a quadratic function, and we directly apply Algorithm 4.

Note that this essentially means projecting gradients of an energy function, where the projection is defined via the orthogonality given by the energy it-

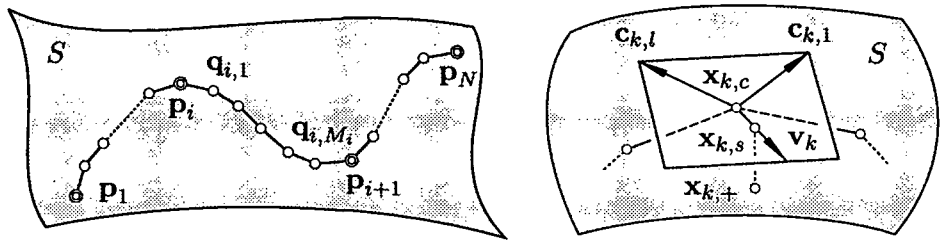


Figure 4.9: (Left) Notation for the discretized spline. (Right) Elementary view of an iteration step of the optimization algorithm.

self. This is a discrete version of a *Sobolev gradient*. The efficiency of Sobolev gradient methods for geometric optimization problems has been pointed out by Renka and Neuberger (see [Neu97]).

An elementary view of the algorithm is the following: We displace the (non fixed) vertices $\mathbf{x}_{k,c}$ of the polygon to the current iterate in their respective tangent spaces of S, such that the displaced polygon minimizes the chosen energy E; then these displacement vectors are scaled by the stepsize s, and the resulting points $\mathbf{x}_{k,s}$ are projected to points $\mathbf{x}_{k,+}$ of S (see Fig. 4.9).

Two possible ways to compute an *initial position* of the spline are:

1. Compute the unrestricted energy-minimizing spline through the given points $\mathbf{p}_i$ (with estimates for their parameter values u_i as suggested in the literature [HL93]) and project it onto the surface S. The projection has to be performed efficiently and therefore depends on the representation of S. For the motion design case, which is the topic of this thesis, an efficient projection is derived in Sect. 2.3 and summarized in Algorithm 1.

2. Use an approximation to a piecewise geodesic curve.

In the motion design case discussed in Chapter 5 we will use the CMD algorithm (Algorithm 2) of Chapter 3 to compute an initial motion.

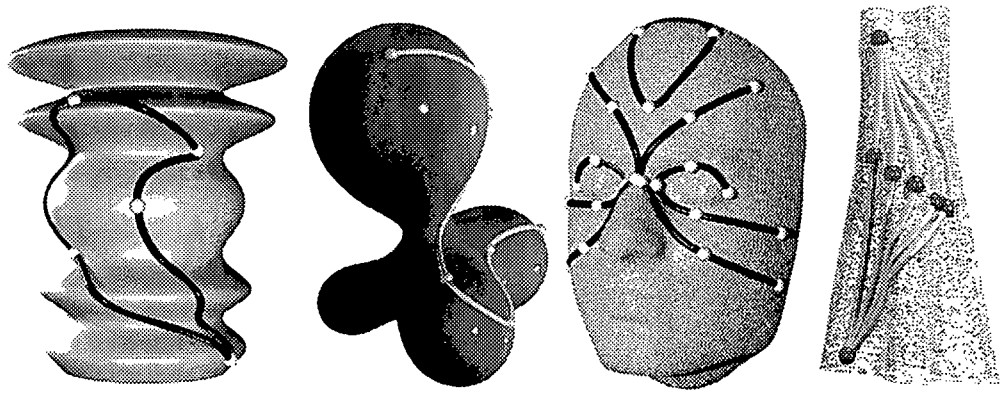


Figure 4.10: Energy-minimizing splines on a parametric surface, a level set surface, a triangle mesh, and a point set surface.

4.2.6 Examples of Energy-Minimizing Splines in Manifolds

The presented geometric optimization algorithm is sufficiently general so as to work for various representations of the surface S, and for any dimension and codimension of S. In [HP04] the computation of splines on two-dimensional surfaces $S \subset \mathbb{R}^3$ given in various representations is discussed. Figures 4.10 and 4.11 show examples of energy-minimizing splines on parametric surfaces, level set surfaces, triangle meshes, and point set surfaces. We will use the geometric optimization procedure (Algorithm 4) in Chapter 5 for variational motion design, and in Chapter 6 for variational curve and motion design in the presence of obstacles.

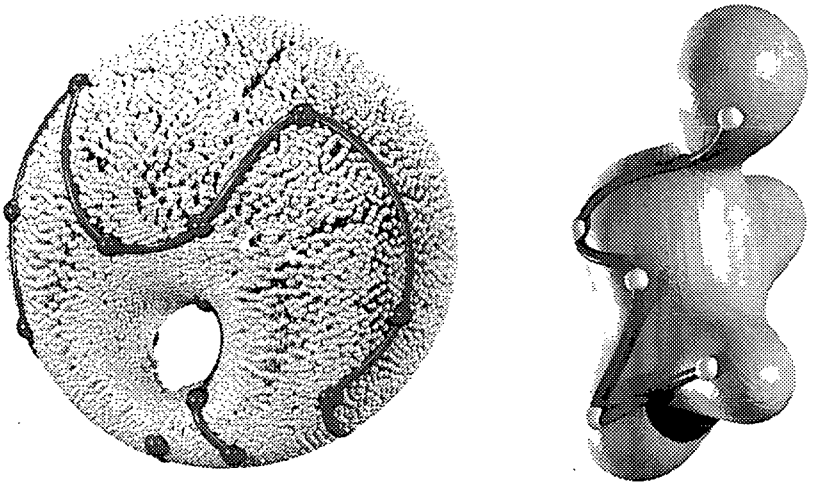


Figure 4.11: Curve minimizing E_2 on a point set surface (left), and level set (right).

Remark 4.7. The presented geometric optimization algorithm can also be used for *smoothing* or *fairing* of a curve $\mathbf{c}^0$ on a surface S, see Fig. 4.12: First fix some points of $\mathbf{c}^0$, then define $\mathbf{c}^0$ as initial position and run a few steps in the optimization to get a smoothed curve $\mathbf{c}$. Of course, this works for any representation of S and any of the possible applications. An example for *motion smoothing* is shown in Fig. 5.4 of Chapter 5.

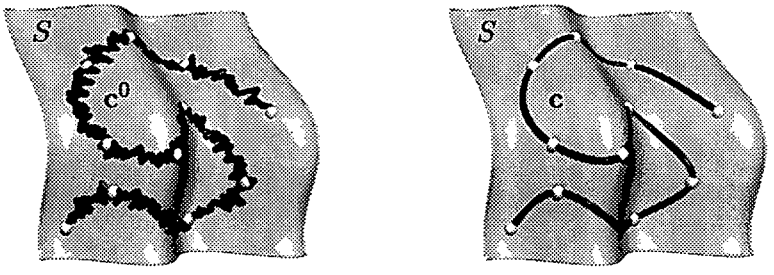


Figure 4.12: Smoothing of a curve on a surface.

Remark 4.8. Inserting only *one* point $\mathbf{q}_i \in S$ between two consecutive points $\mathbf{p}_i, \mathbf{p}_{i+1}$ can be seen as one round of *variational curve subdivision* in a manifold. This is a generalization of the variational subdivision algorithm [KS98] to surfaces. In Fig. 4.13 we compare two curves on a surface S interpolating 12 given points $\mathbf{p}_1, \ldots, \mathbf{p}_{12}$: a spline $\mathbf{c}_2$ minimizing E_2 computed with the algorithm of this chapter, and a variational subdivision curve $\mathbf{c}_v$ computed with the algorithm described in [HPR03]. For the curve $\mathbf{c}_2$ we use a polygon with 31 points between each two given points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. For the curve $\mathbf{c}_2$ the positions of all $341 = 31 \cdot 11$ points are optimized simultaneously. Using 5 subdivision steps to generate the curve $\mathbf{c}_v$ we also get 31 intermediate points between each two given ones $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. However, the positions of the points inserted in each subdivision step are fixed in the later subdivision steps. This explains the difference between the two curves $\mathbf{c}_2$, $\mathbf{c}_v$ and accounts for the better result obtained with Algorithm 4 described in the present chapter.

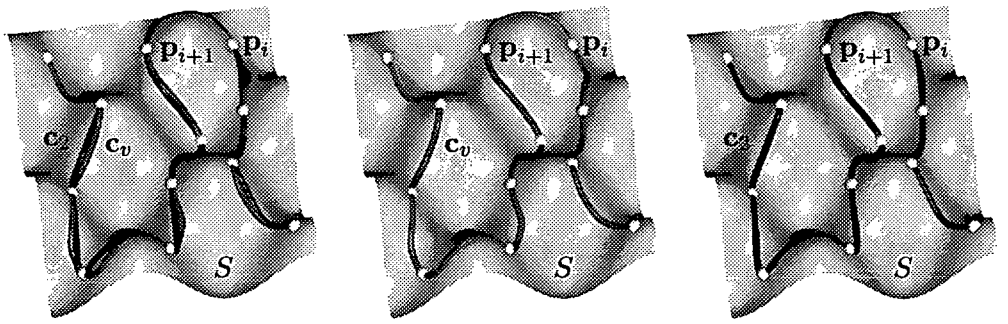


Figure 4.13: Comparison on a parametric surface S of a spline $\mathbf{c}_2$ computed with Algorithm 4 and minimizing E_2, to the curve $\mathbf{c}_v$ computed with the variational subdivision algorithm described in [HPR03].

Remark 4.9. The presented geometric optimization algorithm can also be used for *feature sensitive design* of curves on surfaces. One can use the concept of *image manifolds* [KMS00], that makes it possible to develop image sensitive drawing tools. Such an approach includes images defined on surfaces. Instead of color or texture information we use the surface normals as the image. Then design in the image manifold means feature-sensitive design in the original surface. This type of image manifold has been employed in [PSH+04] for the development of feature sensitive morphological operators on surfaces.

Figure 4.14 compares geodesics connecting two points on a surface, computed in the standard surface metric and in the feature sensitive metric. The latter geodesics follow the features of the surface, while the former ones are just the shortest connecting curves of the two given points. Figure 4.15 compares two spline curves interpolating the same 6 input points on a parametric surface. Again the curve computed in the feature sensitive metric nicely follows the features of the surface.

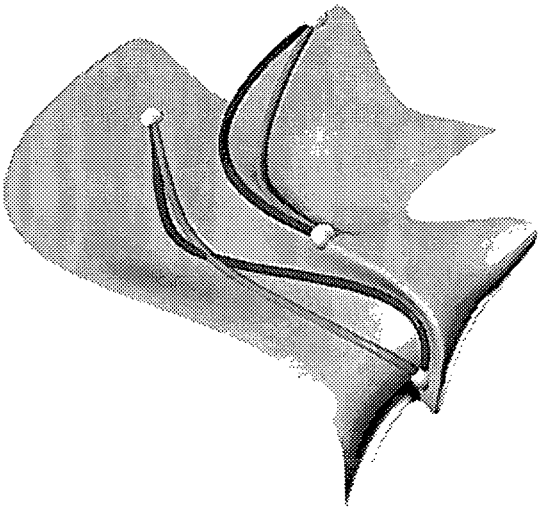


Figure 4.14: Adding feature sensitivity to energy-minimizing curve design on surfaces: Comparison of geodesics connecting two points computed in the standard metric and in the feature sensitive metric.

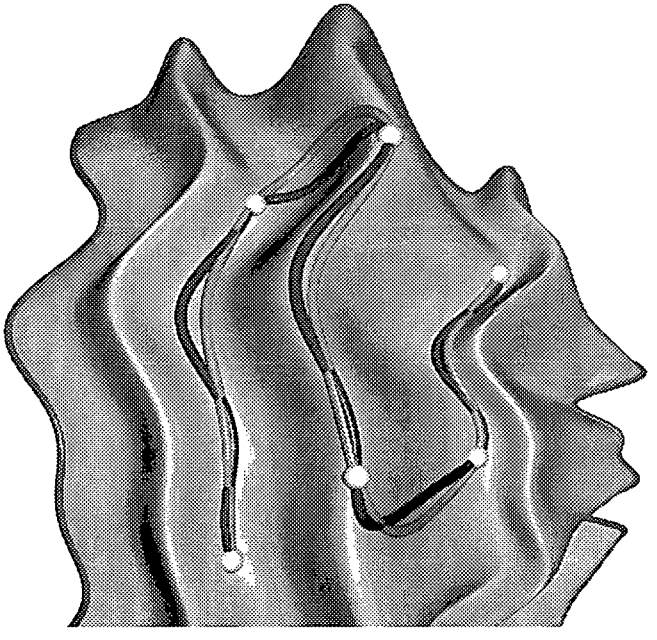


Figure 4.15: Adding feature sensitivity to energy-minimizing curve design on surfaces: Comparison of spline on a surface minimizing E_2 computed in the standard metric and in the feature sensitive metric.

Chapter 5

Variational Motion Design

In this chapter we study the following motion design problem: Given are N positions $\mathcal{B}(u_i)$ of a moving body $\mathcal{B}^0 \subset \mathbb{R}^3$ at time instances u_i. We want to compute a smooth rigid body motion $\mathcal{B}(u)$ which interpolates the given key or control positions $\mathcal{B}(u_i)$ and minimizes a certain energy. The energy is expressed with help of the energies of trajectories of points on the moving body.

We view a Euclidean one-parameter motion as a curve on the manifold $M^6 \subset E^{12}$. Recall that the CMD algorithm of Chapter 3 does not preserve minimum energy properties. Thus, we are now dealing with the problem of constructing energy-minimizing interpolating spline curves on M^6. In this way, we are computing energy-minimizing Euclidean rigid body motions in $\mathbb{R}^3$ which interpolate given positions. For contributions to variational motion design in the literature (via the quaternion 3-sphere) see [BCGH92, PR97, RB97].

In the present chapter we apply the geometric optimization procedure (Algorithm 4) introduced in Chapter 4 to the variational design of rigid body motions. We abbreviate this motion design algorithm as the VMD algorithm (Variational Motion Design). To find an initial motion for the iterative optimization algorithm, the CMD algorithm of Chapter 3 is employed with an energy-minimizing curve scheme, such as interpolating C^2 cubic splines.

Section 5.1 is devoted to the characterization of energy-minimizing motions, which are analogous to known energy-minimizing spline curves, such as C^2 cubic splines or splines in tension. We do this by using the theoretic results obtained in Chapter 4 and interpreting them in the motion case.

Section 5.2 discusses computational aspects. The numerical solution is a special case of the computation of splines in manifolds presented in Chapter 4. We present illustrative examples of energy-minimizing motions.

5.1 Characterizing Energy-Minimizing Motions

Variational motion design is seen as curve design with energy-minimizing splines on $M^6 \subset E^{12}$, using the metric (2.2). For motions, the meaning of minimizing one of the functionals E_1, E_2, or E_t is that the total energy of the sample point trajectories is minimized. It makes sense to base motion design on trajectories of points on the moving body. We view this as an advantage over the known

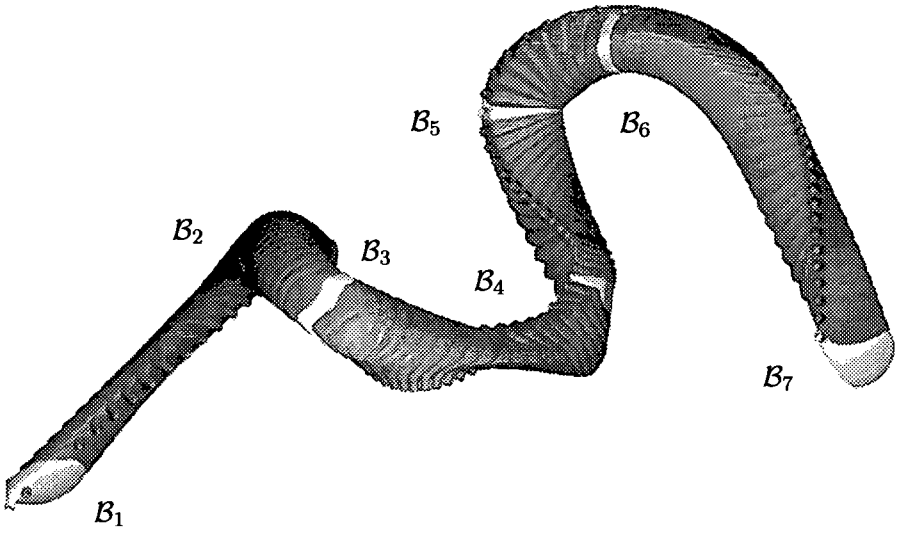


Figure 5.1: Variational motion interpolating 7 positions and minimizing E_2.

purely intrinsic formulations, which are neglecting shape and mass properties of the moving body.

For a geometric characterization of the motions which are computed in this way, we use the concept of a balanced force system from statics, see Section 2.3. A system of forces $\mathbf{f}_i$, attached to points $\mathbf{p}_i$, is in balance, if both, the sum of force vectors and the sum of moment vectors, vanish, i.e., $\sum_i \mathbf{f}_i = 0$, and $\sum_i \mathbf{p}_i \times \mathbf{f}_i = 0$. At an arbitrary time instant u of a sufficiently smooth motion, the k-th derivative vectors at the sample point positions define the k-th derivative force system $\mathbf{S}_k$.

We are viewing u_i as given time instances, at which given positions $\mathcal{B}(u_i)$ of the rigid body, i.e., points $\mathsf{A}_i \in M^6$, have to be interpolated. The energy (4.1) of a curve $\mathsf{X}(u) \subset E^{12}$ uses the norm induced by (2.2) and thus it may also be interpreted as sum of the corresponding energies (L^2 norms of the second derivatives) of the sample point trajectories.

According to Theorem 4, the solution curve $\mathsf{C}(u) \subset M^6$ has 4-th derivative vectors $\mathsf{C}^{(4)}(u)$, which are orthogonal to M^6. The force system $\mathsf{S}_4(u)$ induced by $\mathsf{C}^{(4)}(u)$ consists of the fourth derivative vectors of the sample point trajectories at a given instant u. Using Theorems 1 and 4, we obtain the following result.

Theorem 6. *Consider N input positions, corresponding time instances u_i and differentiability assumptions as in Theorem 4. Then, an interpolating motion minimizing the sum of energies (L^2 norms of the second derivatives) of the sample point trajectories is characterized as follows. The motion is C^2, has at each time instant $u \neq u_i$ a balanced 4-th derivative force system $\mathsf{S}_4(u)$, and at the end positions balanced force systems $\mathsf{S}_2(u_1), \mathsf{S}_2(u_N)$ of second derivatives. In particular, the trajectory of the barycenter of the sample points is an interpolating cubic C^2 spline.*

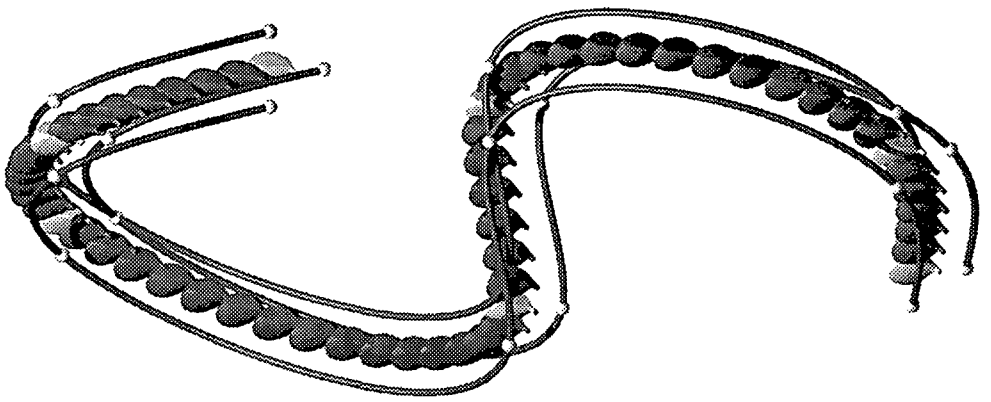


Figure 5.2: A motion with only C^2 cubic spline trajectories is translational.

The result on the trajectory of the barycenter follows from the vanishing of the force components $\mathbf{s}_k$ of the involved k-th derivative systems $\mathsf{S}_k = (\mathbf{s}_k, \bar{\mathbf{s}}_k)$. We see that these motions somehow balance the deviations of the point trajectories from C^2 cubic splines; there the 4-th derivatives would vanish everywhere. Note that a motion with only C^2 cubic spline trajectories must be translational (the image curve lies in a 3-dimensional affine subspace contained in M^6 and is a cubic spline itself), see Fig. 5.2.

It is also quite natural that the moving body via its mass distribution (barycenter and inertia tensor) enters the variational formulation and the interpretation of the solution. That the present approach is natural from the viewpoint of mechanics and kinematics, is also nicely seen if we replace the energy in (4.1) by the L^2 norm of the first derivative, see (4.26), $E_1(\mathbf{x}) = \int_{u_1}^{u_N} \|\dot{\mathbf{x}}(u)\|^2 du$. Moreover, we just prescribe the two end positions. Then, one finds with a known counterpart of Theorem 4 a curve $\mathsf{C}(u) \subset M^6$, whose second derivative vectors $\ddot{\mathsf{C}}$ are orthogonal to M^6. From this we conclude immediately that *the minimizers of E_1 are geodesics on M^6 in a scaled arc length parameterization, i.e., $\|\dot{\mathsf{C}}\| = const$.* This proves the following theorem.

Theorem 7. *Motions which join two given positions and arise from minimization of (4.26) correspond to geodesics on M^6, parameterized by a constant multiple of arc length. At any time instant, such a geodesic motion possesses a balanced force system $\mathsf{S}_2(u)$ of second derivatives. The trajectory of the barycenter of the sample point set on the moving body is a straight line traced with constant speed. These motions are free motions of a body in the sense of mechanics.*

Figure 5.3 compares shortest motions between two positions $\mathcal{B}_1$ and $\mathcal{B}_2$ in the sense of the metric (2.2) and in the quaternion sense. The former is the free motion of a rigid body and tends to generate shorter paths for points close to the moving object.

Note that a helical motion is a geodesic motion between two positions $\mathcal{B}(u_1)$ and $\mathcal{B}(u_2)$, if the positions of the inertia ellipsoids share a common axis, and if this is the axis of the helical motion joining the two positions. For more results on free motions, we refer to [Arn89].

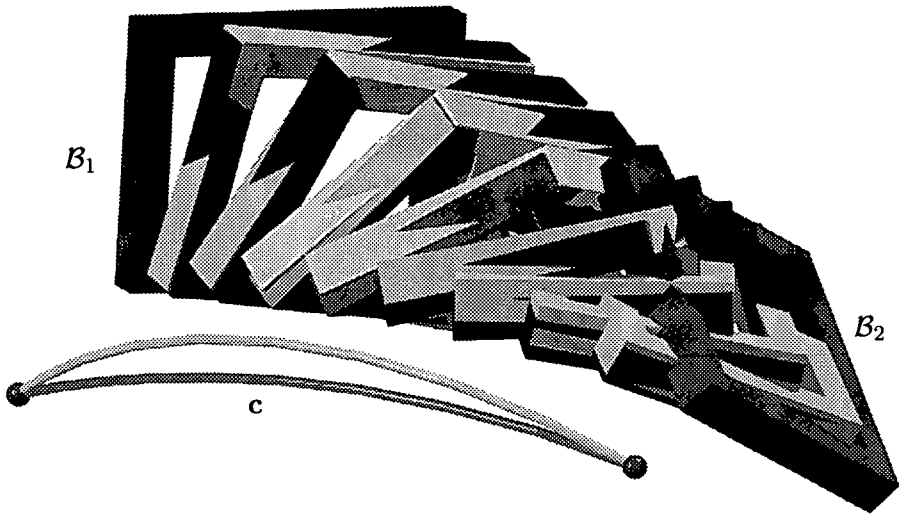


Figure 5.3: Comparing shortest motions between two positions $\mathcal{B}_1$ and $\mathcal{B}_2$: (blue) in the sense of the metric (2.2) and (orange) in the quaternion sense. The former is the free motion of a rigid body and tends to generate shorter paths $\mathbf{c}$ for points close to the moving object.

By minimization of a combination $E_t := E_2 + wE_1$ with a constant positive factor w one obtains the counterparts of *splines in tension* for motions. These are characterized as in Theorem 6, but instead of a balanced 4-th derivative force system, the linearly combined force system $\mathsf{S}_4 - w\mathsf{S}_2$ is in balance. A proof follows from results in Chapter 4. Various other energy-minimizing spline types such as smoothing splines or quintic splines (minimizing the L^2 norm of the third derivative) can be transferred to motion design within the present setting.

Note that the functionals we are considering are also optimizing the parameterization. This is useful for motion design where the time u as parameter plays an important role for applications.

5.2 Computing Energy-Minimizing Motions

Although the presented spline motions possess nice geometric characterizations, the problem is still nonlinear and does not admit an explicit solution as in the unrestricted curve design case. Thus, we have to use numerical algorithms based on a discretization. Geometrically this means that we replace the curve on M^6 by a sufficiently dense polygon with vertices in M^6. These vertices represent positions of the motion at discrete time instances.

Recall from Chapter 4 that the numerical computation of energy-minimizing splines on surfaces is based on a discretization. This means that the final motion is discretely resolved by a certain number of positions, including the input positions. The optimization algorithm in each step computes a tangent vector $s\mathbf{t}$ and requires projecting $\mathbf{x}_s = \mathbf{x}_c + s\mathbf{t}$ onto Φ. Actually both $\mathbf{x}_c$ and $\mathbf{x}_s$ are sequences of positions, and the tangent vector $\mathbf{t}$ is a sequence of tangent

vectors to M^6. We project the sequence $\mathbf{x}_s$ by projecting each single position orthogonally onto M^6. The projection of such a single position is performed using Algorithm 1 of Sect. 2.4.

A basis of the 6-dimensional tangent space at a point $\mathsf{A} = (\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \in M^6 \subset E^{12}$ is given by the six vectors $\mathsf{T}_1, \ldots, \mathsf{T}_6 \in E^{12}$ derived from (2.10). Let $\mathbf{a}_i = (a_{i1}, a_{i2}, a_{i3})^T \in \mathbb{R}^3$ for $i = 0, 1, 2, 3$, then we get

$$\begin{aligned}
\mathsf{T}_1 &= (0, -a_{03}, a_{02}, 0, -a_{13}, a_{12}, 0, -a_{23}, a_{22}, 0, -a_{33}, a_{32})^T, \\
\mathsf{T}_2 &= (a_{03}, 0, -a_{01}, a_{13}, 0, -a_{11}, a_{23}, 0, -a_{21}, a_{33}, 0, -a_{31})^T, \\
\mathsf{T}_3 &= (-a_{02}, a_{01}, 0, -a_{12}, a_{11}, 0, -a_{22}, a_{21}, 0, -a_{32}, a_{31}, 0)^T, \\
\mathsf{T}_4 &= (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T, \\
\mathsf{T}_5 &= (0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T, \\
\mathsf{T}_6 &= (0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T.
\end{aligned}$$

The spline property of the barycenter trajectory implies that motion computation in E^{12} can be decomposed into the computation of this special trajectory (in $\mathbb{R}^3$) and the computation of the rotational part (on the 3-dimensional manifold M^3 in $\mathbb{R}^9$, see Remark 2.1). A basis of the 3-dimensional tangent space at a point $\mathsf{A} = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \in M^3 \subset \mathbb{R}^9$ is given by the three vectors $\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3 \in \mathbb{R}^9$,

$$\begin{aligned}
\mathsf{T}_1 &= (0, -a_{13}, a_{12}, 0, -a_{23}, a_{22}, 0, -a_{33}, a_{32})^T, \\
\mathsf{T}_2 &= (a_{13}, 0, -a_{11}, a_{23}, 0, -a_{21}, a_{33}, 0, -a_{31})^T, \\
\mathsf{T}_3 &= (-a_{12}, a_{11}, 0, -a_{22}, a_{21}, 0, -a_{32}, a_{31}, 0)^T.
\end{aligned}$$

Note that we use the same notation T_i for the basis vectors of M^3 and M^6. This does not lead to any confusion since we will always mention in which manifold we currently work. We summarize variational motion design in the following algorithm.

Algorithm 5 (VMD). *The algorithm employs the following steps:*

1. *Choose an energy functional $E \in \{E_1, E_2, E_t\}$.*

2. *Run the CMD algorithm (Algorithm 2) with a curve scheme minimizing E to get an initial Euclidean motion. This already determines the path of the barycenter (because of the spline property mentioned above).*

3. *Employ the geometric optimization procedure of Chapter 4 (Algorithm 4) to compute an energy-minimizing spline curve on the manifold M^3 (M^6).*

The presented motion design algorithm can also be used for *motion smoothing*. An example of the disturbed motion and the corresponding smoothed motion minimizing E_2 is shown in Fig. 5.4. The smoothed motion is obtained by first fixing some input positions and then applying a few steps of the iterative optimization algorithm. Note that the smoothed motion is determined by the chosen fixed positions.

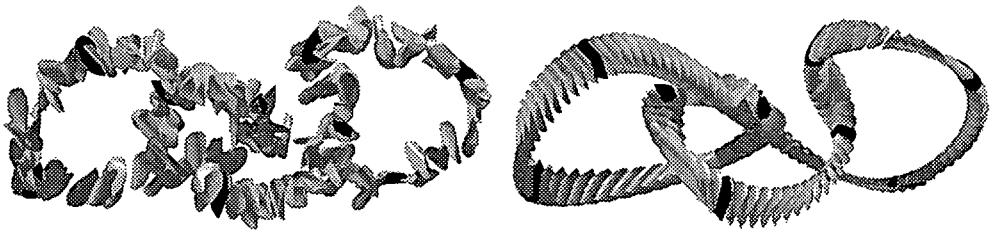


Figure 5.4: (Left) Noisy input motion. (Right) Smoothed motion.

Remark 5.1. Another computational approach directly uses the characterization of the motions in a discretized way. For example, a motion minimizing E^2 has a balanced 4-th central difference screw at any instance in the time discretization. This results in a system of nonlinear equations, which can be solved with a Newton iteration. In view of the bad global behavior of a pure Newton algorithm, we used the quasi-Newton approach (Algorithm 4) introduced in Sect. 4.2.

5.3 Examples of Energy-Minimizing Motions

We implemented our algorithm for variational motion design in *Matlab* and tested it on several examples using a PC with 1.8GHz. Figure 5.5 compares two Euclidean rigid body motions $\mathcal{B}_C(u)$ (computed using the CMD algorithm) and $\mathcal{B}_V(u)$ (computed using the VMD algorithm). As curve design algorithm we used C^2 cubic splines, i.e., the energy functional E_2. The resulting Euclidean motion $\mathcal{B}_C(u)$ of the CMD algorithm is the initial Euclidean motion for the iterative VMD algorithm. After a few iteration steps we get the final energy-minimizing motion $\mathcal{B}_V(u)$.

The affine motion $\mathcal{A}(u)$ computed with the CMD algorithm is the unconstrained energy-minimizing motion interpreted as a point $\mathbf{p}$ in high-dimensional space. The Euclidean motions $\mathcal{B}_C(u)$ and $\mathcal{B}_V(u)$ are seen as points $\mathbf{p}_0$ and $\mathbf{p}^*$, respectively, in the high-dimensional manifold Φ. The point $\mathbf{p}^*$ is the footpoint of $\mathbf{p}$ onto Φ in the Euclidean norm described in Sect. 4.2.1. If we use the function F of Equ. (4.31) to measure the distances $F(\mathbf{p}^*) = \|\mathbf{p}^* - \mathbf{p}\| = 0.2284$ and $F(\mathbf{p}_0) = \|\mathbf{p}_0 - \mathbf{p}\| = 0.8034$ we see, that in this example the energy of the motion $\mathcal{B}_V(u)$ is only about a quarter of the energy of $\mathcal{B}_C(u)$.

The energy-minimizing motion $\mathcal{B}_V(u)$ is also visually more pleasing. Furthermore, if we study animations of our motions we see the following effect: If two adjacent input positions of the moving body $\mathcal{B}$ have almost opposite orientation, then the CMD algorithm tends to deal with that via a rather sudden movement in the 'middle' between the two adjacent positions. The VMD algorithm distributes this huge change in orientation over the whole transition area and thus generates visually more pleasant and practically more useful motions.

The example of Fig. 5.6 has been computed using the energy E_2. We have chosen the input situation such that two adjacent positions $\mathcal{B}_2$ and $\mathcal{B}_3$ are much closer to each other than the remaining adjacent ones. Here we see the advantage of choosing the number of intermediate positions that are inserted

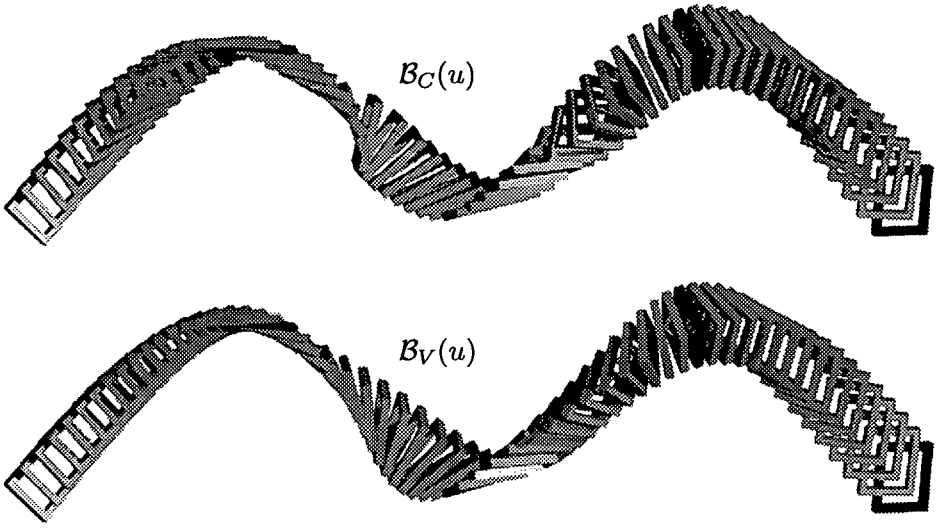


Figure 5.5: Rigid body motions interpolating the same input positions computed using energy E_2: (top) CMD algorithm; (bottom) VMD algorithm.

between two adjacent input positions $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ in dependence of the spatial position. In this example the numbers of intermediate positions between $\mathcal{B}_1$ and $\mathcal{B}_2$, $\mathcal{B}_2$ and $\mathcal{B}_3$, $\mathcal{B}_3$ and $\mathcal{B}_4$, $\mathcal{B}_4$ and $\mathcal{B}_5$ are 46, 5, 36, and 25, respectively.

Figure 5.7 shows an example where we have employed the VMD algorithm (as curve design on $M^6 \subset E^{12}$) with two different initial Euclidean motions. For the variational motion shown in Fig. 5.7 (left) we used the result of the CMD algorithm as initial value, for the variational motion shown in Fig. 5.7 (right) we used the result of the CMD algorithm and disturbed the path of the barycenter by some random error. In this example, starting from the CMD output leads to the variational motion with a lower energy level.

The example shown in Fig. 5.8 compares energy-minimizing cyclic motions that interpolate the same input positions and minimize E_2, E_t, and E_1, respectively. It further shows the paths $\mathbf{c}_2$, $\mathbf{c}_t$, and $\mathbf{c}_1$ of a selected point during those motions. The comparison of $\mathbf{c}_2$ to the C^2 cubic spline $\mathbf{c}$ shows that the two curves differ from each other by different amounts, depending on how much the motion deviates from a pure translational motion minimizing E_2 (for which all point paths would be cubic splines $\mathbf{c}$).

The example shown in Fig. 5.9 compares energy-minimizing cyclic motions that interpolate the same input positions and minimize E_2, E_t, and E_1, respectively. This example illustrates how the weight w in the formulation of the energy E_t can be used by a designer to influence the shape of a rigid body motion. By adjusting w one can design to the same input positions the whole range of rigid body motions from minimizing energy E_2 till minimizing energy E_1 (for a large value of w).

Figure 5.10 shows 18 different views of the energy-minimizing motion depicted in Fig. 5.9 (top). Figure 5.11 shows 485 positions of a rigid body motion computed with the VMD algorithm that interpolates 38 input positions.

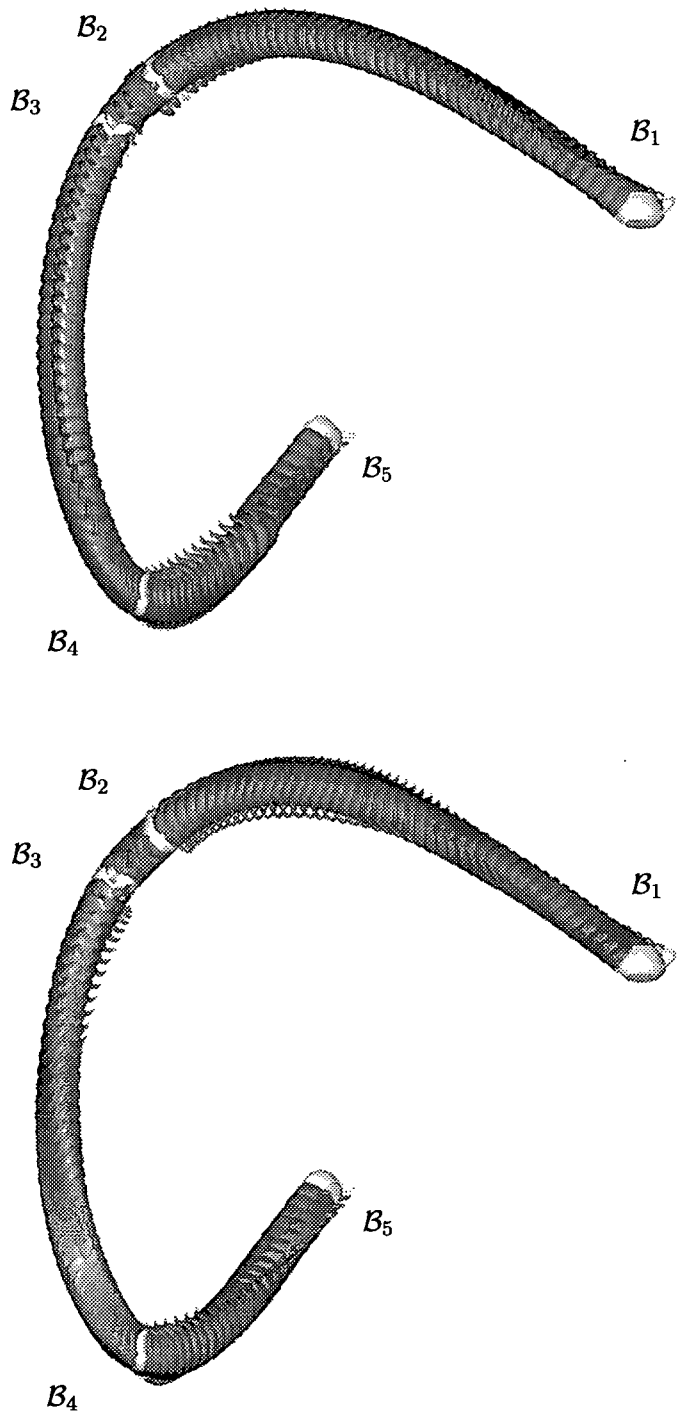


Figure 5.6: Rigid body motions interpolating the same input positions computed using energy E_2: (top) CMD algorithm; (bottom) VMD algorithm.

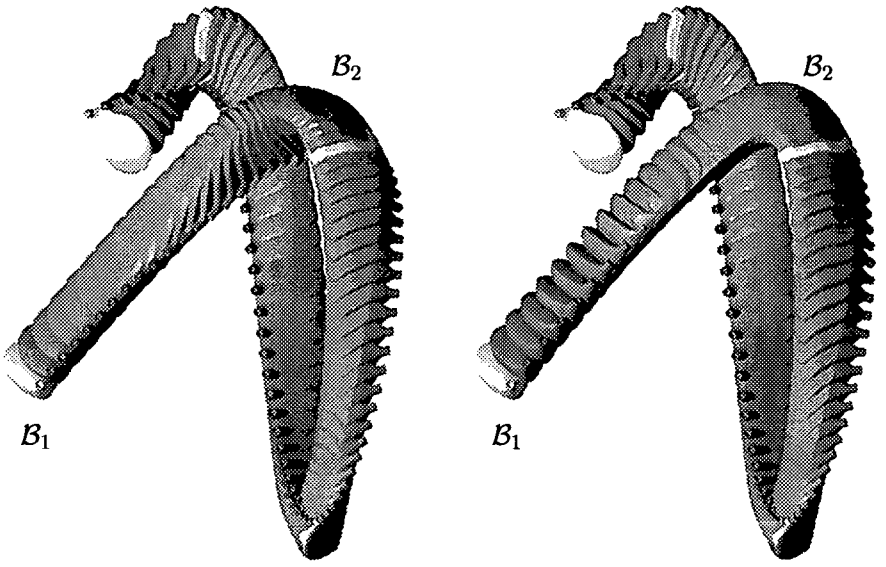


Figure 5.7: Two Euclidean motions computed with the VMD algorithm. Using different initial motions the algorithm reached two different minima (note the difference of the motions between $\mathcal{B}_1$ and $\mathcal{B}_2$) with nearly the same energy level.

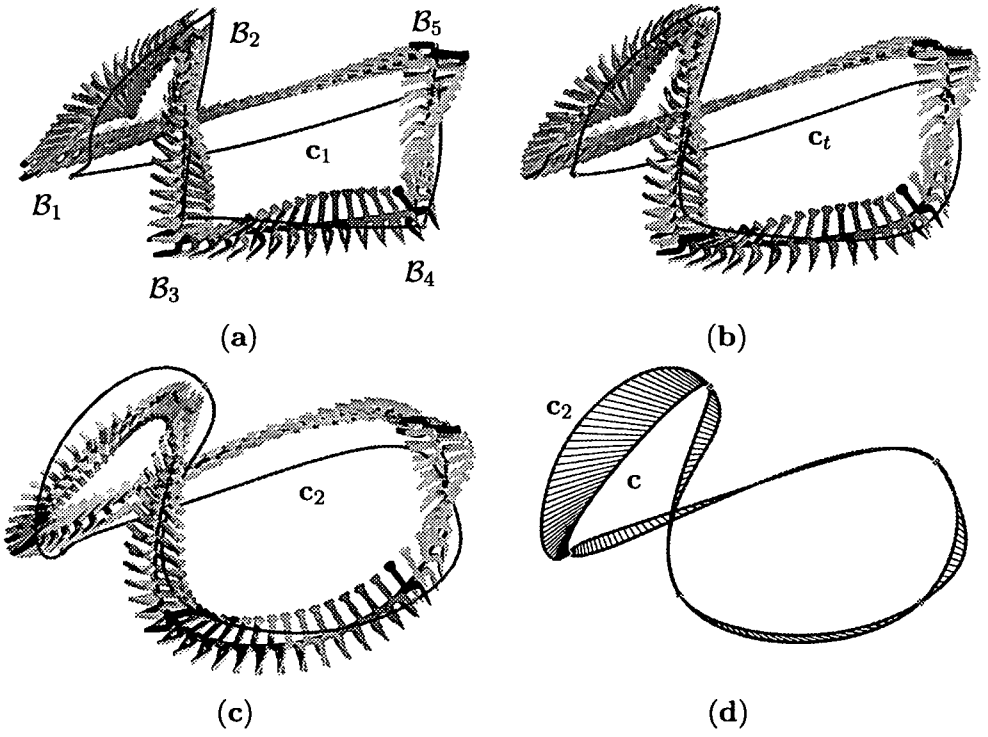


Figure 5.8: Cyclic motion of a robot gripper interpolating 5 positions $\mathcal{B}_1, \ldots, \mathcal{B}_5$ and one point path: Motion minimizing (**a**) E_1, (**b**) E_t with $w = 0.05$, (**c**) E_2. (**d**) Point path $\mathbf{c}_2$ compared to cubic spline $\mathbf{c}$ interpolating the same points.

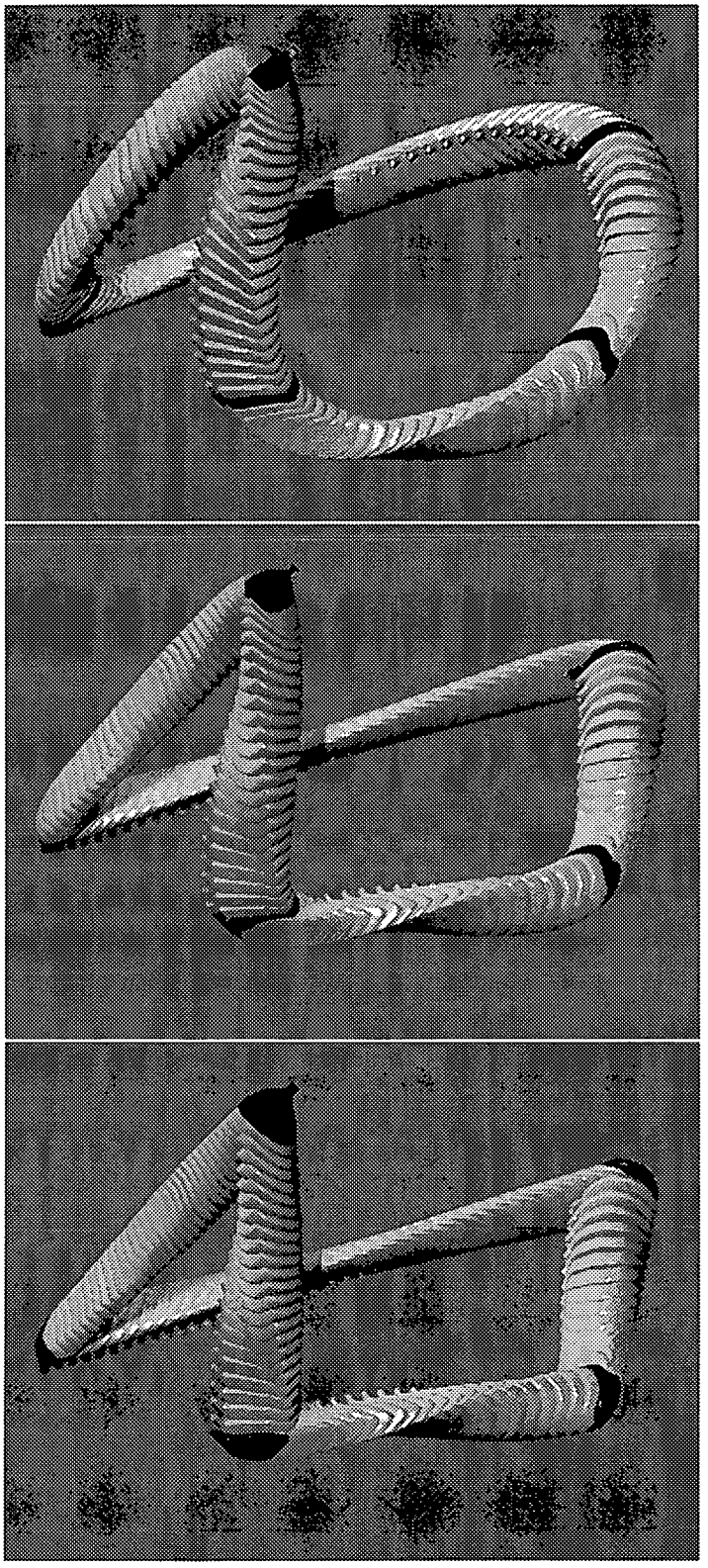


Figure 5.9: Energy minimizing cyclic motions of a golf club model: (top) E_2, (middle) E_t, (bottom) E_1.

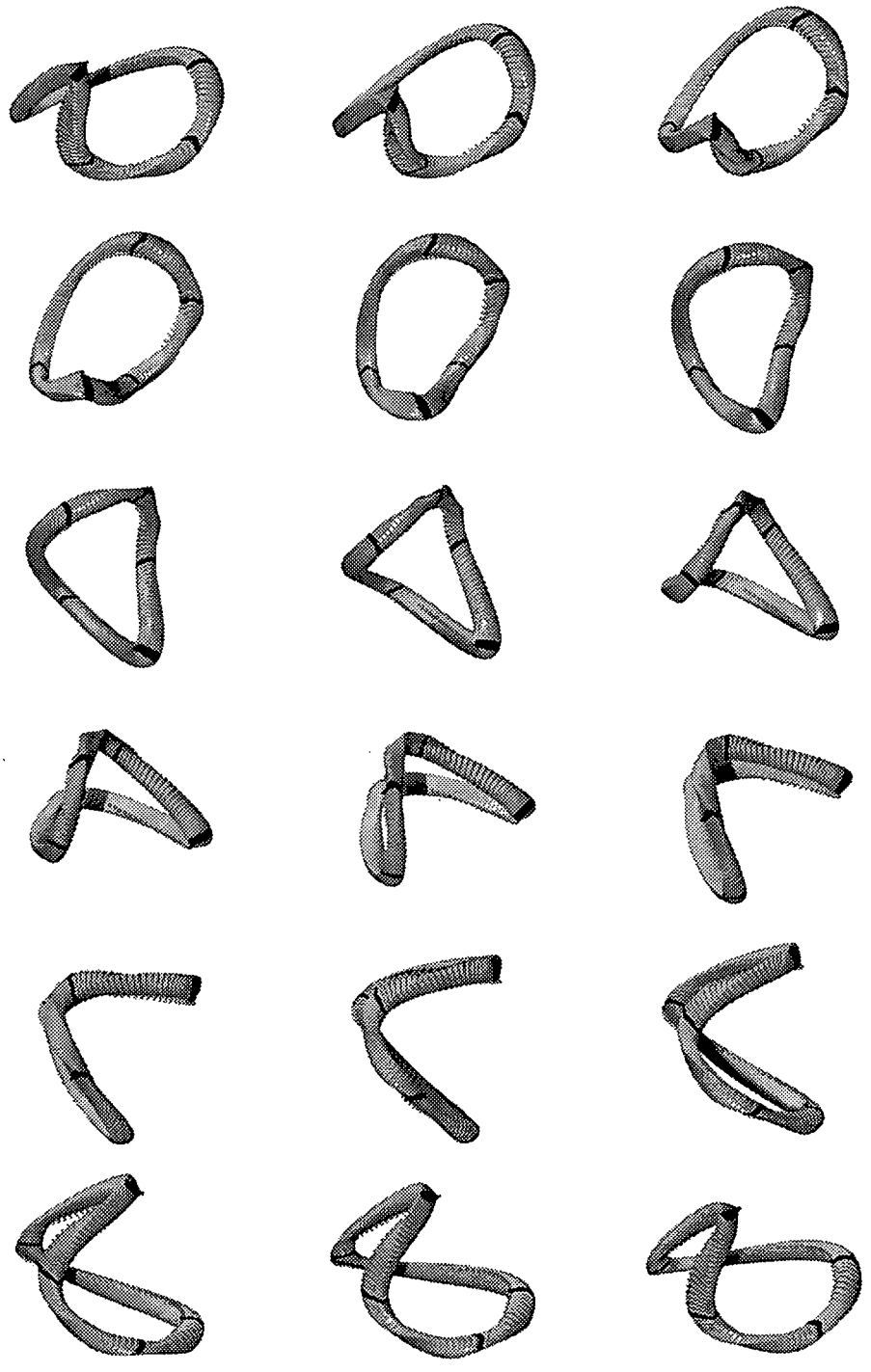


Figure 5.10: Different views of the same Euclidean motion of a golf club model interpolating 5 input positions and minimizing the cubic spline energy E_2.

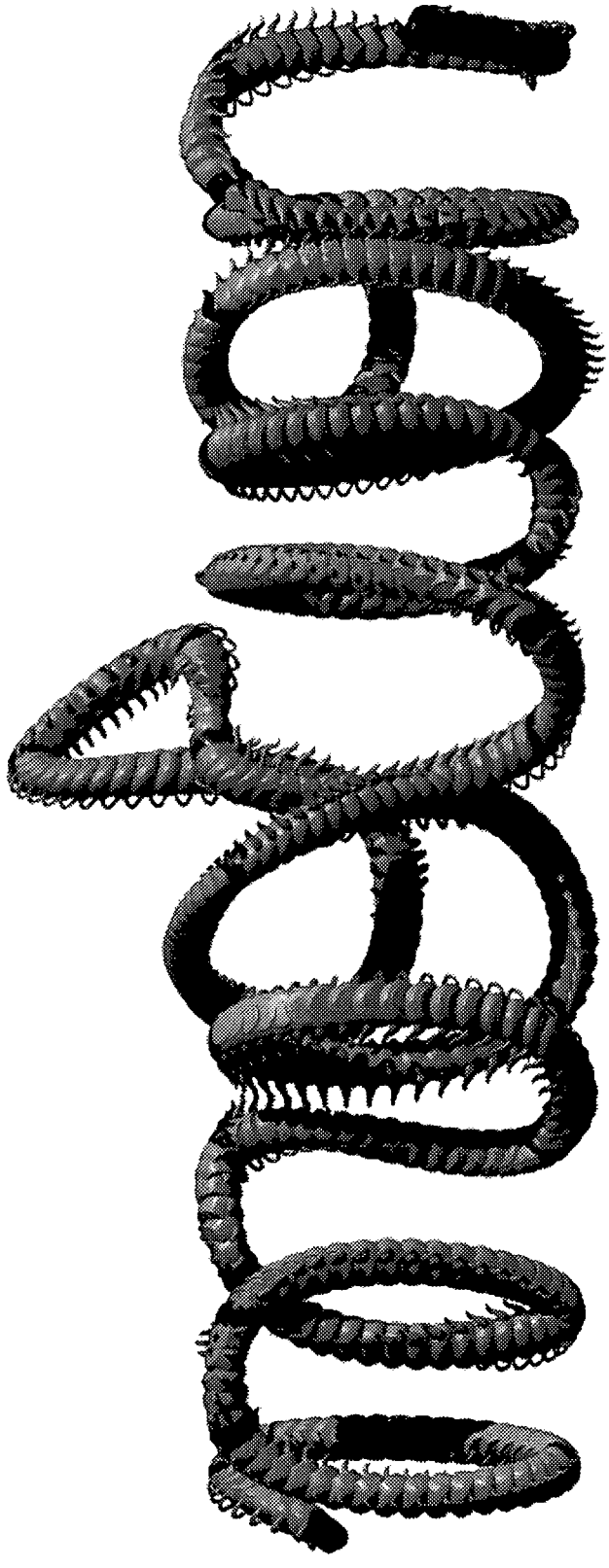


Figure 5.11: Euclidean motion computed with the VMD algorithm interpolating 38 input positions.

Chapter 6

Variational Motion Design in the Presence of Obstacles

In this Chapter we study in Section 6.1 variational curve design in the presence of obstacles and present examples for obstacle avoiding curves in $\mathbb{R}^2$ and $\mathbb{R}^3$. Then we continue in Section 6.2 with a conservative solution for variational motion design in the presence of obstacles. In Section 6.3 we discuss a second algorithm for variational motion design in the presence of obstacles which fully employs the available degrees of freedom. We illustrate the algorithms at hand of several examples.

6.1 Variational Curve Design in the Presence of Obstacles

Although this problem has a variety of practical applications, there are not many contributions dealing with it. We point to work on interpolation with cubic spline functions under linear inequality constraints [OO88], CAD related work on constrained curve design without energy minimization [MOW03], and in particular to Bohl's contribution to splines on parametric surfaces [Boh99]. We will see that we can get rid of the obstacle constraints by introducing an appropriate unbounded auxiliary surface M and working with that.

Let us start with curve design in the plane $\mathbb{R}^2$. As a geometric interpretation of the *barrier method* from constrained optimization [Fle87], we embed $\mathbb{R}^2$ as the plane $z = 0$ into $\mathbb{R}^3$ and remove the interior of each obstacle $\mathcal{O}_i$. We now attach, to each obstacle boundary $\mathbf{b}_i$, a cylindrical surface with z-parallel rulings and smooth out the edge along $\mathbf{b}_i$ which is generated by this procedure. This results in a smooth surface M. Given input points $\mathbf{p}_j$, which do not lie in any obstacle, are lifted onto M in z-direction; only points near obstacle boundaries will be changed by this lifting.

Now we design an interpolating curve $\bar{\mathbf{c}}$ on M; its projection onto $z = 0$ is the desired curve. During the optimization, M is not kept fixed. The blending radius is reduced so that it tends to zero, see Fig. 6.1. Therefore, the final curve $\mathbf{c}$ is a minimizer of the chosen energy under the given constraints. The general results of Chapter 4 imply that $\mathbf{c}$ *is* C^2 *and piecewise cubic where it does not*

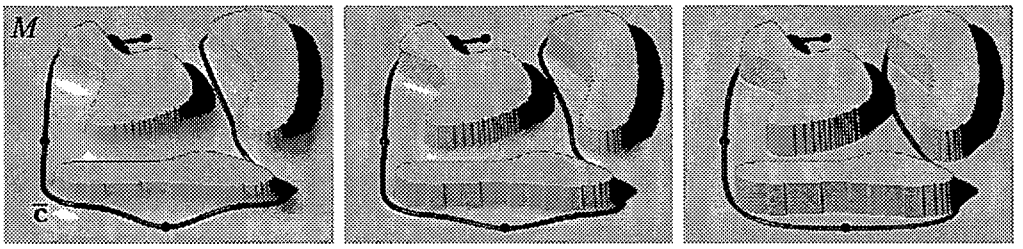


Figure 6.1: Spline computation on barrier surface with decreasing rounding radius during the iteration.

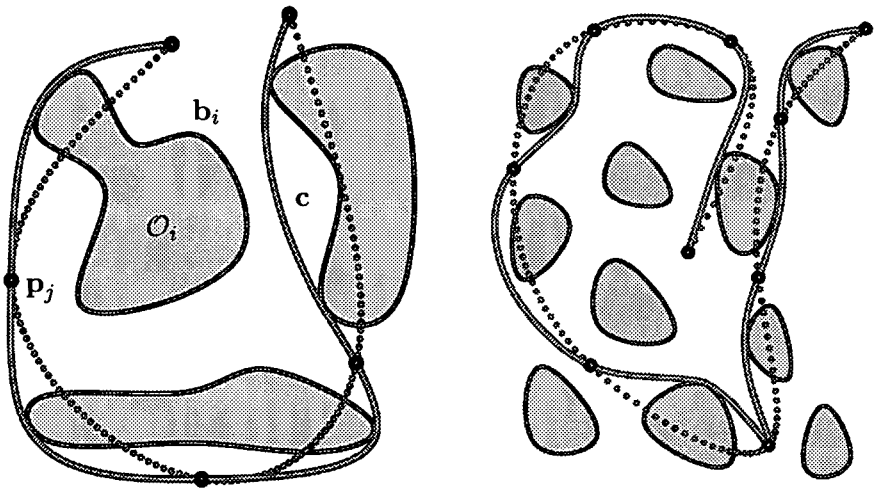


Figure 6.2: Energy-minimizing planar spline curves: The dotted curve is computed without consideration of the obstacles, the solid curve avoids them.

lie in a boundary curve. There can be parts of $\mathbf{c}$ *along a boundary curve* $\mathbf{b}_i$*; the parametrization of* $\mathbf{c}$ *along* $\mathbf{b}_i$ *has its 4-th derivative vectors orthogonal to* $\mathbf{b}_i$.

In view of the obvious extension to $\mathbb{R}^3$, we construct the blending areas of M with help of distance fields to the obstacles. We use an algorithm from [Tsa02] for distance field computation, since it stores the normal footpoints; this is helpful for the projection onto M. Let $d_i(x, y)$ be the signed distance of the point (x, y) from the obstacle $\mathcal{O}_i$. Then, the corresponding blending surface is given implicitly by the equation $f(d_i(x, y), z) = 0$, where f describes the shape of the blend profile (Fig. 6.1).

Note that the initial curve already determines the *combinatorial type* of the final spline curve. For an automatic determination we suggest to use an initial curve composed of geodesics (in the presence of obstacles) between consecutive points $\mathbf{p}_i, \mathbf{p}_{i+1}$. These are computed by propagating the distance field originating in $\mathbf{p}_i$ within M. As soon as the propagating wave reaches $\mathbf{p}_{i+1}$ and $\mathbf{p}_{i-1}$, we trace back with a gradient descent method, see Fig. 6.3. The distance propagation is done with an adapted version of Zhao's sweeping algorithm [Zha04], where grid nodes inside obstacles get a flag and are not used for the propagation.

The spline through the channel shown by Fig. 6.4, right, also shows that

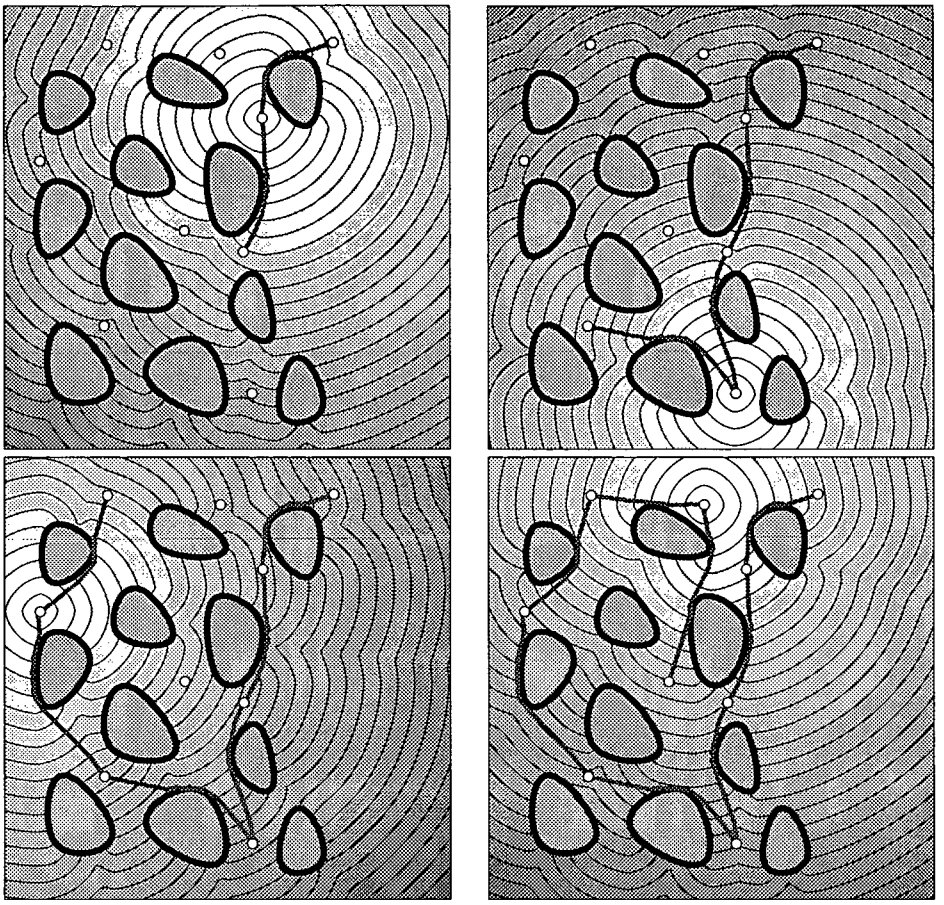


Figure 6.3: Computing the initial curve piecewise via backward gradient flow on the distance field to a point that respects the given obstacles.

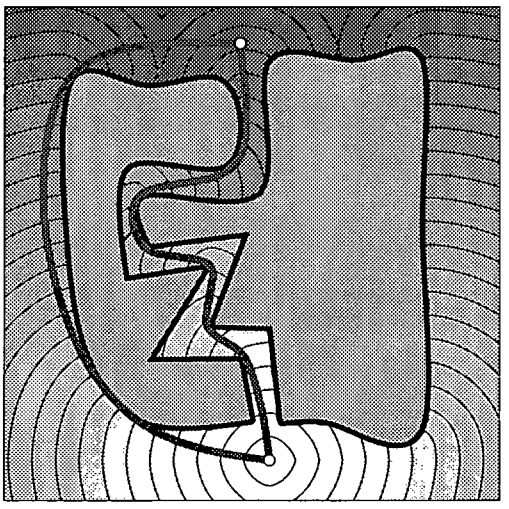


Figure 6.4: Variational curve design in the presence of obstacles is also a combinatorial problem. The path through the channel is shorter but has higher energy than the minimum energy curve passing the two obstacles on the left.

corners do not cause problems: at a convex corner, the blend is smooth; at a concave corner, a blend as defined above would have a sharp edge, but a smooth spline never reaches a concave corner. Moreover, this figure also illustrates a second, combinatorially different solution; it is longer than the path through the channel, but has a smaller energy E_2.

The extension from 2D to 3D is completely straightforward. Figure 6.5 shows 3D examples of spline curves in the presence of obstacles.

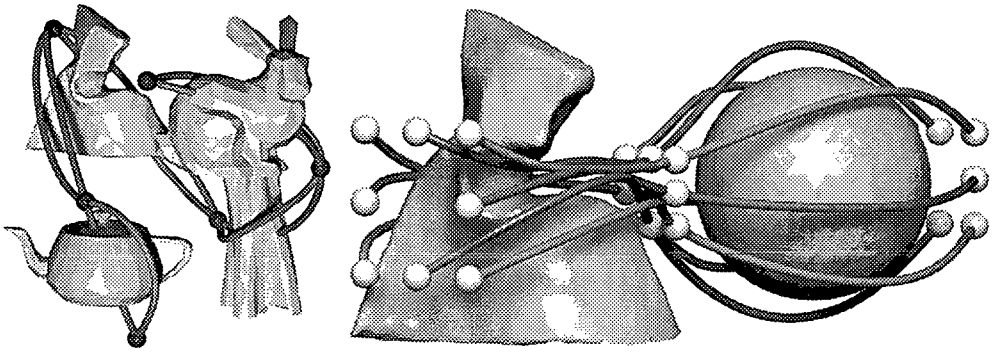


Figure 6.5: Variational curve design in the presence of obstacles in 3D.

We summarize the algorithm for Variational Curve Design in the Presence of Obstacles (VCDPO):

Algorithm 6 (VCDPO). *The algorithm employs the following steps:*

1. *Given are N points $\mathbf{p}_j$ outside the obstacles $\mathcal{O}_i$ and an energy functional $E \in \{E_1, E_2, E_t\}$.*

2. *Compute an initial curve connecting the input points that avoids the obstacles: First, for every second point $\mathbf{p}_2, \mathbf{p}_4, \ldots$, compute the distance field to this point that respects the obstacles (with the modified sweeping algorithm of Zhao [Zha04]). Then with a gradient backward flow connect the two adjacent points $\mathbf{p}_1$ and $\mathbf{p}_3$ to $\mathbf{p}_2$, $\mathbf{p}_3$ and $\mathbf{p}_5$ to $\mathbf{p}_4$ and so forth.*

3. *Compute the distance function to the obstacles with the sweeping algorithm of Tsai [Tsa02].*

4. *Use a profile function $f(d)$ to compute from the distance function of step 3 the family of barrier manifolds M.*

5. *Lift the initial curve onto the barrier manifold with the largest rounding radius.*

6. *Run the iterative geometric optimization procedure (Algorithm 4 of Chapter 4) for the curve on the barrier manifold M. During the iteration reduce the rounding radius of the blending part of M till it vanishes.*

7. *The resulting planar spline curve is the output of the algorithm.*

6.2 Variational Motion Design in the Presence of Obstacles I

In this section we describe how to compute an energy-minimizing motion where the moving body $\mathcal{B}$ avoids given obstacles via a single enclosing ball $\mathcal{B}_e$ centered in the barycenter of the moving body. The enclosing ball $\mathcal{B}_e$ itself avoids the obstacles; since it is centered in the barycenter of $\mathcal{B}$, the problem can be split into the computation of the trajectory of the ball's center and the computation of the rotational part. In this way we combine the methods from Chapter 5 and Sect. 6.1. Only the first part of the computation needs to consider the obstacles: we compute the path such that it does not come closer to the obstacles than the radius r of the enclosing ball $\mathcal{B}_e$. This approach is rather conservative and does not fully exploit the available degrees of freedom. Figure 6.7 shows an example for Variational Motion Design in the Presence of Obstacles. In the following we abbreviate this algorithm as VMDPO I.

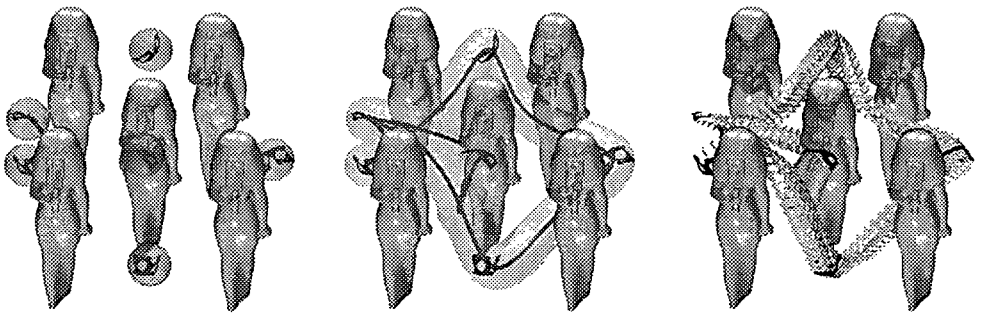


Figure 6.6: Motion design in the presence of obstacles I: (left) collision free input positions of the moving body; (middle) obstacle avoiding path of barycenter; (right) obstacle avoiding rigid body motion.

Algorithm 7 (VMDPO I). *The algorithm employs the following steps:*

1. *Given are N collision free input positions $\mathcal{B}(u_j)$ at time instances u_j, obstacles $\mathcal{O}_i$, and an energy functional $E \in \{E_1, E_2, E_t\}$.*

2. *Replace the moving body $\mathcal{B}$ by a minimum enclosing ball $\mathcal{B}_e$ with radius r centered in the barycenter $\mathbf{s}$ of $\mathcal{B}$, see Fig. 6.6 (left).*

3. *Using Algorithm 6 compute the path of the barycenter $\mathbf{s}(u)$ such that it minimizes E and avoids the given obstacles with a minimum distance r, see Fig. 6.6 (middle).*

4. *Separately compute the rotational part of the motion using step 3 of Algorithm 5 (the VMD algorithm) such that the energy E of a curve on $M^3 \subset \mathbb{R}^9$ is minimized, see Fig. 6.6 (right).*

In Fig. 6.7 we compare energy-minimizing motions interpolating the same input positions. Fig. 6.7 (left) shows the motion computed without consideration of the present obstacles. Fig. 6.7 (right) shows the motion computed with VMDPO I such that the given obstacles are avoided in a conservative way.

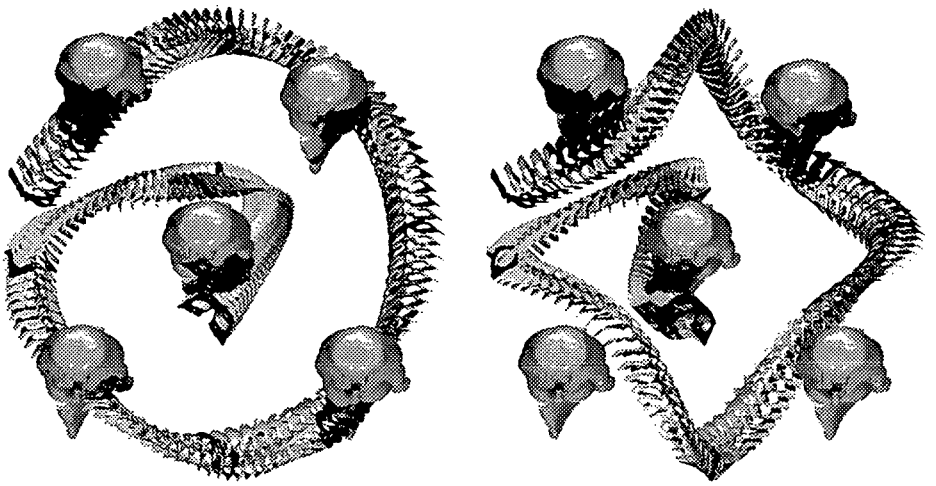


Figure 6.7: Motion design in the presence of obstacles I: Motion minimizing E_2 (left) unconstrained, (right) avoiding the 5 obstacles.

6.3 Variational Motion Design in the Presence of Obstacles II

So far, we have only described how to compute an energy-minimizing motion where the moving body $\mathcal{B}$ avoids given obstacles via a single enclosing ball $\mathcal{B}_e$. The enclosing ball $\mathcal{B}_e$ itself avoids the obstacles; since it is centered in the barycenter of $\mathcal{B}$, the problem can be split into the computation of the trajectory of the ball's center and the computation of the rotational part. Only the first part of the computation needs to consider the obstacles. This approach is rather conservative and does not fully exploit the available degrees of freedom. In the present section we capture the shape of the moving body $\mathcal{B}$ more precisely. For the theoretic considerations we consider the moving body $\mathcal{B}$ and the obstacle $\mathcal{O}$ to be smooth solid bodies. Again we use the embedding of the Euclidean motion group as a manifold M^6 in the space E^{12} of affine maps and the metric in E^{12} is computed with the mass distribution of the moving body introduced in Chapter 2.

6.3.1 Definition of the Barrier Manifold

The moving body $\mathcal{B}$ is a solid. A position $\alpha(\mathcal{B})$ of the body corresponds to some displacement α, seen as a point A in M^6. In the following, we speak of a single obstacle $\mathcal{O}$, but note that it may contain several components. For simplicity,

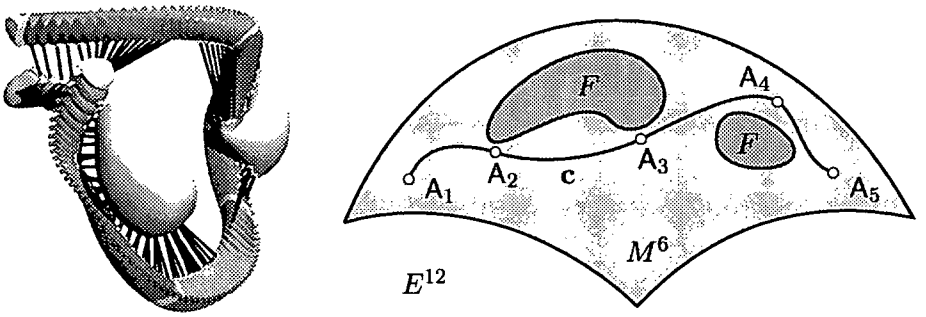


Figure 6.8: (Left) Rigid body motion $\mathcal{B}(u)$ in $\mathbb{R}^3$ interpolating 5 given positions with shortest distances to the obstacles. (Right) Forbidden region F and the curve $\mathbf{c}$ corresponding to $\mathcal{B}(u)$ on the manifold $M^6 \subset E^{12}$.

we assume right now that both $\mathcal{B}$ and $\mathcal{O}$ have smooth boundary surfaces. In our implementation we use triangle meshes to represent both, $\mathcal{B}$ and $\mathcal{O}$.

Those positions $\alpha(\mathcal{B})$, which penetrate $\mathcal{O}$, have to be avoided. One may view these forbidden positions as points in some subset F of M^6, see Fig. 6.8. Like $\mathcal{O}$, it may have several components. In order to stay away from F, we build a barrier manifold against it, according to the concept presented in Sect. 6.1, see Fig. 6.9. To do so, we use a *distance function d*. The distance $d(\alpha) = d(\mathsf{A}) =: d_0$ is the shortest distance between the corresponding position $\alpha(\mathcal{B})$ of $\mathcal{B}$ and the obstacle $\mathcal{O}$, see Fig. 6.10. Note that we view d as a function defined on M^6.

In case of penetration, we define $d(\mathsf{A}) = -1$. Of special interest is the zero level set of d, since it is the boundary ∂F of F. It contains exactly those positions in which $\alpha(\mathcal{B})$ is tangent to $\mathcal{O}$, but not penetrating the obstacle. Likewise, any other level set to a constant distance value d_0 contains the positions, where $\alpha(\mathcal{B})$ is in contact with the offset $\widetilde{\mathcal{O}}$ of $\mathcal{O}$ at distance d_0, see Fig. 6.10.

As in Sect. 6.1, we use a function $f(d)$ (blending profile function) which is supported on some interval $[0, D]$. It describes a smooth blend between f-axis and d-axis, and thus it has a positive value h and infinite derivative at 0, and satisfies $f(D) = f'(D) = 0$, see Fig. 6.9.

The barrier manifold $M_{\mathcal{B}}$ is a 6-dimensional manifold, embedded in $\mathbb{R}^{13}$. Part of it can be parameterized over M^6: If $d(\mathsf{A}) > 0$, the corresponding point on $M_{\mathcal{B}}$ is $(\mathsf{A}, f(d(\mathsf{A})))$. This surface contains a part in M^6 (for $d \geq D$), and a blending part which reaches height h in the 13-th coordinate, when A reaches the boundary ∂F of the forbidden region F. There, the surface is joined with a cylinder surface defined over ∂F. Let P be a parametrization of ∂F (over an appropriate subset of $\mathbb{R}^5$), then we obtain with an additional parameter $v \geq h$ a parametrization (P, v) of this 6-dimensional cylindrical part.

6.3.2 Tangent Spaces of the Barrier Manifold

Tangent spaces of M^6 follow from velocity vector fields $\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}$ in $\mathbb{R}^3$. According to Equ. 2.10 of Sect. 2.3, the corresponding tangent vector at a position $\mathsf{A} = (\mathbf{a}_0, \ldots, \mathbf{a}_3)$ is given by

$$\mathsf{T} = (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{a}_0, \mathbf{c} \times \mathbf{a}_1, \mathbf{c} \times \mathbf{a}_2, \mathbf{c} \times \mathbf{a}_3). \tag{6.1}$$

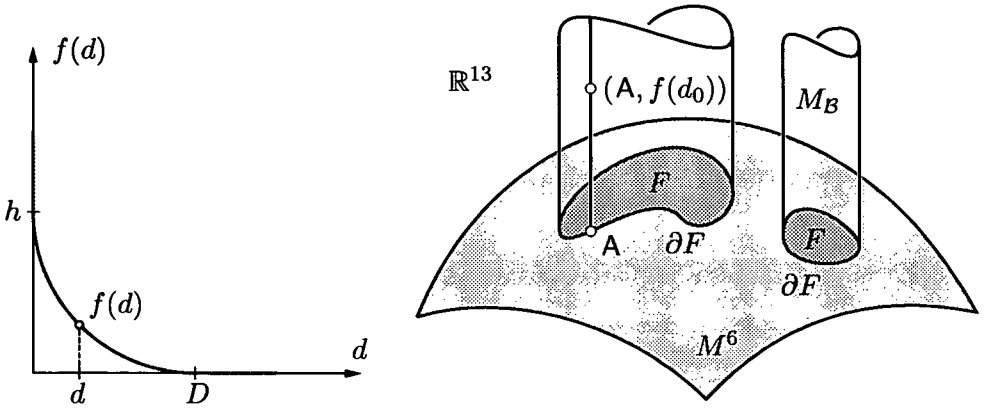


Figure 6.9: (Left) Blending profile function $f(d)$. (Right) Forbidden region F with boundary ∂F and cylindric parts of barrier manifold $M_{\mathcal{B}}$.

Case 1: $d(\mathsf{A}) \geq D$. Equation (6.1) suffices to compute the tangent space of $M_{\mathcal{B}}$ for $d(\mathsf{A}) \geq D$; we just have to add a zero as 13-th coordinate.

Case 2: $0 < d(\mathsf{A}) < D$. This is the case of main interest. Let N_c be the common normal between $\alpha(\mathcal{B})$ and $\mathcal{O}$, along which the shortest distance $d_0 = d(\mathsf{A})$ occurs, see Fig. 6.10. N_c meets $\alpha(\mathcal{B})$ at a foot point $\mathbf{p}_f$. There, we consider two straight lines T_1, T_2 which are orthogonal to the contact normal N_c and thus tangent to the position $\alpha(\mathcal{B})$ of the body $\mathcal{B}$, see Fig. 6.10. We may assume orthogonal T_i's and then T_1, T_2, N_c define a Cartesian frame at the foot point $\mathbf{p}_f$. At first, we derive those tangent vectors of $M_{\mathcal{B}}$, whose 13-th coordinate is zero. These are characterized by vanishing directional derivative of the function d. Clearly, they belong to velocity fields of gliding motions along the offset $\widetilde{\mathcal{O}}$ of $\mathcal{O}$ at distance d_0. Note that $\mathbf{p}_f$ is the contact point and N_c is the contact normal for such a gliding motion. By a well known result from kinematical geometry [PW01], these velocity fields are characterized by an equation

$$\bar{\mathbf{n}}_c \cdot \mathbf{c} + \mathbf{n}_c \cdot \bar{\mathbf{c}} = 0. \tag{6.2}$$

Here, $(\mathbf{n}_c, \bar{\mathbf{n}}_c)$ are the Plücker coordinates of N_c. This says that the common normal N_c is contained in the linear complex of instantaneous path normals.

It is easy to derive five independent velocity fields of gliding motions: We use instantaneous rotations about T_1, T_2, N_c and translations parallel to T_1, T_2. With $(\mathbf{t}_i, \bar{\mathbf{t}}_i)$ as Plücker coordinates of T_i, the corresponding velocity fields $(\mathbf{c}, \bar{\mathbf{c}})$ are

$$(\mathbf{c}_1, \bar{\mathbf{c}}_1) = (\mathbf{t}_1, \bar{\mathbf{t}}_1),\ (\mathbf{c}_2, \bar{\mathbf{c}}_2) = (\mathbf{t}_2, \bar{\mathbf{t}}_2),\ (\mathbf{c}_3, \bar{\mathbf{c}}_3) = (\mathbf{n}_c, \bar{\mathbf{n}}_c), \tag{6.3}$$

and

$$(\mathbf{c}_4, \bar{\mathbf{c}}_4) = (0, \mathbf{t}_1),\ (\mathbf{c}_5, \bar{\mathbf{c}}_5) = (0, \mathbf{t}_2). \tag{6.4}$$

To verify equation (6.2) for these velocity fields, one uses the condition $\bar{\mathbf{g}} \cdot \mathbf{h} + \mathbf{g} \cdot \bar{\mathbf{h}} = 0$ for two intersecting lines $G = (\mathbf{g}, \bar{\mathbf{g}})$ and $H = (\mathbf{h}, \bar{\mathbf{h}})$. Thus, we obtain with (6.1) five tangent vectors of $M_{\mathcal{B}}$ at the point $(\mathsf{A}, f(d(\mathsf{A})))$,

$$\mathsf{T}_i = (\bar{\mathbf{c}}_i + \mathbf{c}_i \times \mathbf{a}_0, \mathbf{c}_i \times \mathbf{a}_1, \mathbf{c}_i \times \mathbf{a}_2, \mathbf{c}_i \times \mathbf{a}_3, 0), \quad i = 1, \ldots, 5. \tag{6.5}$$

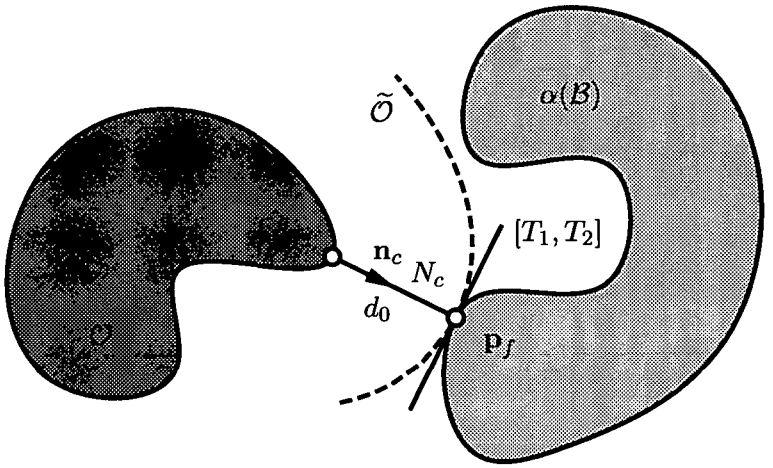


Figure 6.10: Shortest distance d_0 between the moving body position $\alpha(\mathcal{B})$ and the obstacle $\mathcal{O}$.

We assume that $\mathbf{n}_c$ is normalized and points outside the obstacle, i.e., in direction of increasing d. Displacing $\alpha(\mathcal{B})$ by a translation with vector $\lambda\mathbf{n}_c$ changes the distance value to $d_0 + \lambda$, since the contact normal remains the same for sufficiently small λ. In other words, $d(\mathsf{A}_\lambda) = d_0 + \lambda$ at $\mathsf{A}_\lambda = (\mathbf{a}_0 + \lambda\mathbf{n}_c, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$. This leads to a curve C in the barrier manifold, parameterized with help of λ,

$$\mathsf{C}(\lambda) = (\mathbf{a}_0 + \lambda\mathbf{n}_c, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, f(d_0 + \lambda)). \tag{6.6}$$

The tangent of this curve at $\lambda = 0$ gives us the 6-th tangent vector,

$$\mathsf{T}_6 = (\mathbf{n}_c, 0, 0, 0, f'(d_0)). \tag{6.7}$$

Case 3: $d(\mathsf{A}) = 0$. To each point A in the boundary ∂F of the forbidden region, i.e., $d_0 = d(\mathsf{A}) = 0$, we have an infinite number of points in $M_\mathcal{B}$, namely in its cylindrical part, see Fig. 6.9. However, at all points of such a cylinder ruling, the tangent space is the same. It is spanned by the five vectors (6.5) to instantaneous gliding motions of the body along the obstacle and by $(0, \ldots, 0, 1) \in \mathbb{R}^{13}$.

As mentioned earlier, one should not use a graph representation of the profile. If (t_1, t_2) is a tangent vector of the profile curve at d_0, i.e., (t_1, t_2) is parallel to $(1, f'(d_0))$, we use

$$\mathsf{T}_6 = (t_1\mathbf{n}_c, 0, 0, 0, t_2). \tag{6.8}$$

This representation also holds for $d_0 = 0$, since then we have $(t_1, t_2) = (0, 1)$.

6.3.3 Projection Onto the Barrier Manifold

In the previous section we have derived basis vectors of the 6-dimensional tangent space at a point $(\mathsf{A}, f(d(\mathsf{A})))$ of the barrier manifold $M_\mathcal{B} \subset \mathbb{R}^{13}$. This allows us to compute the tangent space of the high-dimensional surface Φ used

in Algorithm 4, the geometric optimization algorithm presented in Chapter 4. Once we have computed the stepsize of the current iteration step, we apply the displacement in the tangent space of Φ. Then we need an admissible projection from a point in the tangent space of Φ onto Φ. One possibility is to perform the projection in the low-dimensional space for each position $\mathsf{A}_s \subset \mathbb{R}^{13}$ of the moving body separately. First we project the point A_s orthogonally onto M^6 by setting the 13-th coordinate equal to zero. The new translation of the Euclidean displacement in $\mathbb{R}^3$ is given by the first three coordinates of the point A_s, and the new rotation matrix in $\mathbb{R}^3$ is the best-fit orthogonal matrix to the affine matrix given by the 4-th till 12-th coordinate of A_s, computed with the methods presented in Sect. 2.4. Then we compute the distance of each new position of the moving body to the obstacle $\mathcal{O}$ to get the new 13-th coordinate, i.e., all together the new position on the barrier manifold $M_{\mathcal{B}}$. In case of penetration we translate the moving body position in direction of the common normal out of the obstacle.

Algorithm 8 (VMDPO II). *The algorithm employs the following steps:*

1. *Compute an initial variational motion that avoids the given obstacles with Algorithm 7 (VMDPO I).*

2. *Further optimize the Euclidean motion of step 1 by using the iterative geometric optimization (Algorithm 4) and the barrier manifold $M_{\mathcal{B}}$.*

6.3.4 Examples of Energy-Minimizing Motions Avoiding Obstacles

The example illustrated in Fig. 6.11 shows 4 obstacles and 3 different Euclidean motions that interpolate the same 5 given positions of the rigid body $\mathcal{B}$. All motions minimize the energy functional E_2, but using different algorithms: The unconstrained Euclidean motion computed with Algorithm. 5 (VMD) penetrates two of the four obstacles, see Fig. 6.11 (left). An energy-minimizing motion that avoids the given obstacles in a conservative way is computed with Algorithm 7 (VMDPO I), see Fig. 6.11 (middle). Algorithm 8 (VMDPO II) gives a better result since it uses all available degrees of freedom in the optimization procedure, see Fig. 6.11 (right).

In the implementation, the computation time for all optimization parts is in sum a few seconds. The distance computations with the algorithms form the literature [Tsa02, Zha04] are a bit more time consuming.

In Fig. 6.12 we use the same input situation, i.e., the same obstacles and the same input positions as in the example shown in Fig. 6.11. The only difference is that now we compute cyclic motions. Again we compare the results of VMD (the unconstrained case) with the resulting obstacle avoiding motions computed with VMDPO I and VMDPO II. The energy level of the rigid body motion computed with VMDPO II is lower than the one computed with VMDPO I.

Figures 6.13 and 6.14 show another example of an open and a cyclic energy-minimizing motion in the presence of obstacles, where the same 4 input positions are interpolated and one obstacle with a hole is avoided.

The final example of this thesis is illustrated in Fig. 6.15. We compare two unconstrained energy-minimizing motions penetrating two obstacles with the energy-minimizing motions computed with VMDPO II that avoid the given obstacles. Each of the two motions interpolates three given collision free input positions. Figure 6.16 shows 18 different views of the example illustrated in Fig. 6.15 (bottom). Figure 6.17 shows three more views of the two motions computed with VMDPO II shown in Fig. 6.15 (bottom) and is the final figure of this thesis.

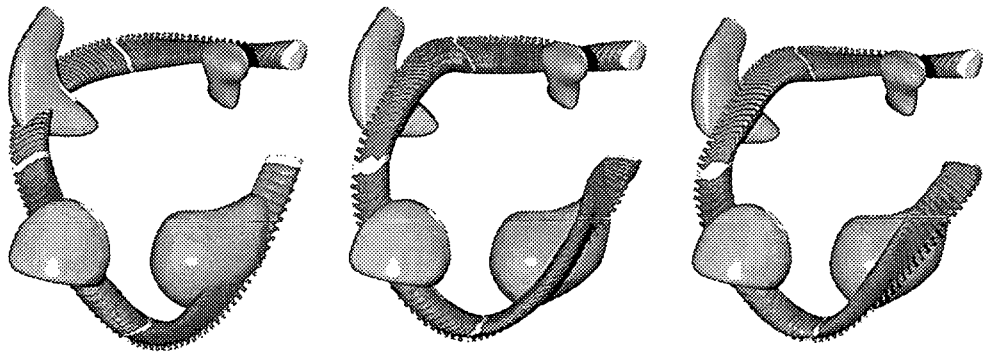


Figure 6.11: Comparison of energy-minimizing Euclidean motions: (left) VMD, (middle) VMDPO I, (right) VMDPO II.

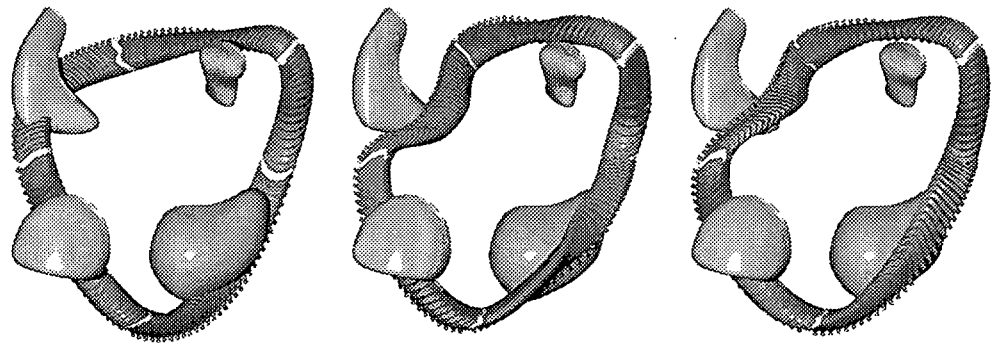


Figure 6.12: Comparison of cyclic energy-minimizing Euclidean motions: (left) VMD, (middle) VMDPO I, (right) VMDPO II.

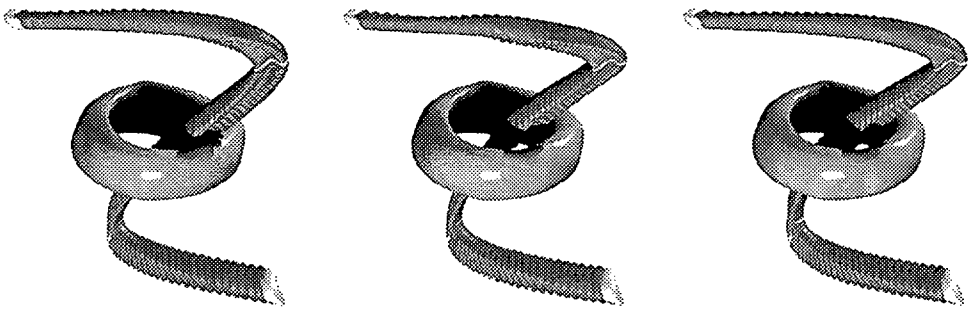


Figure 6.13: Comparison of energy-minimizing Euclidean motions: (left) VMD, (middle) VMDPO I, (right) VMDPO II.

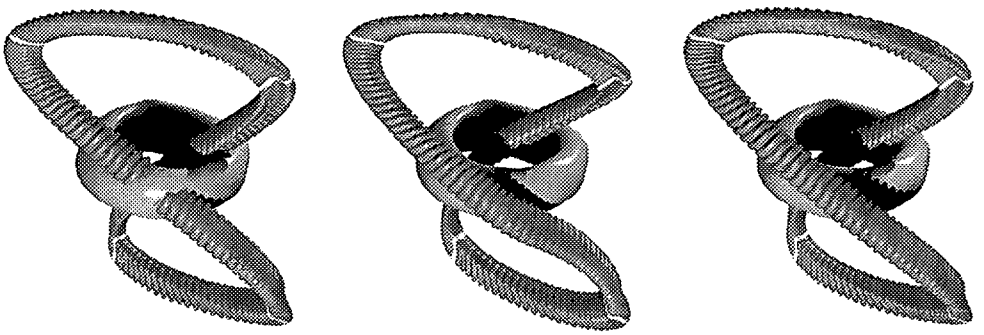


Figure 6.14: Comparison of cyclic energy-minimizing Euclidean motions: (left) VMD, (middle) VMDPO I, (right) VMDPO II.

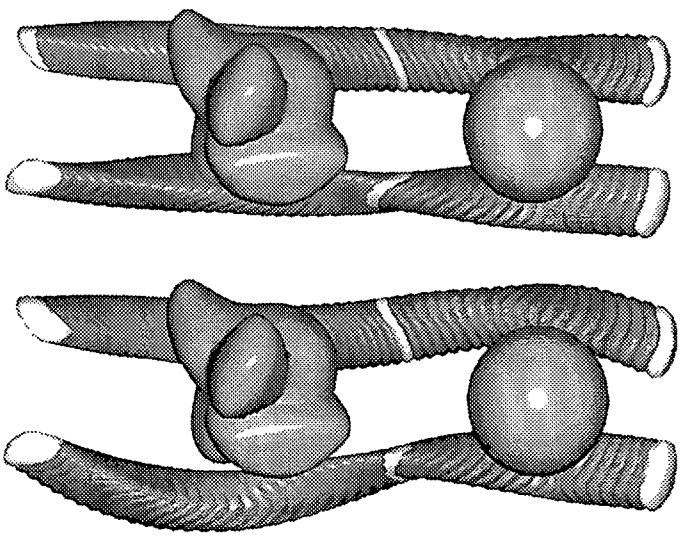


Figure 6.15: Two energy-minimizing Euclidean motions: (top) unconstrained, (bottom) avoiding the two obstacles.

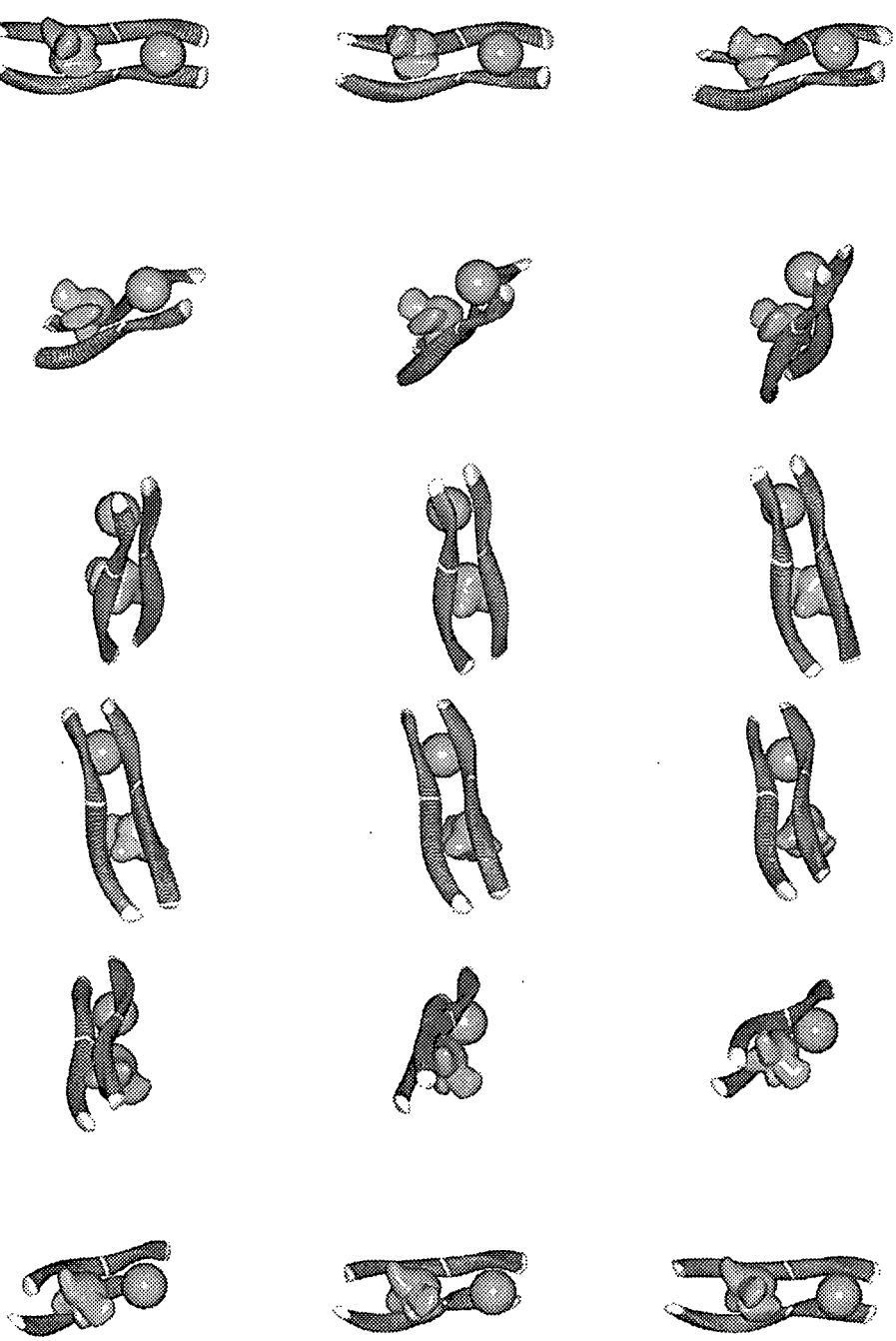


Figure 6.16: Different views of the same obstacle avoiding Euclidean motions.

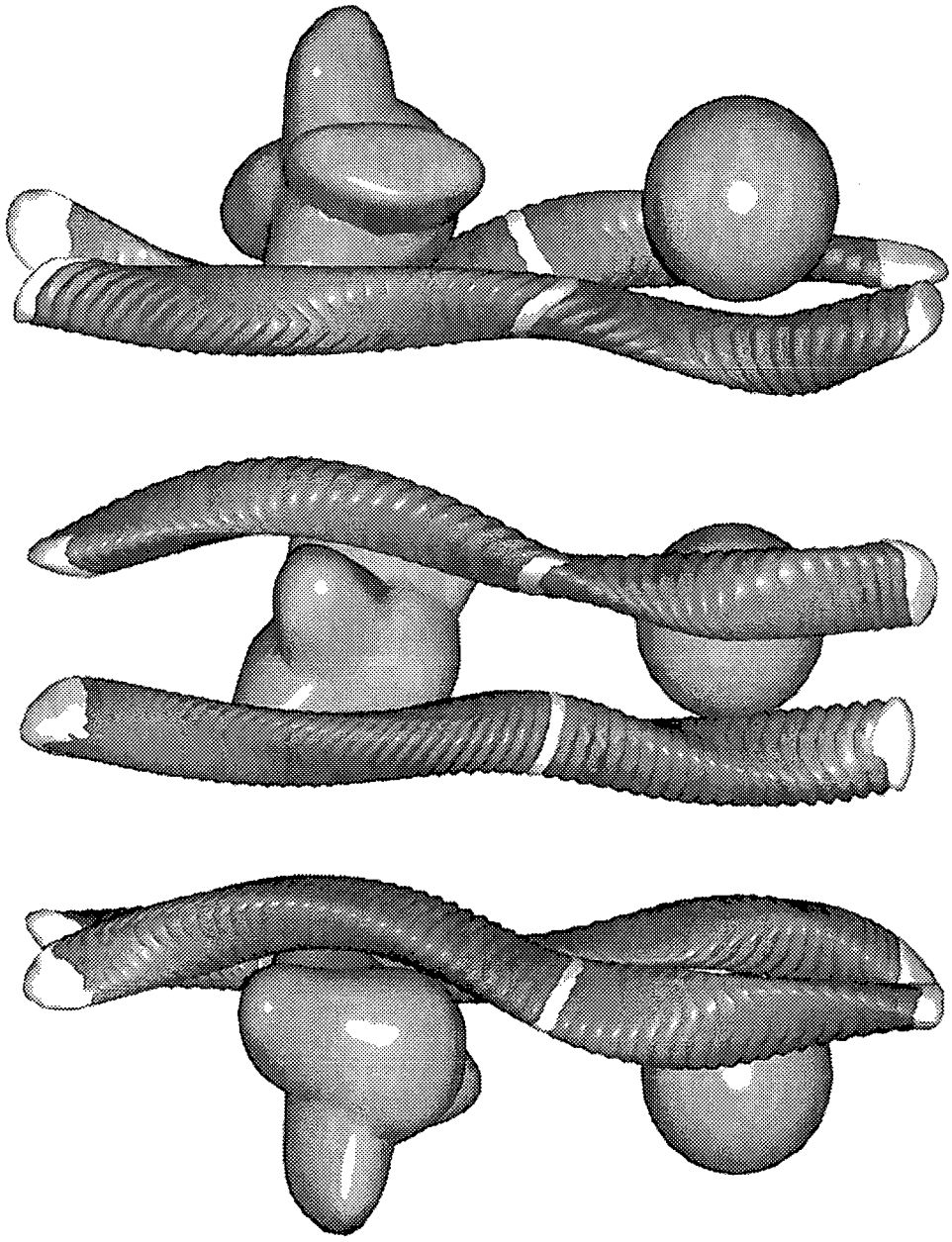


Figure 6.17: Three different views of two energy-minimizing motions that avoid two given obstacles.

Chapter 7

Conclusions and Outlook

In this thesis a kinematic image space is used where Euclidean maps in three dimensional space are viewed as points of a six dimensional manifold M^6, embedded in 12-dimensional Euclidean space E^{12}. The latter space corresponds to affine maps, and the metric therein is defined naturally with help of sample points (a mass distribution) of the moving body. A first motion design algorithm (the CMD algorithm) is a transfer principle from curve design algorithms to the design of rigid body motions. Since it does not preserve energy-minimizing properties we only use it to find a good initial Euclidean motion for the iterative VMD algorithm, that computes an energy-minimizing motion in the unconstrained case. We have studied energy-minimizing splines in manifolds as the counterparts on surfaces to well known energy-minimizing splines of geometric modelling such as C^2 cubic splines or splines in tension. Theoretic results and a geometric optimization algorithm have been derived for splines on surfaces of arbitrary dimension and codimension. Variational motion design and variational motion design in the presence of obstacles are special instances of this general setting, and are thus transferred to variational curve design on an appropriate manifold. In the unconstrained case the manifold is the group of Euclidean congruence transformations, in the obstacle case we have derived an appropriate barrier manifold on which we perform our computations. We studied two algorithms for variational motion design in the presence of obstacles. The first one is a conservative approach that reduces the motion design problem to a curve design problem in the presence of obstacles. The second one fully employs the available degrees of freedom.

Future research concerns fair webs in manifolds, i.e., curve networks that minimize a chosen energy-functional. The generalization from the curve to the surface case leads to the study of energy-minimizing surfaces in higher-dimensional manifolds and also variational surface design in the presence of obstacles. In this thesis energy-minimizing *one-parametric* motions in the presence of obstacles were studied. A possible extension of this work is also to study *k-parametric* energy-minimizing motions, also in the presence of obstacles. The avoidance of obstacles has further applications in the area of numerically controlled machining. Finally, to include the avoidance of obstacles in a combination of registration and surface fitting algorithms is a promising direction of future research.

Appendix A

Quaternions and Kinematics

We include some basic facts about quaternions and their relation to spatial kinematics that are used in Sect. 2.4.3. Quaternions have been introduced in 1853 by Hamilton. A modern primer on quaternions is the book by Kuipers [Kui02]. A *quaternion* $\mathbf{q}$ can be considered as the extension of a complex number to four components,

$$\mathbf{q} = q_0 + iq_1 + jq_2 + kq_3, \tag{A.1}$$

with $q_0, \ldots, q_3 \in \mathbb{R}$. The symbols i, j, k are called *imaginary units* and fulfill the following rules

$$i \circ i = j \circ j = k \circ k = -1 \tag{A.2}$$

$$\begin{aligned} i \circ j &= k = -j \circ i && \text{(A.3)} \\ j \circ k &= i = -k \circ j && \text{(A.4)} \\ k \circ i &= j = -i \circ k. && \text{(A.5)} \end{aligned}$$

We denote by $\mathbb{H} = \mathbb{R}^4$ the set of all quaternions. There is a straightforward way to embed vectors $\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$ into $\mathbb{H} = \mathbb{R}^4$ as $(0, x_1, x_2, x_3) = ix_1 + jx_2 + kx_3$. For a quaternion $\mathbf{q} = q_0 + iq_1 + jq_2 + kq_3$ one refers to

$$S(\mathbf{q}) := q_0, \quad V(\mathbf{q}) := iq_1 + jq_2 + kq_3 \tag{A.6}$$

as the *scalar part* $S(\mathbf{q})$ and *vector part* $V(\mathbf{q})$ respectively. Quaternion addition $+ : \mathbb{H} \times \mathbb{H} \to \mathbb{H}$ of two quaternions $\mathbf{p}, \mathbf{q}$ is defined componentwise,

$$\mathbf{p} + \mathbf{q} := (p_0 + q_0) + i(p_1 + q_1) + j(p_2 + q_2) + k(p_3 + q_3). \tag{A.7}$$

Quaternion multiplication $\circ : \mathbb{H} \times \mathbb{H} \to \mathbb{H}$ follows from the above multiplication rules for the imaginary units,

$$\mathbf{p} \circ \mathbf{q} := [p_0 q_0 - V(\mathbf{p}) \cdot V(\mathbf{q})] + [p_0 V(\mathbf{q}) + q_0 V(\mathbf{p}) + V(\mathbf{p}) \times V(\mathbf{q})]. \tag{A.8}$$

Thereby $\cdot$ and $\times$ denote the scalar product and vector product of vectors in $\mathbb{R}^3$. Quaternion multiplication is for $V(\mathbf{p}) \times V(\mathbf{q}) \neq 0$ not commutative, since

$$\mathbf{p} \circ \mathbf{q} - \mathbf{q} \circ \mathbf{p} = 2V(\mathbf{p}) \times V(\mathbf{q}). \tag{A.9}$$

It follows that $(\mathbb{H}, +, \circ)$ is a *skew field*. For a quaternion $\mathbf{q}$ its *conjugate quaternion* $\overline{\mathbf{q}}$ is defined as

$$\overline{\mathbf{q}} = q_0 - iq_1 - jq_2 - kq_3. \tag{A.10}$$

Similar to the real and imaginary part of a complex number, which can be computed using the complex number and its conjugate, the scalar and vector part of a quaternion can be computed from the quaternion and its conjugate as follows:

$$S(\mathbf{q}) = \frac{1}{2}(\mathbf{q} + \overline{\mathbf{q}}), \tag{A.11}$$

$$V(\mathbf{q}) = \frac{1}{2}(\mathbf{q} - \overline{\mathbf{q}}). \tag{A.12}$$

For each two quaternions $\mathbf{q}, \mathbf{p}$ the conjugate of their quaternion product equals the quaternion product of their conjugates in reversed order, i.e.,

$$\overline{\mathbf{q} \circ \mathbf{p}} = \overline{\mathbf{p}} \circ \overline{\mathbf{q}}. \tag{A.13}$$

The *norm* $N(\mathbf{q})$ of a quaternion $\mathbf{q}$ is defined by

$$N(\mathbf{q}) := q_0^2 + q_1^2 + q_2^2 + q_3^2 = \mathbf{q} \circ \overline{\mathbf{q}} = \overline{\mathbf{q}} \circ \mathbf{q}. \tag{A.14}$$

The *multiplicative inverse* $\mathbf{q}^{-1}$ of a nonzero quaternion $\mathbf{q}$ is given by

$$\mathbf{q}^{-1} = \frac{\overline{\mathbf{q}}}{N(\mathbf{q})}. \tag{A.15}$$

The *scalar product* of two quaternions $\mathbf{q}, \mathbf{p}$ is the scalar product of two vectors in $\mathbb{R}^4$,

$$\langle \mathbf{q}, \mathbf{p} \rangle := q_0 p_0 + q_1 p_1 + q_2 p_2 + q_3 p_3, \tag{A.16}$$

and can also be written as

$$\langle \mathbf{q}, \mathbf{p} \rangle = \frac{1}{2}(\mathbf{q} \circ \overline{\mathbf{p}} + \mathbf{p} \circ \overline{\mathbf{q}}) = \frac{1}{2}(\overline{\mathbf{q}} \circ \mathbf{p} + \overline{\mathbf{p}} \circ \mathbf{q}). \tag{A.17}$$

For three quaternions $\mathbf{p}, \mathbf{q}, \mathbf{r}$ we have the following property of the scalar product,

$$\langle \mathbf{q} \circ \mathbf{p}, \mathbf{r} \rangle = \langle \mathbf{p}, \overline{\mathbf{q}} \circ \mathbf{r} \rangle. \tag{A.18}$$

Quaternion multiplication of a quaternion $\mathbf{x}$ by a quaternion $\mathbf{q}$ from the left can be written in matrix notation as

$$\mathbf{q} \circ \mathbf{x} = \begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} =: Q \cdot \mathbf{x}. \tag{A.19}$$

Note that if $N(\mathbf{q}) = q_0^2 + \cdots + q_3^2 = 1$, then the matrix Q is orthogonal with $\det Q = 1$. Quaternion multiplication of a quaternion $\mathbf{x}$ by a quaternion $\mathbf{q}$ from the right can be written in matrix notation as

$$\mathbf{x} \circ \mathbf{q} = \begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} =: \widetilde{Q} \cdot \mathbf{x}. \tag{A.20}$$

The matrix $\widetilde{Q}$ differs from Q only in the lower right 3 by 3 submatrix, which is the transpose of the corresponding submatrix of Q. Again we have that $\widetilde{Q}$ is orthogonal if $N(\mathbf{q}) = 1$. The following proposition shows the relation between unit quaternions and rotations in $\mathbb{R}^3$.

Proposition 1. *Let $\mathbf{x} \in \mathbb{R}^3$ be a vector and let $\mathbf{q} \in \mathbb{H}$ be a fixed unit quaternion, i.e., $N(\mathbf{q}) = 1$. Then the map*

$$\mathbf{x}' := \mathbf{q} \circ \mathbf{x} \circ \overline{\mathbf{q}} \tag{A.21}$$

describes a rotation $\mathbf{x}' = R \cdot \mathbf{x}$ with the rotation matrix

$$R = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}. \tag{A.22}$$

This gives an explicit parametrization of the group of orthogonal matrices with parameters q_0, q_1, q_2, q_3.

Proof. If we embed $\mathbf{x} \in \mathbb{R}^3$ into $\mathbb{H}$ we have seen above that the scalar part of such a quaternion vanishes, i.e., $S(\mathbf{x}) = 0$. In order to show that $\mathbf{x}' = \mathbf{q} \circ \mathbf{x} \circ \overline{\mathbf{q}}$ maps vectors $\mathbf{x} \in \mathbb{R}^3$ to vectors $\mathbf{x}' \in \mathbb{R}^3$, we need to show that also the scalar part of $\mathbf{x}'$ vanishes. Manipulating (A.11) by using property (A.13) of quaternions, we see that the scalar part of $\mathbf{x}'$ vanishes,

$$\begin{aligned} 2S(\mathbf{x}') &= \mathbf{x}' + \overline{\mathbf{x}}' \\ &= \mathbf{q} \circ \mathbf{x} \circ \overline{\mathbf{q}} + \overline{\mathbf{q} \circ \mathbf{x} \circ \overline{\mathbf{q}}} \\ &= \mathbf{q} \circ \mathbf{x} \circ \overline{\mathbf{q}} + \mathbf{q} \circ \overline{\mathbf{x}} \circ \overline{\mathbf{q}} \\ &= \mathbf{q} \circ \mathbf{x} \circ \overline{\mathbf{q}} - \mathbf{q} \circ \mathbf{x} \circ \overline{\mathbf{q}} \\ &= 0. \end{aligned}$$

Thus the image $\mathbf{x}'$ is again a vector in $\mathbb{R}^3$. If $N(\mathbf{q}) = \overline{\mathbf{q}} \circ \mathbf{q} = \mathbf{q} \circ \overline{\mathbf{q}} = 1$, then

$$N(\mathbf{x}') = \mathbf{x}' \circ \overline{\mathbf{x}}' = \mathbf{q} \circ \mathbf{x} \circ \overline{\mathbf{q}} \circ \mathbf{q} \circ \overline{\mathbf{x}} \circ \overline{\mathbf{q}} = N(\mathbf{x}), \tag{A.23}$$

which means that the map (A.21) preserves the length of vectors.

To get a matrix representation of the map $\mathbf{q} \circ \mathbf{x} \circ \overline{\mathbf{q}}$, we use the matrix representation (A.19) for quaternion multiplication, $\mathbf{q} \circ \mathbf{x} = Q \cdot \mathbf{x}$ and $\overline{\mathbf{q}} \circ \mathbf{x} = Q^T \cdot \mathbf{x}$. We abbreviate $\mathbf{y} := \mathbf{q} \circ \mathbf{x}$ and use (A.20) to get the matrix representation $\mathbf{y} \circ \overline{\mathbf{q}} = \widetilde{Q}^T \cdot \mathbf{y}$. Altogether we have

$$\begin{aligned} \mathbf{q} \circ \mathbf{x} \circ \overline{\mathbf{q}} &= \mathbf{y} \circ \overline{\mathbf{q}} \\ &= \widetilde{Q}^T \cdot \mathbf{y} \\ &= \widetilde{Q}^T \cdot \mathbf{q} \circ \mathbf{x} \\ &= \widetilde{Q}^T \cdot Q \cdot \mathbf{x}. \end{aligned}$$

This reveals that the map $\mathbf{x}' = \mathbf{q} \circ \mathbf{x} \circ \overline{\mathbf{q}}$ has the matrix representation $\widetilde{Q}^T \cdot Q$,

$$\widetilde{Q}^T \cdot Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & & & \\ 0 & & R & \\ 0 & & & \end{pmatrix}, \tag{A.24}$$

where R is the matrix (A.22). As a product of two orthogonal matrices, (A.24) is itself orthogonal with $\det(\widetilde{Q}^T \cdot Q) = 1$. This implies that R is a 3 by 3 orthogonal matrix with $\det R = 1$. Thus $\mathbf{x}' = \mathbf{q} \circ \mathbf{x} \circ \overline{\mathbf{q}}$ is a linear map which maps vectors to vectors and preserves lengths. Hence it describes a rotation $\mathbf{x}' = R \cdot \mathbf{x}$. □

Remark A.1 (Rotation matrix from axis and angle of rotation). Using the normalized direction vector $\mathbf{g}$ of the axis of rotation and the angle of rotation φ, one computes a unit quaternion $\mathbf{q}$ describing this rotation as follows:

$$\mathbf{q} = \cos\frac{\varphi}{2} + \sin\frac{\varphi}{2}\mathbf{g}. \tag{A.25}$$

Then one can use Prop. 1 to derive the rotation matrix R from the unit quaternion $\mathbf{q}$. Note that R describes a rotation around an axis passing through the origin. Further note that $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation, and that $\mathbf{q}$ is unchanged if we modify the orientation of $\mathbf{g}$ and the sign of φ simultaneously.

Bibliography

[AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, 1987.

[Arn89] V. I. Arnol'd. *Mathematical Methods of Classical Mechanics.* Springer, 1989.

[BC94] G. Brunnett and P. E. Crouch. Elastic curves on the sphere. *Adv. Comput. Math.*, 2:23–40, 1994.

[BCGH92] A. H. Barr, B. Currin, S. Gabriel, and J. F. Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. In *Proceedings of ACM SIGGRAPH 92*, pages 313–320. ACM Press / ACM SIGGRAPH, 1992.

[BHS93] G. Brunnett, H. Hagen, and P. Santarelli. Variational design of curves and surfaces. *Surveys Math. Ind.*, 3:1–27, 1993.

[BI98] A. Blake and M. Isard. *Active Contours.* Springer, 1998.

[BK00] C. Belta and V. Kumar. An efficient geometric approach to rigid body motion interpolation. In *26th ASME Biennial Mechnanisms Conference, Baltimore, MD*, pages 334–345, 2000.

[BK02] C. Belta and V. Kumar. An SVD-projection method for interpolation on SE(3). *IEEE Trans. Robotics Automation*, 18(3):334–345, 2002.

[BM92] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. and Mach. Intell.*, 14(2):239–256, 1992.

[Boh99] H. Bohl. *Kurven minimaler Energie auf getrimmten Flächen.* PhD thesis, Univ. Stuttgart, 1999.

[Can86] J. Canny. Collision detection for moving polyhedra. *IEEE Trans. Pattern Anal. and Machine Intell.*, 8:200–209, 1986.

[Can88] J. Canny. *The Complexity of Robot Motion Planning.* PhD thesis, MIT, Cambridge, 1988. ACM Doctoral Dissertation Award 1987.

[CBMO02] B. T. Cheng, P. Burchard, B. Merriman, and S. Osher. Motion of curves constrained on surfaces using a level set approach. *Journal of Computational Physics*, 175(2):604–644, 2002.

[CKS97] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *Int. J. Comp. Vision*, 22:61–79, 1997.

[CLC01] M. Camarinha, F. Silva Leite, and P. E. Crouch. On the geometry of Riemannian cubic polynomials. *Diff. Geom. Applic.*, 15:107–135, 2001.

[CM92] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10:145–155, 1992.

[dBvKOS00] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry*. Springer, 2 edition, 2000.

[dC76] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[ELF97] D. W. Eggert, A. Larusso, and R. B. Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, 9:272–290, 1997.

[FH83] O. D. Faugeras and M. Hebert. A 3-d recognition and positioning algorithm using geometric matching between primitive surfaces. In *International Joint Conference on Artificial Intelligence*, volume 8, pages 996–1002, August 1983.

[FHK+98] Y. C. Fang, C. C. Hsieh, M. J. Kim, J. J. Chang, and T. C. Woo. Real time motion fairing with unit quaternions. *Computer-Aided Design*, 30(3):191–198, 1998.

[Fle87] R. Fletcher. *Practical Methods of Optimization*. Wiley, 1987.

[GF63] I. M. Gelfand and S. V. Fomin. *Calculus of Variations*. Prentice-Hall, Englewood Cliffs, N.J., 1963.

[GK85] S. Gabriel and J. Kajiya. Spline interpolation in curved space. In *ACM SIGGRAPH 85 course notes for State of the ArtSynthesis*. ACM Press / ACM SIGGRAPH, 1985.

[GR94a] Q. J. Ge and B. Ravani. Computer aided geometric design of motion interpolants. *ASME J. of Mechanical Design*, 116:756–762, 1994.

[GR94b] Q. J. Ge and B. Ravani. Geometric construction of Bézier motions. *ASME J. of Mechanical Design*, 116:749–755, 1994.

[Har99] E. Hartmann. On the curvature of curves and surfaces defined by normalforms. *Comp. Aided Geom. Des.*, 16:355–376, 1999.

[HHN88] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *J. Opt. Soc. Am.*, 5(7):1127–1135, July 1988.

[HJK01] D. E. Hyun, B. Jüttler, and M. S. Kim. Minimizing the distortion of affine spline motions. In *9th Pacific Conference on Computer Graphics and Applications*, pages 50–59. IEEE Press, 2001.

[HL93] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design.* A. K. Peters, 1993.

[Hor87] B. K. P. Horn. Closed form solution of absolute orientation using unit quaternions. *J. Optical Soc. A*, 4:629–642, 1987.

[HP04] M. Hofer and H. Pottmann. Energy-minimizing splines in manifolds. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004)*, 23(3):284–293, 2004.

[HPR02] M. Hofer, H. Pottmann, and B. Ravani. Subdivision algorithms for motion design based on homologous points. In J. Lenarčič and F. Thomas, editors, *Advances in Robot Kinematics*, pages 235–244. Kluwer Academic Publ., 2002.

[HPR03] M. Hofer, H. Pottmann, and B. Ravani. Geometric design of motions constrained by a contacting surface pair. *Comput. Aided Geom. Design*, 20:523–547, 2003.

[HPR04] M. Hofer, H. Pottmann, and B. Ravani. From curve design algorithms to the design of rigid body motions. *The Visual Computer*, 20(5):279–297, 2004.

[Jüt94] B. Jüttler. Visualization of moving objects using dual quaternion curves. *Computers & Graphics*, 18(3):315–326, 1994.

[JW96] B. Jüttler and M. Wagner. Computer aided design with spatial rational B–spline motions. *ASME J. Mechanical Design*, 118:193–201, 1996.

[JW02] B. Jüttler and M. Wagner. Kinematics and animation. In G. Farin, M. S. Kim, and Josef Hoschek, editors, *Handbook of Computer Aided Geometric Design*, pages 723–748. Elsevier, 2002.

[Kel99] C. T. Kelley. *Iterative Methods for Optimization.* SIAM, 1999.

[KMG98] N. Khaneja, M. I. Miller, and U. Grenander. Dynamic programming generation of curves on brain surfaces. *IEEE PAMI*, 20(11):1260–1265, 1998.

[KMS00] R. Kimmel, R. Malladi, and N. Sochen. Images as embedded maps and minimal surfaces: movies, color, texture and volumetric medical images. *Int. J. Comp. Vision*, 39:111–129, 2000.

[KN95] M. S. Kim and K. W. Nam. Interpolating solid orientations with circular blending quaternion curves. *Computer-Aided Design*, 27:385–398, 1995.

[Kob96] L. Kobbelt. A variational approach to subdivision. *Comp. Aided Geom. Design*, 13:743–761, 1996.

[KS98] L. Kobbelt and P. Schröder. A multiresolution framework for variational subdivision. *ACM Trans. Graphics*, 17(4):209–237, 1998.

[Kui02] J. B. Kuipers. *Quaternions and Rotation Sequences.* Princeton University Press, 2002.

[KWT87] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Intl. J. of Comp. Vision*, 1:321–331, 1987.

[Lat01] J. C. Latombe. *Robot Motion Planning.* Kluwer, 6 edition, 2001.

[LL02] Y. Lee and S. Lee. Geometric snakes for triangular meshes. *Computer Graphics Forum*, 21(3):229–238, 2002.

[Mal02] R. Malladi. *Geometric Methods in Bio-Medical Image Processing.* Springer, 2002.

[Mir96] B. Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools*, 1(2):31–50, 1996.

[MOW03] D. S. Meek, B. H. Ong, and D. J. Walton. Constrained interpolation with rational cubics. *Computer-Aided Design*, 20:253–275, 2003.

[MS93] H. Moreton and C. Sequin. Scale invariant minimum cost curves: fair and robust design implements. *Computer Graphics Forum*, 12:473–484, 1993.

[MS01] F. Memoli and G. Sapiro. Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces. *J. Comput. Phys.*, 173(2):730–764, 2001.

[Neu97] J. W. Neuberger. *Sobolev Gradients and Differential Equations.* Springer, 1997.

[NHP89] L. Noakes, G. Heinzinger, and B. Paden. Cubic splines on curved spaces. *IMA J. Math. Cont. & Inf.*, 6:465–473, 1989.

[Noa03] L. Noakes. Null cubics and Lie quadratics. *J. Math. Physics*, 44(3):1436–1448, 2003.

[OO88] G. Opfer and H. J. Oberle. The derivation of cubic splines with obstacles by methods of optimization and optimal control. *Numer. Math.*, 52:17–31, 1988.

[Per83] T. Lozano Perez. Spatial planning: a configuration space approach. *IEEE Trans. Comp.*, 32:108–120, 1983.

[PH04] H. Pottmann and M. Hofer. Algorithms for constrained minimization of quadratic functions — geometry and convergence analysis. Technical Report 121, Geometry Preprint Series, TU Vienna, January 2004.

[PH05] H. Pottmann and M. Hofer. A variational approach to spline curves on surfaces. *Comput. Aided Geom. Design*, 2005. to appear.

[PHYH04] H. Pottmann, Q.-X. Huang, Y.-L. Yang, and S.-M. Hu. Geometry and convergence analysis of algorithms for registration of 3D shapes. Technical Report 117, Geometry Preprint Series, TU Wien, June 2004.

[PLH02] H. Pottmann, S. Leopoldseder, and M. Hofer. Simultaneous registration of multiple views of a 3D object. *ISPRS Archives*, 34(3A):265–270, 2002.

[PR97] F. C. Park and B. Ravani. Smooth invariant interpolation of rotations. *ACM Transactions on Graphics*, 16(3):277–295, 1997.

[PR00] H. Pottmann and B. Ravani. Singularities of motions constrained by contacting surfaces. *Mech. Mach. Theory*, 35:963–984, 2000.

[PSH+04] H. Pottmann, T. Steiner, M. Hofer, C. Haider, and A. Hanbury. The isophotic metric and its application to feature sensitive morphology on surfaces. In T. Pajdla and J. Matas, editors, *Computer Vision — ECCV 2004, Part IV*, volume 3024 of *Lecture Notes in Computer Science*, pages 560–572. Springer, 2004.

[PW01] H. Pottmann and J. Wallner. *Computational Line Geometry.* Mathematics + Visualization. Springer, Heidelberg, 2001.

[RB97] R. Ramamoorthi and A. Barr. Fast construction of accurate quaternion splines. In *Proceedings of ACM SIGGRAPH 1997*, pages 287–292, New York, 1997. ACM Press / ACM SIGGRAPH.

[Rei67] C. H. Reinsch. Smoothing by spline functions. *Num. Math.*, 10:177–183, 1967.

[RL01] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. 3rd Int. Conf. on 3D Digital Imaging and Modeling*, Quebec, 2001.

[Rös98] O. Röschel. Rational motion design — a survey. *Computer-Aided Design*, 30:169–178, 1998.

[Sap01] G. Sapiro. *Geometric Partial Differential Equations and Image Analysis.* Cambridge Univ. Press, 2001.

[Sch66] D. Schweikert. An interpolating curve using a spline in tension. *J. Math. Phys.*, 45:312–317, 1966.

[Set99] J. A. Sethian. *Level Set Methods and Fast Marching Methods.* Cambridge University Press, 1999.

[Sha97] M. Sharir. Algorithmic motion planning. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 733–754. CRC Press, New York, 1997.

[Sho85] K. Shoemake. Animating rotation with quaternion curves. In *Proceedings of ACM SIGGRAPH 85*, volume 19 of *Computer Graphics*, pages 245–254. ACM, ACM Press / ACM SIGGRAPH, 1985.

[Spi75] M. Spivak. *A Comprehensive Introduction to Differential Geometry.* Publish or Perish, 1975.

[SS87] J. T. Schwartz and M. Sharir. Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves. *Int. J. Rob. Res.*, 6(2):29–44, 1987.

[Tsa02] R. Tsai. Rapid and accurate computation of the distance function using grids. *J. Comput. Phys.*, 178(1):175–195, 2002.

[Ume91] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *PAMI*, 13(4):376–380, 1991.

[VM02] T. Várady and R. Martin. Reverse engineering. In G. Farin, J. Hoschek, and M.-S. Kim, editors, *Handbook of Computer Aided Geometric Design*, pages 651–681. Elsevier, 2002.

[Wah90] G. Wahba. *Spline Models for Observational Data.* SIAM, Philadelphia, 1990.

[Wal02] J. Wallner. L^2 approximation by Euclidean motions. Technical Report 93, Institute of Geometry, May 2002.

[Wal04a] J. Wallner. Existence of set-interpolating and energy-minimizing curves. *Comput. Aided Geom. Design*, 21(9):883–892, 2004.

[Wal04b] J. Wallner. Gliding spline motions and applications. *Comput. Aided Geom. Design*, 21(1):3–21, 2004.

[WSV91] M. W. Walker, L. Shao, and R. A. Volz. Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Underst.*, 54(3):358–367, 1991.

[Zha04] H.-K. Zhao. A fast sweeping method for eikonal equations. *Math. Comp.*, 73, 2004.

[ZK98] M. Zefran and V. Kumar. Interpolation schemes for rigid body motions. *Computer-Aided Design*, 30:179–189, 1998.

Curriculum Vitae

Michael Hofer

July 19, 1975	born in Oberzeiring, Austria
Marital Status	married
Citizenship	Austrian
Talents	Initiative, creativity and endurance
Languages	German (native), English (fluent), French (basic)

Education

since Oct. 2000	Ph.D. studies in Applied Geometry Vienna University of Technology, Austria
Jul. 2000	Mag. (with honors) in Mathematics & Geometry Technical University of Graz, Austria
Oct. 1994–Jul. 2000	Studies of Mathematics & Geometry University and Technical Univ. of Graz, Austria
Aug. 1998–Jun. 1999	Studies of Mathematics University of Oklahoma, Norman, USA
Jun. 1993	Matura (with honors) Bundesrealgymnasium Judenburg, Austria

International Research Stays

Feb. 2004: Visiting research stay at School of Mathematics and Statistics (Prof. Lyle Noakes), The University of Western Australia, Perth, Australia.

Oct. 2002: Visiting research stay at Department of Computer Science (Prof. Herbert Edelsbrunner), Duke University, Durham, NC, USA.

Oct. 2001: Visiting research stay at Department of Computer Science (Prof. Peter Schröder), California Institute of Technology, Pasadena, CA, USA.

Aug. 2001: Participant at Summer School *Principles of Multiresolution in Geometric Modelling*, Munich, Germany.

Employment, Research, and Teaching Experience

since Dec. 2002:
Research associate in group of Helmut Pottmann, Department of Mathematics, Vienna University of Technology, Austria (FWF grant P16002-N05).
• Explored and implemented algorithms for geometric optimization with moving and deformable objects and variational motion design in the presence of obstacles.
• Instructor in advanced geometry course for students of architecture and basic CAD course for students of civil engineering.

Aug. 2002–Nov. 2002:
Postgraduate Researcher in group of Bahram Ravani, Department of Mechanical and Aeronautical Engineering, University of California at Davis, USA.
• Explored and implemented algorithms for constrained motion design.

Aug. 2000–Jul. 2002:
'Vertragsassistent' in group of Helmut Pottmann, Institute of Geometry, Vienna University of Technology, Austria.
• Explored and implemented algorithms for registration and motion design.
• Instructor in geometric modeling courses for students of architecture.

Oct. 1997–Jun. 1998:
'Studienassistent', Institute of Geometry, Technical Univ. of Graz, Austria
• Teaching assistant for descriptive geometry courses.

Service

Reviewer for CAGD Journal, ACM Solid Modeling Conference, Geometric Modeling and Processing Conference, Dagstuhl Seminar Proceedings.

Management of the innovative project '3D Technology' at Vienna University of Technology 2002-2004, http://www.geometrie.tuwien.ac.at/3dtechnik/.

Teaching math at the gifted kids week (Pro Talent) in Graz, July 2000. Organization of the international mathematics competition *Känguru der Mathematik* for 10.000 high school students age 10-18 in Vienna in 2001 and 2002.

Public talks presenting recent research results for general audience (University Meets Public), for continuing education courses of high school teachers of Mathematics and Geometry, and for high school students.

Scientific Talks

06. Dec. 2004: *Energy-minimizing splines in manifolds and applications*, Berlin Colloquium for Scientific Visualization, Zuse Institute Berlin, Germany.

30. Nov. 2004 *Variational motion design in the presence of obstacles*, Institute for Automation, University of Leoben, Austria.

09. Aug. 2004: *Energy-minimizing splines in manifolds*, ACM SIGGRAPH Conference, Los Angeles, USA.

01. Jul. 2004: *Variational motion design*, Conference on Advances in Robot Kinematics, Sestri Levante, Italy.

28. May 2004: *A feature sensitive metric with applications in geometric computing*, Advanced Computer Vision Colloquium, Vienna, Austria.

24. Mar. 2004: *A feature sensitive metric with applications in geometric computing*, Dagstuhl Seminar on 'Geometric Properties from Incomplete Data', Dagstuhl, Germany.

11. Feb. 2004: *Energy-minimizing splines in manifolds and applications*, Pure Math Seminar, The University of Western Australia, Perth, Australia.

16. Jan. 2004: *Splines in manifolds.* Advanced Computer Vision Colloquium, Vienna, Austria.

13. Nov. 2003: *Variational curve design in the presence of obstacles*, 8th SIAM Conference on Geometric Design and Computing, Seattle, USA.

24. Sep. 2003: *Registrierungsalgorithmen in der 3D-Technik*, Continuing Education 'Laserscanning', Vienna University of Technology, Vienna, Austria.

28. May 2003: *Recognition and Reconstruction of Special Surfaces from Scattered Data*, SampTA03 Conference, Strobl, Austria.

07. Nov. 2002: *Geometric Positioning Problems*, Duke University Algorithms Seminar, Durham, USA.

30. May 2002: *Subdivision algorithms for motion design based on homologous points*, Geometrie Tagung, Vorau, Austria.

16. Apr. 2002: *Geometric Positioning Problems*, Geometry Seminar, University of Linz, Austria.

07. Nov 2001: *Designing Smooth Motions in the Presence of Obstacles*, 7th SIAM Conference on Geometric Design and Computing, Sacramento, USA.

31. Oct. 2001: *Optimal Geometric Positioning*, Computer Science Seminar, California Institute of Technology, Pasadena, USA.

23. Nov. 2000: *Reconstruction of Rotational and Helical Surfaces for Reverse Engineering*, CMP Seminar, Prague Technical Univ., Prague, Chech Republic.

Scientific Publications

[1] M. Hofer, H. Pottmann, and B. Ravani. Subdivision algorithms for motion design based on homologous points. In J. Lenarčič and F. Thomas, editors, *Advances in Robot Kinematics*, pages 235–244. Kluwer Academic Publ., 2002.

[2] H. Pottmann, S. Leopoldseder, and M. Hofer. Approximation with active B-spline curves and surfaces. In S. Coquillart, S.-M. Hu, and H.-Y. Shum, editors, *10th Pacific Conference on Computer Graphics and Applications* (Tsinghua University, Beijing), pages 8–25. IEEE Press, 2002.

[3] H. Pottmann, S. Leopoldseder, and M. Hofer. Simultaneous registration of multiple views of a 3D object. *ISPRS Archives*, 34(3A):265–270, 2002.

[4] H. Pottmann and M. Hofer. Geometry of the squared distance function to curves and surfaces. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 223–244. Springer, 2003.

[5] M. Hofer, H. Pottmann, and B. Ravani. Geometric design of motions constrained by a contacting surface pair. *Computer Aided Geometric Design*, 20:523–547, 2003.

[6] M. Hofer and H. Pottmann. Orientierung von Laserscanner-Punktwolken. *Vermessung & Geoinformation*, 91:297–306, 2003.

[7] H. Pottmann, M. Hofer, B. Odehnal, and J. Wallner. Line geometry for 3D shape understanding and reconstruction. In T. Pajdla and J. Matas, editors, *Computer Vision — ECCV 2004, Part I*, volume 3021 of *Lecture Notes in Computer Science*, pages 297–309. Springer, 2004.

[8] H. Pottmann, T. Steiner, M. Hofer, C. Haider, and A. Hanbury. The isophotic metric and its application to feature sensitive morphology on surfaces. In T. Pajdla and J. Matas, editors, *Computer Vision — ECCV 2004, Part IV*, volume 3024 of *Lecture Notes in Computer Science*, pages 560–572. Springer, 2004.

[9] M. Hofer, H. Pottmann, and B. Ravani. From curve design algorithms to the design of rigid body motions. *The Visual Computer*, 20(5):279–297, 2004.

[10] H. Pottmann, M. Hofer, and B. Ravani. Variational motion design. In J. Lenarčič and C. Galletti, editors, *On Advances in Robot Kinematics*, pages 361–370. Kluwer, 2004.

[11] H. Pottmann, S. Leopoldseder, and M. Hofer. Registration without ICP. *Computer Vision and Image Understanding*, 95(1):54–71, 2004.

[12] M. Hofer and H. Pottmann. Energy-minimizing splines in manifolds. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004)*, 23(3):284–293, 2004.

[13] H. Pottmann, S. Leopoldseder, M. Hofer, T. Steiner, and W. Wang. Industrial Geometry: recent advances and applications in CAD. *Computer-Aided Design Appl.*, 1:513–522, 2004 (short conference version).

[14] H. Pottmann and M. Hofer. A variational approach to spline curves on surfaces. *Computer Aided Geometric Design*, to appear.

[15] H. Pottmann, S. Leopoldseder, M. Hofer, T. Steiner, and W. Wang. Industrial geometry: recent advances and applications in CAD. Computer-Aided Design, to appear (extended journal version).

Vienna, November 2004