The Visual Computer paper No. TVC02-0716

From curve design algorithms to the design of rigid body motions

Michael Hofer[1⋆], Helmut Pottmann[1], Bahram Ravani[2]

[1] Geometric Modeling and Industrial Geometry Group,
Vienna University of Technology,
Wiedner Hauptstraße 8-10/104, A-1040 Vienna, Austria
e-mail: hofer@geometrie.tuwien.ac.at,
e-mail: pottmann@geometrie.tuwien.ac.at

[2] Department of Mechanical & Aeronautical Engineering,
University of California,
1013 Academic Surge, Davis, CA 95616-5294, USA
e-mail: bravani@ucdavis.edu

Received: July 2002 / Revised version: May 2003

To the memory of Peter Steiner

Abstract. We discuss the following problem which arises in computer animation and robot motion planning: Given are N positions or keyframes $\Sigma(t_i)$ of a moving body $\Sigma \subset \mathbb{R}^3$ at time instances t_i. Compute a smooth rigid body motion $\Sigma(t)$ which interpolates or approximates the given positions $\Sigma(t_i)$, such that chosen feature points of the moving system run on smooth paths. We present an algorithm that can be considered as a transfer principle from curve design algorithms to motion design. The algorithm relies on known curve design algorithms, and on registration techniques from Computer Vision. We prove that the motion generated in this way is of the same smoothness as the curve design algorithm employed.

Keywords. Motion design – Motion planning – Registration – Subdivision algorithm – Variational design

1 Introduction

Motion design is an important and widely studied problem originating in robotics (cf. Latombe 2001, Halperin et al. 1997, Sharir 1997). Today, this active research field is concerned with e.g. camera motion design in computer animation (cf. Marchand, Courty 2002), designing the motion of digital actors (cf. Kalisiak, Panne 2001), creating realistic human and animal motions via keyframing and motion capture (cf. Arikan, Forsyth 2002), motion planning in virtual environments (cf. Salomon et al. 2003), designing molecular motions for folding and docking of proteins (cf. Song, Amato 2002), motion design of deformable objects (cf. Bayazit et al. 2002) and many more (cf. Latombe 1999).

In the present paper we are interested in *smooth* (and *fair*) motions of one rigid body with interpolation or approximation constraints. We confine ourselves to differentiable one-parameter motions in Euclidean 3-space; for an investigation of the geometric background we refer to Pottmann, Wallner 2001. For applications it is important to be able to control the smoothness level of a motion. Forces (and therefore vibrations) depend on second derivatives and jumps in the velocity would be visible in applications such as camera motions in virtual environments, computer animated objects, or architectural walkthroughs, thereby giving an unaesthetic effect.

1.1 Related work

Contributions to the solution of our type of motion design problem originated in Computer Graphics, where Shoemake 1985 was among the first to apply the techniques of Computer Aided Geometric Design for visualizing moving objects in Computer Animation. In Computer Graphics, the idea of interpolating rotations with quaternions was further developed by e.g. Barr et al. 1992, Ramamoorthi, Barr 1997, and Hanson 1998. Quaternion techniques separate the translational and rotational part of the motion. Then they can employ a curve design algorithm to the translational part in $\mathbb{R}^3$, but the rotational part has to be designed on $S^3 \subset \mathbb{R}^4$. However, nonlinear extensions of spline constructions in affine spaces to the sphere S^3 are difficult to deal with for optimization purposes. Dual quaternion curves have also been used for the visualization of moving objects, see Jüttler 1994. Affine spline motions with min-

⋆ *Correspondence to:* Michael Hofer

imal distortion have been studied by Hyun et al. 2001. A singular value decomposition projection method for interpolation on the group of Euclidean motions $SE(3)$ has been presented by Belta, Kumar 2002. Part of our work is closely related to this paper, although developed completely independent. Algorithms for *motion fairing* using the quaternion representation have been proposed by Fang et al. 1998 and Hsieh, Chang 2003, where the latter are using genetic algorithms for the solution.

To describe rigid body motions it is necessary to use *rational* rather than polynomial representations (see Röschel 1998 and the references therein). However, rational representations are much less suitable for variational design and efficient optimization techniques than polynomial ones. For smooth motions that interpolate or approximate given positions and that make use of *NURBS techniques* see Jüttler, Wagner 1996. For a recent survey on how techniques from Computer Aided Geometric Design are applied to Kinematics and Computer Animation we refer to Jüttler, Wagner 2002. The problem of *constraint* motion design has been investigated by Wallner 2003, who discussed gliding spline motions using an active motion approach resulting in near-Euclidean near-contact spline motions, and Hofer et al. 2003, who study the design of rigid body motions constraint by a contacting surface pair.

1.2 Contributions

The present paper provides the following new approach to motion design: We show how to transfer *any* curve design algorithm to motion design. Our algorithm first chooses feature points that give a good representation of the moving body. In the following steps we work with this cloud of feature points: We apply the same curve design algorithm to the sequences of homologous positions of those feature points. This yields a smooth path for each of these feature points. They do not yet correspond to a rigid body motion; however, for a linear curve design algorithm they determine an affine motion. To turn the distorted motion into a rigid body motion we use the so-called *registration* process from Computer Vision. The new concept has several advantages over previous approaches:

- Translational and rotational part of the motion are not separated in the design process
- The properties of the used curve design algorithm are transfered (within certain bounds) to the trajectories of the points of the moving body
- It is possible to perform variational motion design in an approximate way, and to deal with *motion fairing* and *motion optimization*.

A rigid body motion is a curve in the Euclidean motion group, and the fairness of that curve can be expressed in an intrinsic way with respect to the motion group (see Park, Ravani 1997). However, for the applications we have in mind, not the motion as such, but its action on a certain rigid body is employed in fairness criteria. Therefore, our formulation of 'fairness' is based on fair trajectories of chosen feature points.

Besides curve design algorithms, our motion design method uses *registration* (with known correspondences, cf. Horn 1987). This is a special case of results recently obtained by Wallner 2002, who has investigated the L^2-approximation of deformations by Euclidean motions in d-space. Geometrically, the registration of a rigid body motion to an affine motion corresponds to an *orthogonal projection* of a curve in the 12-dimensional affine space of affine transformations onto a 6-dimensional manifold, which represents the Euclidean motions. Conditions on the uniqueness of this projection and thus the feasibility of our approach follow from Wallner 2002.

The paper is organized as follows. In Sect. 2 we summarize a technique from Computer Vision for the *registration* of two point clouds with known correspondences. In Sect. 3 we present a motion design algorithm which uses curve design algorithms and registration with known correspondences. In Sect. 4 we discuss motion design using variational subdivision algorithms for curves. We conclude the paper in Sect. 5 with an outlook towards future research.

2 Registration with known correspondences

Consider two finite sequences ('point clouds') X, Y of corresponding points $\mathbf{x}^k$ and $\mathbf{y}^k$ for $k = 1, \ldots, K$. The problem of applying to one cloud, say X, a Euclidean motion m which brings each $\mathbf{x}^k$ as close as possible to $\mathbf{y}^k$ is well studied. If the meaning of 'as close as possible' is to minimize the sum of squared distances

$$\sum_{k=1}^{K} \|m(\mathbf{x}^k) - \mathbf{y}^k\|^2 \to \min, \tag{1}$$

then the solution amounts to an eigenvalue problem (see e.g. Horn 1987, Eggert et al. 1997, Wallner 2002) which we summarize below. A Euclidean motion

$$m(X) = \mathbf{R}X + \mathbf{t} \tag{2}$$

consists of a rotational part, described by the orthogonal matrix $\mathbf{R}$ with $\det \mathbf{R} = 1$, and a translational part, described by the vector $\mathbf{t}$. Let $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ be the barycenters of the point clouds X, Y, i.e.,

$$\bar{\mathbf{x}} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}^k, \qquad \bar{\mathbf{y}} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{y}^k. \tag{3}$$

If we use the barycenter of each point cloud as the origin of a new coordinate system, we get for $k = 1, \ldots, K$ the coordinates

$$\begin{aligned} \mathbf{x}^{k\prime} &= \mathbf{x}^k - \bar{\mathbf{x}} = (x_X^{k\prime}, y_X^{k\prime}, z_X^{k\prime}), \\ \mathbf{y}^{k\prime} &= \mathbf{y}^k - \bar{\mathbf{y}} = (x_Y^{k\prime}, y_Y^{k\prime}, z_Y^{k\prime}). \end{aligned} \tag{4}$$

In Horn 1987 it has been shown that the rotation $\mathbf{R}$ can be computed in the following way: Let $\mathbf{M}$ be the symmetric 4×4 matrix

$$\mathbf{M} = \begin{pmatrix} S_{xx}+S_{yy}+S_{zz} & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yz} \\ S_{yz}-S_{zy} & S_{xx}-S_{yy}-S_{zz} & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & -S_{xx}+S_{yy}-S_{zz} & S_{yz}+S_{zy} \\ S_{xy}-S_{yz} & S_{zx}+S_{xz} & S_{yz}+S_{zy} & -S_{xx}-S_{yy}+S_{zz} \end{pmatrix} \tag{5}$$

where $S_{xx}, \ldots, S_{zz}$ are the nine entries of the matrix $\sum_k \mathbf{x}^{k\prime}(\mathbf{y}^{k\prime})^T$, i.e.,

$$S_{xz} = \sum_{k=1}^{K} x_X^{k\,\prime} z_Y^{k\,\prime} \tag{6}$$

and so forth. Now compute the maximal eigenvalue λ_m of $\mathbf{M}$ and a corresponding unit eigenvector (a_0, a_1, a_2, a_3). Then the 3×3 rotation matrix $\mathbf{R}$ is given by

$$\mathbf{R} = \begin{pmatrix} a_0^2+a_1^2-a_2^2-a_3^2 & 2(a_1a_2+a_0a_3) & 2(a_1a_3-a_0a_2) \\ 2(a_1a_2-a_0a_3) & a_0^2-a_1^2+a_2^2-a_3^2 & 2(a_2a_3+a_0a_1) \\ 2(a_1a_3+a_0a_2) & 2(a_2a_3-a_0a_1) & a_0^2-a_1^2-a_2^2+a_3^2 \end{pmatrix}. \tag{7}$$

The translation $\mathbf{t}$ is the difference between the barycenter $\bar{\mathbf{y}}$ of the point cloud Y and the rotated barycenter $\mathbf{R} \cdot \bar{\mathbf{x}}$ of the point cloud X,

$$\mathbf{t} = \bar{\mathbf{y}} - \mathbf{R} \cdot \bar{\mathbf{x}}. \tag{8}$$

Remark 1 The vector (a_0, a_1, a_2, a_3) is the representation of a Euclidean motion by a *unit quaternion* $\mathbf{a} = a_0 + ia_1 + ja_2 + ka_3$, see e.g. Pottmann, Wallner 2001, p. 525.

Remark 2 The problem of finding the best transformation that minimizes the sum of squared distances between two point clouds with known correspondences is a special case of the so-called *registration problem* which originated in *Computer Vision*. In a more general setting, registration deals with the problem that partial scans of measurement points of an object have to be aligned, or a cloud of measurement points of an object has to be matched to a CAD model of that object. A standard approach to the solution of such problems is the *iterative closest point (ICP) algorithm* which has been introduced by Chen, Medioni 1992 and Besl, McKay 1992. It employs the previously discussed registration with known correspondences in each iteration step. A recent summary with new results on the acceleration of the ICP algorithm has been given by Rusinkiewicz, Levoy 2001. Pottmann et al. 2002a have presented an alternative to the closed form solution of the registration problem. This iterative algorithm linearizes the motion using instantaneous kinematics and can be used for industrial inspection or the simultaneous alignment of more than two geometric objects. For related research on registration of geometry and texture during 3D model acquisition in Computer Graphics we refer the reader to the recent survey of Bernardini, Rushmeier 2002.

2.1 Dependency of registration on the choice of corresponding points

In this section we discuss the dependency of registration with known correspondences of two point clouds X, Y on the choice of the involved points $\mathbf{x}^1, \ldots, \mathbf{x}^K$. We assume that there is an *affine* transformation such that $Y = \mathbf{A}X + \mathbf{a}$.

Definition 1 *Given a finite number of points $\mathbf{x}^1, \ldots, \mathbf{x}^K$ of unit point masses, the matrix*

$$\mathbf{J} = \sum_{k=1}^{K} \mathbf{x}^k \mathbf{x}^{kT}, \tag{9}$$

is called (in mechanics) the coordinate matrix of the inertia tensor.

Proposition 1 *If there is an* affine *transformation such that $Y = \mathbf{A}X + \mathbf{a}$, with a 3×3 matrix $\mathbf{A}$, then it follows that, in the case of* $\det \mathbf{A} > 0$, *the rotation matrix $\mathbf{R}$ can be determined by $\mathbf{R} = \mathbf{Q}_1\mathbf{Q}_2$, when $\mathbf{AJ} = \mathbf{Q}_1\mathbf{D}\mathbf{Q}_2$ is a singular value decomposition.*

A proof of Prop. 1 can be found in Wallner 2002.

Proposition 2 *Let X, Y be two clouds of corresponding points $\mathbf{x}^k$ and $\mathbf{y}^k$, $k = 1, \ldots, K$, and let Y be as in Prop. 1. Then the registration of X to Y only depends on the barycenter $\bar{\mathbf{x}}$ and the inertia tensor $\mathbf{J}$ of X.*

Proof The registration of X to Y is performed by the Euclidean motion m defined in (2). The translational part $\mathbf{t}$ of m only depends on the barycenter $\bar{\mathbf{x}}$ of X. According to Prop. 1, the rotational part $\mathbf{R}$ only depends on the inertia tensor $\mathbf{J}$. □

By a well-known result from mechanics, we can replace the points $\mathbf{x}^1, \ldots, \mathbf{x}^K$ by the six special points

$$\pm\, \tilde{\mathbf{x}}^j := \bar{\mathbf{x}} \pm \sqrt{\frac{\lambda_j}{2}}\mathbf{e}^j, \qquad j = 1, 2, 3, \tag{10}$$

without changing the barycenter and the inertia tensor of X. Thereby, $\lambda_1, \lambda_2, \lambda_3$ and $\mathbf{e}^1, \mathbf{e}^2, \mathbf{e}^3$ are the eigenvalues and eigenvectors of $\mathbf{J}$.

This implies that, if we have to perform the registration of X to several point clouds Y_i (that result from an affine transformation of X), then we only have to compute (10) once, and then we can use these 6 special points for all the individual registration computations of X to Y_i.

3 The motion design algorithm

In the following we give a detailed discussion of the individual steps of the motion design algorithm and prove that the smoothness of the generated motion is the same as the smoothness of the used curve design algorithm. Examples of motion design are presented in Sect. 4. The first step of the algorithm creates a *distorted* motion, which is an *affine* motion in the case of a *linear* curve scheme. The second step of the algorithm computes a Euclidean motion which fits the distorted motion best. The input to the algorithm are N positions $\Sigma_i := \Sigma(t_i)$ of a moving body $\Sigma \subset \mathbb{R}^3$ at time instances t_i. We want to compute a smooth motion $\Sigma(t)$ which interpolates (or approximates) the given N positions $\Sigma(t_i)$, such that chosen feature points of the moving system run on smooth paths.

3.1 Distorted motion via the curve design algorithm

Our algorithm starts by choosing a number of $K > 4$ feature points $\mathbf{f}^k, k = 1, \ldots, K$ on the moving body Σ, see Fig. 1. We compute their locations $\mathbf{f}_i^k$ at the given time instances t_i, see Fig. 2a. Positions $\mathbf{f}_1^k, \mathbf{f}_2^k, \ldots, \mathbf{f}_N^k$ of the same feature point $\mathbf{f}^k$ at different time instances t_i are called *homologous points*. To each of the K sequences of homologous points we apply the chosen C^k curve design algorithm. An interpolating (approximating, resp.) curve design algorithm will later yield an interpolating (approximating, resp.) motion. We get K curves $\mathbf{f}^1(t), \ldots, \mathbf{f}^K(t)$ which we refer to as *feature curves*, see Fig 2b. For each t the K points $\mathbf{f}^1(t), \ldots, \mathbf{f}^K(t)$ may be considered as image points of $\mathbf{f}^1, \ldots, \mathbf{f}^K$ under some deformation.

In fact, the distortion is not determined on all points of the moving body, but just on the feature points. However, it will be convenient to speak of a distorted copy $\Sigma'(t)$ of the moving body. By applying the same curve design algorithm to all feature points we obtain a time dependent family of distorted copies $\Sigma'(t)$ of the body Σ, a so-called *distorted motion*, see Fig. 2c. Since we want to design a Euclidean motion, in the second step of the algorithm we use registration to find the best fit, in the least squares sense, of the rigid moving body Σ to $\Sigma'(t)$, see Fig. 2d.

Remark 3 If we use a subdivision algorithm we can apply several steps of the algorithm to create a sufficiently dense set of discrete positions of the moving body. Those might be sufficient for the application in mind, or they can easily be interpolated with standard motion design techniques without caring about desired properties of the motion (interpolating, approximating, smoothness, fairness, ...) anymore.

If we use the same *linear* curve design algorithm for all feature points, then it is not necessary to apply it to *all* sequences of homologous points. This is so, since we obtain *affine copies* of the moving body as intermediate positions. A proof of this property relies on the *linearity* of the used curve design algorithm; it says that each intermediate position $\mathbf{f}^k(t)$ is a linear combination of $\mathbf{f}_1^k, \ldots, \mathbf{f}_N^k$. Hence, at a time instance t the position $\mathbf{f}^k(t)$ of the k-th feature point $\mathbf{f}^k$ is of the form

$$\mathbf{f}^k(t) = \sum_j \lambda_j (\mathbf{a}_j + \mathbf{A}_j \cdot \mathbf{f}^k). \tag{11}$$

Here, we have already used the fact that the j-th homologous position $\mathbf{f}_j^k$ results from an initial position $\mathbf{f}^k$ by an affine map. After reordering, we obtain

$$\begin{aligned} \mathbf{f}^k(t) &= \sum_j \lambda_j \mathbf{a}_j + (\sum_j \lambda_j \mathbf{A}_j) \cdot \mathbf{f}^k \\ &=: \mathbf{b} + \mathbf{B} \cdot \mathbf{f}^k. \end{aligned} \tag{12}$$

We see that the inserted position results from the initial one by application of an affine map $\alpha(t)$ with matrix $\mathbf{B}$ and translational part $\mathbf{b}$.

This property allows us to apply the curve design algorithm to only four sequences of non coplanar homologous points. Using the affine maps, we then compute intermediate positions also for the remaining feature points. This is necessary since the following registration depends on all feature points.

If we use a nonlinear curve design algorithm we do no longer get affine copies and thus have to apply the curve design algorithm to *all* sequences of homologous points. In the subsequent registration step, it does not matter whether the curve design algorithm produces affine copies of the original point cloud or not.

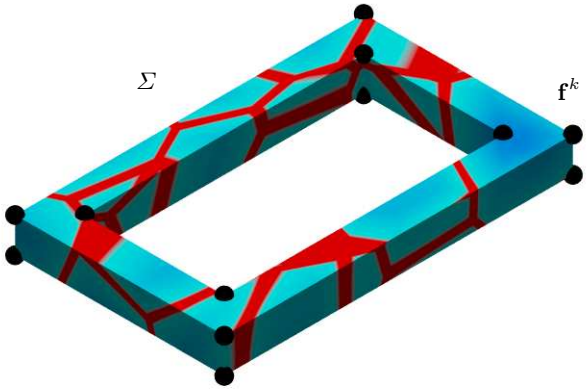


Fig. 1 A rigid body Σ with 16 feature points $\mathbf{f}^k$.

3.2 Euclidean motion via registration

So far we have computed a motion (by distortions) of the moving body, which is an affine motion in the case

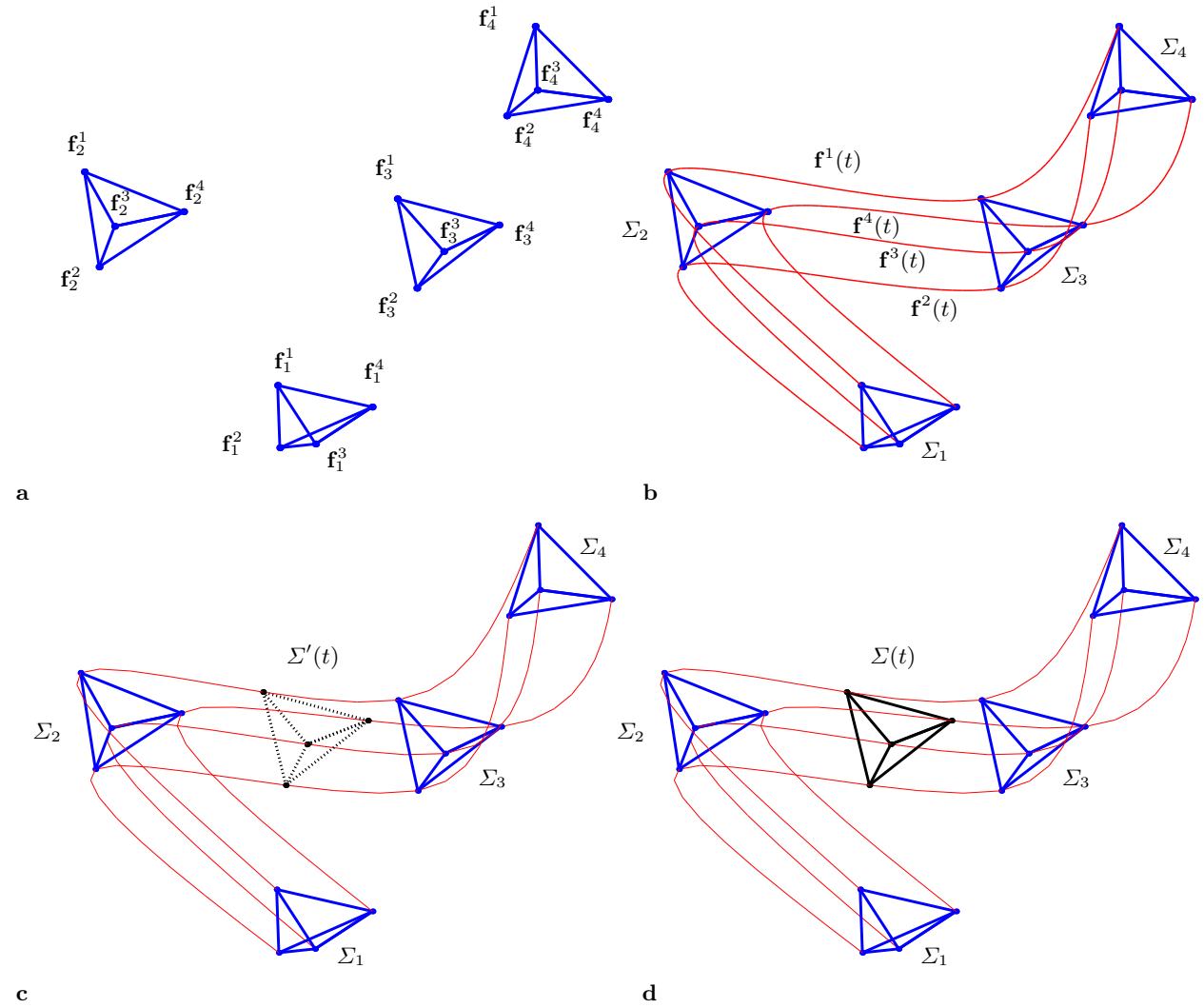


Fig. 2 **a** The feature points $\mathbf{f}^k$ in their homologous positions $\mathbf{f}_i^k$ representing the moving body in the input positions $\Sigma_1, \ldots, \Sigma_4$. **b** Feature curves $\mathbf{f}^1(t), \ldots, \mathbf{f}^K(t)$ computed by the curve design algorithm. **c** Resulting distorted version $\Sigma'(t)$ of the moving body at a time instance t. **d** We compute the best fit of the Euclidean body Σ to $\Sigma'(t)$ at each time instance t via registration.

of a linear curve scheme. Now we discuss how to fit a Euclidean motion to this distorted motion.

At each time instance t we register the moving body Σ to the distorted version $\Sigma'(t)$ in order to find an intermediate position resulting from a rigid body motion. Registration with known correspondences (presented in Sect. 2) computes a Euclidean motion m, which brings the feature points $\mathbf{f}^1, \ldots, \mathbf{f}^K$ as close as possible to the previously computed points $\mathbf{f}^1(t), \ldots, \mathbf{f}^K(t)$. Thereby the sum of squared distances

$$\sum_{k=1}^{K} \|m(\mathbf{f}^k) - \mathbf{f}^k(t)\|^2 \tag{13}$$

is minimized. In this way we compute the approximation of a distorted motion by a Euclidean motion.

3.3 Smoothness of the resulting motion

3.3.1 Smoothness of the resulting motion in the general case 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 feature point, does our motion design algorithm produce a C^k motion? We formulate the following theorem.

Theorem 1 *If we use a C^k curve design algorithm, then our motion design algorithm generates a C^k motion, provided that the maximal eigenvalue $\lambda_m(t)$ of the matrix $\mathbf{M}(t)$ in (5) has multiplicity one.*

Proof The first step of our algorithm generates for all t a distorted copy $\Sigma'(t)$ of the moving body Σ. We already know that every feature point runs on a C^k path.

Then, for all t we register the rigid moving body Σ to $\Sigma'(t)$. We have to prove that the registration, described in Sect. 2, is a C^k operation. In order to find the Euclidean motion that performs the registration we have to compute the eigenvector $\mathbf{x}_m(t)$ corresponding to the maximal eigenvalue $\lambda_m(t)$ of the matrix $\mathbf{M}(t)$ described by (5).

The eigenvalues of $\mathbf{M}(t)$ are the zeros of the quartic polynomial $\det(\mathbf{M}(t) - \lambda(t) \cdot \mathbf{I})$, with the identity matrix $\mathbf{I}$. Eigenvectors $\mathbf{x}_m(t)$ corresponding to the maximal positive eigenvalue $\lambda_m(t)$ are found by solving the homogeneous linear system of equations $(\mathbf{M}(t) - \lambda_m(t) \cdot \mathbf{I})\mathbf{x}_m(t) = 0$. From a unit eigenvector $\mathbf{x}_m(t)$ we find the rotation matrix $\mathbf{R}(t)$ in (7). 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 $\mathbf{M}(t)$ consists of entries that are polynomial in the coordinates of the points $\mathbf{x}_i$ and $\mathbf{y}_i$. Thus the function $\mathbf{M}(t)$ is clearly C^k. Similarly, the computation of $\mathbf{R}$ from (a_0, a_1, a_2, a_3) is polynomial. The maximal eigenvalue of $\mathbf{M}(t)$ is smoothly dependent on $\mathbf{M}(t)$ if we can solve the equation

$$\det(\mathbf{M}(t) - \lambda_m(t) \cdot \mathbf{I}) = 0 \tag{14}$$

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

$$\frac{\partial}{\partial \lambda} \det(\mathbf{M} - \lambda \cdot \mathbf{I}) \neq 0 \quad \text{for} \quad \lambda = \lambda_m, \tag{15}$$

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

Lemma 1 *Suppose that $\mathbf{A}(t)$ and $\lambda(t)$ are C^k functions. If for all t in some interval $[a, b]$, $\lambda(t)$ is a single eigenvalue of the matrix $\mathbf{A}(t)$, then there is a C^k function $\mathbf{x}(t)$, defined in the same interval, which is a unit eigenvector of $\mathbf{A}(t)$ to the eigenvalue $\lambda(t)$.*

This result is well-known and easy to show. For the convenience of the reader we write down a proof.

Proof (of Lemma 1) Note that differentiability is a local property. We consider t in a neighborhood of some t_0. The rank of the matrix $(\mathbf{A} - \lambda \cdot \mathbf{I})$ equals $n - 1$. Thus, by removing one row and one column of this matrix we get a $(n-1) \times (n-1)$-matrix $\mathbf{B}$ with $\det(\mathbf{B}) \neq 0$. Since $\det(\mathbf{B}(t))$ is continuous it does not vanish in a neighborhood of t_0. Hence, for all t_0 there is a neighborhood where the same submatrix $\mathbf{B}$ is regular. Now we use this matrix $\mathbf{B}$ to solve the linear system of equations $(\mathbf{A} - \lambda \cdot \mathbf{I})\mathbf{x} = 0$. Without loss of generality let $\mathbf{B}$ consist of the first $n - 1$ rows and the first $n - 1$ columns of $(\mathbf{A} - \lambda \cdot \mathbf{I}) =: (c_{ij})$. We first solve for $(x_1, \ldots, x_n)$ with $x_n = 1$ which means that we have to solve

$$\mathbf{B}(x_1, \ldots, x_{n-1})^T = (-c_{1n}, \ldots, -c_{n-1,n})^T. \tag{16}$$

Hence the solution vector is found to be

$$(x_1, \ldots, x_{n-1})^T = \mathbf{B}^{-1}(-c_{1n}, \ldots, -c_{n-1,n})^T. \tag{17}$$

This computation did not use the n-th equation of the linear system, but that one is automatically fulfilled, since we know that $(\mathbf{A} - \lambda \cdot \mathbf{I})$ has rank $n - 1$. The cofactor formula for computation of $\mathbf{B}^{-1}$ shows that $\mathbf{B}^{-1}$ is C^k if $\mathbf{B}$ is C^k. Thus, the eigenvector $(x_1, \ldots, x_{n-1}, 1)$ is C^k. By normalizing we get a C^k 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 $\mathbf{x}(t)$ which do not fit together. This is easily remedied by replacing a finite number of locally defined $\mathbf{x}(t)'s$ by their opposites $-\mathbf{x}(t)$. □

This completes the proof that, by using a C^k curve design algorithm, our motion design algorithm generates a C^k motion. □

3.3.2 Smoothness of the resulting motion for linear curve schemes By applying the same linear curve design algorithm to all feature points we obtain a time dependent family of affine copies $\Sigma'(t)$ of the body Σ, 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 Σ, we may associate a point in 12-dimensional affine space A^{12}. To get its coordinates, we may just collect the coordinates of the image points of four independent feature points under the affine mapping under consideration. The image points of Euclidean motions form a 6-dimensional submanifold $M^6 \subset A^{12}$. The input positions $\Sigma(t_i)$ correspond to points s_i on M^6. The affine motion $\Sigma'(t)$ corresponds to a curve $\mathbf{c}'(t)$ which interpolates or approximates the points s_i, but does not lie on M^6 (see Fig. 3).

We can measure the distance between two affine maps by means of the sum of squared distances between the images of selected feature points. This is equivalent to the introduction of a Euclidean metric in A^{12} (see also Pottmann, Peternell 2000). Thus, we have an orthogonality in A^{12}. The way in which we compute a rigid body motion $\Sigma(t)$ from the affine motion $\Sigma'(t)$ 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 $\mathbf{M}(t)$ (and therefore the dimension of the eigenspaces) there are four different footpoints from a point $\mathbf{c}'(t)$ on the manifold M^6 or infinitely many.

In Wallner 2002 the following result is shown: Consider the matrix $\mathbf{A}_j$ in the description of the affine map which generates $\Sigma'(t)$ and has $\mathbf{c}'(t)$ as image point in A^{12}; if this matrix $\mathbf{A}_j$ has a positive determinant, we can be sure that the conditions of Thm. 1 are fulfilled. Geometrically this means that $\mathbf{c}'(t)$ is not on the medial axis of M^6 and thus it possesses a unique closest point $\mathbf{c}(t)$ on M^6.

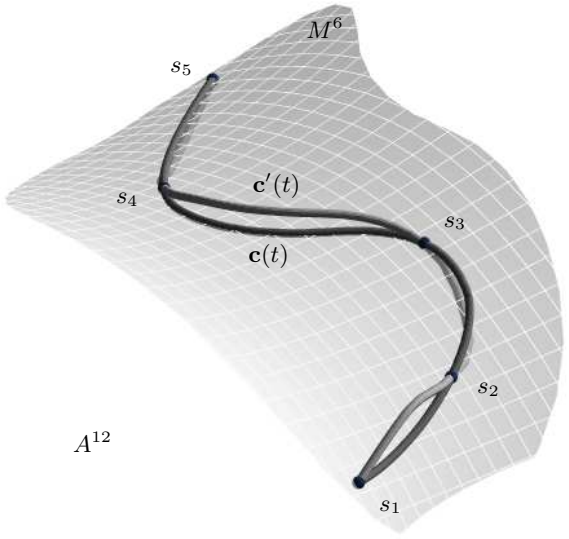


Fig. 3 An affine motion corresponds to a curve $\mathbf{c}'(t)$ in A^{12} and the designed motion $\mathbf{c}(t)$ is the orthogonal projection of $\mathbf{c}'(t)$ onto the 6-dimensional manifold M^6 of Euclidean motions.

3.4 Remarks on the algorithm

The input data may be subject to measurement errors, which of course depend on the device they have originally been obtained with. If a CAD model of the moving body is known, then the initial input positions may also be corrected by using registration. If a precise model of the moving body is unknown, the computation of a copy of the moving body via registration of the positions against each other has to be done in a preprocessing step. Note also that the use of feature points may be advantageous for the design of motions where the given positions have been captured by methods of Computer Vision.

4 Motion design with variational subdivision

In this section we give several examples in which the presented algorithm is applied. We base motion design on a variational subdivision scheme for curves which intends to minimize the change in velocity. This is interesting for applications in computer animation and robotics, where sudden speeding up or slowing down is usually undesirable. Note that the resulting motions are only approximate variational.

First we review known interpolatory variational subdivision for curves and extend these schemes to approximating ones, where one can control the interpolation or approximation of each input point by a parameter. Then we use these curve design algorithms for the design of interpolating or approximating rigid body motions which we illustrate at hand of several examples. Finally, we mention how to achieve shape modifications such as tension effects and the use of local subdivision schemes.

4.1 Interpolatory variational subdivision for curves

Interpolatory variational subdivision has been introduced by Kobbelt, 1996 and involves the minimization of some quadratic energy functional in order to control the fairness of the curves which are constructed. Kobbelt, Schröder 1998 have extended variational subdivision from the *uniform* to the *non-uniform* parameter setting and discussed it in a multiresolution framework.

The subdivision scheme requires a sequence of points $(\mathbf{f}_i)$ as input data. In the first step these points are connected to a piecewise linear curve. Then by minimizing a quadratic fairness functional, we iteratively insert new points, which results in a smooth limit curve. Let each point $\mathbf{f}_i$ of our polygon correspond to a parameter value t_i. In our application the t_i's will most likely be given, since they are related to the timing of the motion. In case we have to estimate t_i, we may take it from a *centripetal parametrization*, i.e.,

$$t_{i+1} - t_i := \sqrt{\|\mathbf{f}_{i+1} - \mathbf{f}_i\|}. \tag{18}$$

In order to obtain an estimate of the jump in velocity at a certain time instance, the subdivision scheme we are using is based on the following numerical differentiation rule: Given are three values v_0, v_1, v_2. Then the coefficients $c_0, c_1, c_2 \in \mathbb{R}$ of the quadratic interpolating polynomial $P_2(t) = c_0 + c_1 t + c_2 t^2$ such that $P_2(t_i) = v_i$ $(i = 0, 1, 2)$ can be computed easily. The second derivative $P_2''(t) = 2c_2$ is constant and can be used as a numerical estimate for the change in velocity at t_1. The coefficient c_2 is found to be a linear combination of v_0, v_1, v_2,

$$c_2 = \frac{v_0}{\Delta_0(\Delta_1 + \Delta_0)} - \frac{v_1}{\Delta_1 \Delta_0} + \frac{v_2}{\Delta_1(\Delta_1 + \Delta_0)}, \tag{19}$$

where $\Delta_i = t_{i+1} - t_i$. If we rearrange these terms we see that c_2 describes the *second divided difference* for a *non-uniform* parametrization. For a *uniform* parametrization with $\Delta_i = 1$, c_2 corresponds to the *second forward difference*

$$2c_2 = \Delta^2 v_0 = v_0 - 2v_1 + v_2. \tag{20}$$

By minimizing the function

$$\sum \|c_2\|^2 \to \min \tag{21}$$

we find the variational subdivision scheme in the uniform and non-uniform parameter setting presented in Kobbelt, 1996 and Kobbelt, Schröder 1998.

Let us derive explicit formulae for the uniform setting (the non-uniform case is completely analogous). In each iteration step we insert new points $\mathbf{q}_i$ between the given points $\mathbf{f}_i$ from the previous iteration step, see Fig. 4a. In the *uniform* parameter setting for *open* polygons the

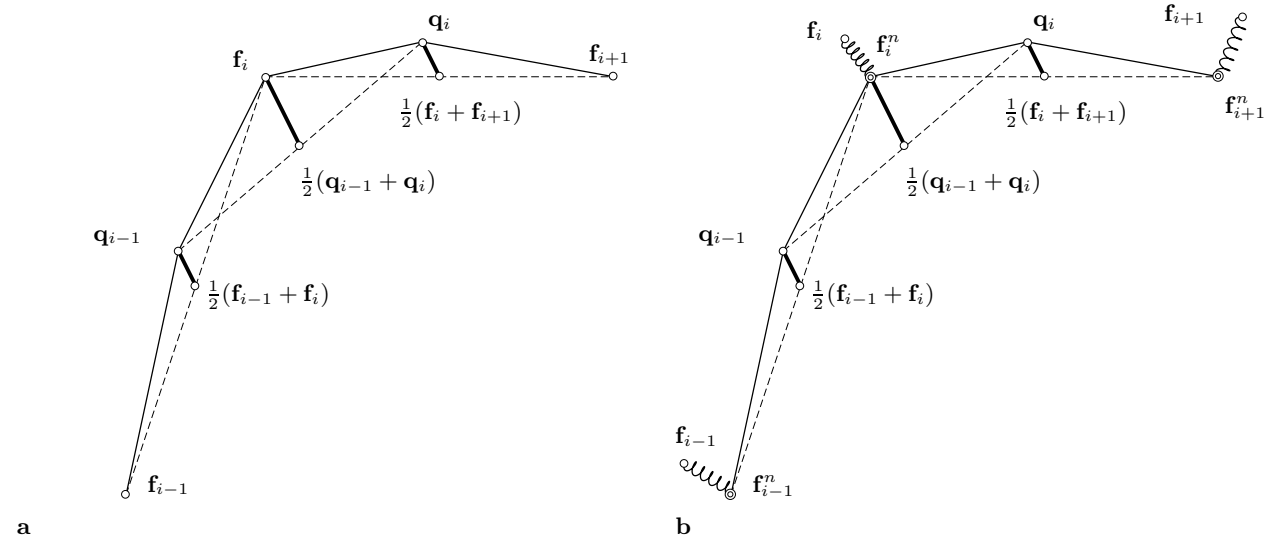


Fig. 4 Uniform **a** *interpolatory* and **b** *approximating* variational subdivision.

position of the new points $\mathbf{q}_i$ is found by minimizing the objective function

$$F(\mathbf{q}_1, \ldots, \mathbf{q}_{N-1}) = \sum_{i=1}^{N-1} \|\mathbf{f}_i - 2\mathbf{q}_i + \mathbf{f}_{i+1}\|^2 + \sum_{i=1}^{N-2} \|\mathbf{q}_i - 2\mathbf{f}_{i+1} + \mathbf{q}_{i+1}\|^2. \quad (22)$$

F is *quadratic* in the unknowns $\mathbf{q}_i \in \mathbb{R}^d$. The minimization of F can be computed by letting the partial derivatives of F with respect to the unknowns $\mathbf{q}_i$ equal to zero, which leads to a tridiagonal linear system of equations,

$$\underbrace{\begin{pmatrix} 5 & 1 & & & \\ 1 & 6 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 6 & 1 \\ & & & 1 & 5 \end{pmatrix}}_{:=\mathbf{A}} \cdot \begin{pmatrix} q_1 \\ \vdots \\ q_{N-1} \end{pmatrix} = \underbrace{\begin{pmatrix} 2 & 4 & & & \\ & 4 & 4 & & \\ & & \ddots & \ddots & \\ & & & 4 & 4 \\ & & & & 4\ 2 \end{pmatrix}}_{:=\mathbf{B}} \begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix}. \quad (23)$$

The matrices $\mathbf{A}$ and $\mathbf{B}$ are of size $(N-1)\times(N-1)$ and $(N-1)\times N$, respectively. Non-bold q_i and f_i denote one coordinate of $\mathbf{q}_i$ and $\mathbf{f}_i$.

In the *uniform* parameter setting for *closed* polygons, the position of the new points $\mathbf{q}_i$ is found by minimizing the objective function

$$F(\mathbf{q}_1, \ldots, \mathbf{q}_N) = \sum_{i=1}^{N} \|\mathbf{f}_i - 2\mathbf{q}_i + \mathbf{f}_{i+1}\|^2 + \sum_{i=1}^{N} \|\mathbf{q}_i - 2\mathbf{f}_{i+1} + \mathbf{q}_{i+1}\|^2, \quad (24)$$

where we identify $\mathbf{f}_{N+1} \equiv \mathbf{f}_1$ and $\mathbf{q}_{N+1} \equiv \mathbf{q}_1$. Minimizing (24) leads to solving the following linear system of N equations:

$$\underbrace{\begin{pmatrix} 6 & 1 & & & 1 \\ 1 & 6 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 6 & 1 \\ 1 & & & 1 & 6 \end{pmatrix}}_{N\times N} \begin{pmatrix} q_1 \\ \vdots \\ q_N \end{pmatrix} = \underbrace{\begin{pmatrix} 4\ 4 & & & \\ & \ddots & \ddots & \\ & & & 4\ 4 \end{pmatrix}}_{N\times(N+1)} \begin{pmatrix} f_1 \\ \vdots \\ f_N \\ f_1 \end{pmatrix}.$$

The presented refinement scheme for open and closed curves is global, i.e., every new point depends on all points of the polygon to be refined. Interpolation is guaranteed since the old points belong to the newly calculated finer version. Kobbelt, 1996 has shown that these schemes generate at least C^2 curves. Thus, according to Thm. 1, we can use these schemes for variational motion design of a C^2 motion. The variational subdivision scheme in the *non-uniform parameter setting* also leads to the solution of a linear system of equations.

4.2 Approximating variational subdivision for curves

The previously discussed variational subdivision scheme is an interpolating one. The given points $\mathbf{f}_i$ remain unchanged. However, we can pick some or all of these points and allow a change as well. For those points which may be changed, we add a term $\lambda_i(\mathbf{f}_i - \mathbf{f}_i^n)^2$ to the functional (22) to be minimized. It may be seen as placing a spring between the given point $\mathbf{f}_i$ and the new location $\mathbf{f}_i^n$, see Fig. 4b. The influence of the spring is governed by the

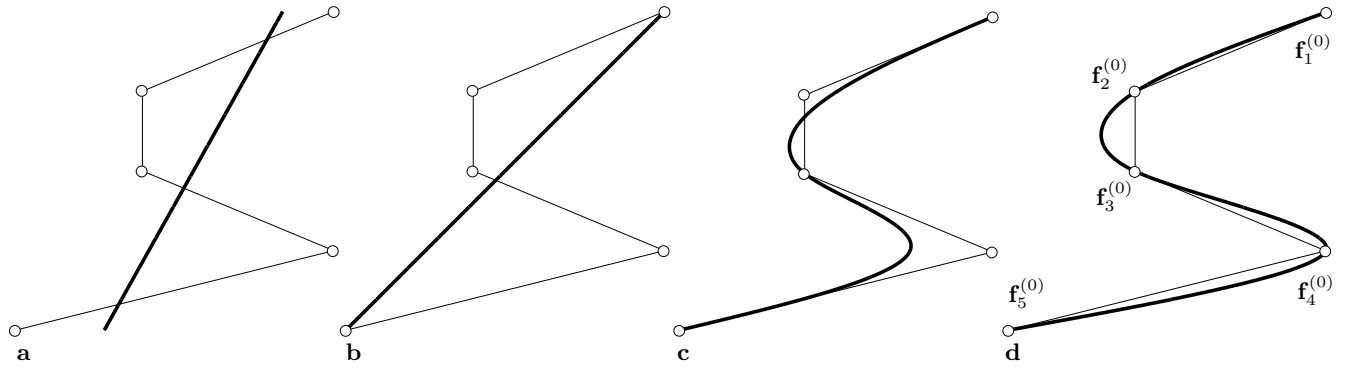


Fig. 5 Uniform approximating variational subdivision for curves. Shown are the input polygon (thin) consisting of 5 points $\mathbf{f}_1^{(0)},\dots,\mathbf{f}_5^{(0)}$ and the refined polygon (bold) after 4 subdivision steps consisting of 65 points. **a** $\lambda_i = 0.01$ for $i = 1,\dots,N_m$. **b** $\lambda_1 = \lambda_{N_m} = 10^8$, $\lambda_i = 0.01$ for $i = 2,\dots,N_m - 1$. **c** $\lambda_1 = \lambda_{2\cdot 2^m+1} = \lambda_{N_m} = 10^8$, $\lambda_{2^m+1} = 0.1$, $\lambda_{3\cdot 2^m+1} = 2.2$ and $\lambda_i = 1$ for all other i. **d** $\lambda_i = 10^8$ for $i = 1,\dots,N_m$; $m = 1,2,3,4$.

real number $\lambda_i > 0$. The functional to be minimized is again quadratic, namely a discretization of the functional used for smoothing splines, see e.g. Wahba 1990. If all $\mathbf{f}_i$ are allowed to change their position corresponding to a parameter $\lambda_i > 0$, then the functional to be minimized is given by

$$F(\mathbf{q}_1,\dots,\mathbf{q}_{N-1},\mathbf{f}_1^n,\dots,\mathbf{f}_N^n) = \sum_{i=1}^{N-1} \|\mathbf{f}_i^n - 2\mathbf{q}_i + \mathbf{f}_{i+1}^n\|^2 + \sum_{i=1}^{N-2} \|\mathbf{q}_i - 2\mathbf{f}_{i+1}^n + \mathbf{q}_{i+1}\|^2 + \sum_{i=1}^{N} \lambda_i \|\mathbf{f}_i - \mathbf{f}_i^n\|^2. \quad (25)$$

The minimization of (25) leads to a linear system of equations $\mathbf{A}^a\mathbf{x} = \mathbf{b}$, where we collect the unknowns in a $2N-1$ vector $\mathbf{x} = (\mathbf{q}_1,\dots,\mathbf{q}_{N-1},\mathbf{f}_1^n,\dots,\mathbf{f}_N^n)^T$. The $(2N-1)\times(2N-1)$ symmetric coefficient matrix $\mathbf{A}^a$ has a band structure, is sparse and can be described as a block matrix, $\mathbf{b}$ is a $2N-1$ vector:

$$\mathbf{A}^a = \left(\begin{array}{c|c} \mathbf{A} & -\mathbf{B} \\ \hline -\mathbf{B}^T & \mathbf{L} \end{array}\right),\ \mathbf{b} = (\underbrace{0,\dots,0}_{N-1},\lambda_1 f_1,\dots,\lambda_N f_N)^T. \quad (26)$$

Non-bold f_i denote one coordinate of the vector $\mathbf{f}_i$. The block matrices $\mathbf{A}$ and $\mathbf{B}$ in (26) are the $(N-1)\times(N-1)$ and $(N-1)\times N$ matrices of (23), and the $N\times N$ block matrix $\mathbf{L}$ is given by

$$\mathbf{L} := \begin{pmatrix} 1+\lambda_1 & 1 & & & \\ 1 & 6+\lambda_2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 6+\lambda_{N-1} & 1 \\ & & & 1 & 1+\lambda_N \end{pmatrix}. \quad (27)$$

In the remainder of this section we denote the N_0 points of the input polygon by $\mathbf{f}_1^{(0)},\dots,\mathbf{f}_{N_0}^{(0)}$. After m subdivision steps we have $N_m := (N_0-1)2^m + 1$ points $\mathbf{f}_1^{(m)},\dots,\mathbf{f}_{N_m}^{(m)}$, and the input point $\mathbf{f}_i^{(0)}$ corresponds to the point $\mathbf{f}_{(i-1)2^m+1}^{(m)}$ of the refined sequence.

Note that, if a spring is weak, i.e., if we let $\lambda_i \to 0$, then this point is 'neglected' and only the remaining points are taken into consideration for the shape of the final curve. Thus, if we let $\lambda_i \to 0$ for all i, then the limit curve will just be a straight line approximating the input points, see Fig. 5a.

For $\lambda_i \to \infty$ the corresponding input point $\mathbf{f}_i^{(0)}$ is interpolated by the final curve. It is common use in geometric curve design that the first and last input point are interpolated by the final curve. Thus the first and the last spring are assumed to be very strong ($\lambda_1, \lambda_{N_m} \to \infty$) in every subdivision step. If all other springs are weak in every subdivision step ($\lambda_2,\dots,\lambda_{N_m-1} \to 0$), then the limit curve is a straight line connecting the first and last input point, see Fig. 5b.

If the intermediate springs $\lambda_{(i-1)2^m+1}$ corresponding to the given points $\mathbf{f}_i^{(0)}$ are chosen constant in every subdivision, we get interpolation of $\mathbf{f}_i^{(0)}$ for $\lambda_{(i-1)2^m+1} \to \infty$ and approximation of $\mathbf{f}_i^{(0)}$ if $0 < \lambda_{(i-1)2^m+1} < \infty$, $i = 2,\dots,N_0-1$. For all other λ_j in every subdivision step we assume $0 < \lambda_j < \infty$. Fig. 5c shows an example where $\mathbf{f}_1^{(0)}, \mathbf{f}_3^{(0)}, \mathbf{f}_5^{(0)}$ are interpolated and $\mathbf{f}_2^{(0)}, \mathbf{f}_4^{(0)}$ are approximated by the refined polygon.

If we let $\lambda_i \to \infty$ for all i, then we get in the limit an interpolating curve, see Fig. 5d. In that case we recommend to use the original interpolatory variational subdivision scheme.

Fig. 6 illustrates an example where we show the first to fourth subdivision step of the approximating variational subdivision algorithm for curve design applied to 8 input points. The input points $\mathbf{f}_1^{(0)}, \mathbf{f}_4^{(0)}, \mathbf{f}_8^{(0)}$ are interpolated and the remaining input points $\mathbf{f}_i^{(0)}, i = 2,3,5,6,7$ are approximated by the refined polygon with $\lambda_i = 5$.

Remark 4 It is straightforward to extend this approximating scheme to closed polygons and to the nonuniform case. All presented algorithms can be employed to input points in $\mathbb{R}^d$.

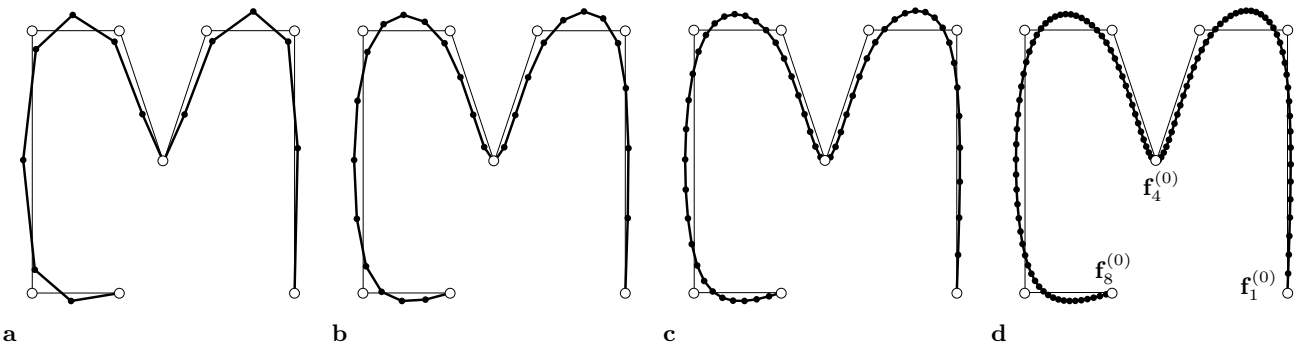


Fig. 6 Uniform approximating variational subdivision of the input polygon (thin) consisting of 8 points $\mathbf{f}_1^{(0)}, \dots, \mathbf{f}_8^{(0)}$. **a-d** show the refined polygon after $m = 1, \dots, 4$ subdivision steps (bold) with $\lambda_1 = \lambda_{3 \cdot 2^m + 1} = \lambda_{N_m} = 10^8$ and $\lambda_i = 5$ for all other i.

4.3 Interpolatory and approximating design of rigid body motions

In this section we employ in several examples the curve design algorithms of Sect. 4.1 and Sect. 4.2 in the motion design algorithm presented in Sect. 3. Given are N input positions of the moving body Σ at time instances $t_i, i = 1, \dots, N$. Since both, the interpolatory and the approximating variational subdivision algorithm for curve design are *linear* schemes, we use the following strategy (see also Fig. 2): Given a moving body Σ represented by K feature points $\mathbf{f}^1, \dots, \mathbf{f}^K$, choose 4 affinely independent points, denoted by $\hat{\mathbf{f}}^1, \dots, \hat{\mathbf{f}}^4$, and compute in a preprocessing step

- the barycenter $\bar{\mathbf{f}}$ of $\mathbf{f}^1, \dots, \mathbf{f}^K$
- the inertia tensor $\mathbf{J}$ of $\mathbf{f}^1, \dots, \mathbf{f}^K$, see (9)
- the 6 special points $\tilde{\mathbf{f}}^1, \dots, \tilde{\mathbf{f}}^6$ using $\bar{\mathbf{f}}$ and $\mathbf{J}$, see (10)
- the barycentric coordinates of $\tilde{\mathbf{f}}^1, \dots, \tilde{\mathbf{f}}^6$ with respect to $\hat{\mathbf{f}}^1, \dots, \hat{\mathbf{f}}^4$.

For the actual motion design employ several steps of the variational subdivision algorithm to the points $\hat{\mathbf{f}}^1, \dots, \hat{\mathbf{f}}^4$ in their homologous positions $\hat{\mathbf{f}}_i^1, \dots, \hat{\mathbf{f}}_i^4$. This gives the four feature curves $\hat{\mathbf{f}}^1(t), \dots, \hat{\mathbf{f}}^4(t)$. At every time instance t, the four points $\hat{\mathbf{f}}^1(t), \dots, \hat{\mathbf{f}}^4(t)$ determine the affine map $\alpha_t : \hat{\mathbf{f}}^k \mapsto \hat{\mathbf{f}}^k(t), k = 1, \dots, 4$. The affine images $\tilde{\mathbf{f}}^j(t) := \alpha_t(\tilde{\mathbf{f}}^j)$, $j = 1, \dots, 6$, are then found using the barycentric coordinates (computed in the preprocessing) with respect to $\hat{\mathbf{f}}^1(t), \dots, \hat{\mathbf{f}}^4(t)$. Finally, at every time instance t we register the point cloud $\tilde{\mathbf{f}}^1, \dots, \tilde{\mathbf{f}}^6$ to the point cloud $\tilde{\mathbf{f}}^1(t), \dots, \tilde{\mathbf{f}}^6(t)$ which results in a discrete sequence of Euclidean positions $\Sigma(t)$ of the moving body Σ. After the m-th subdivision step $2^m - 1$ intermediate positions have been inserted between each two adjacent given positions Σ_i and Σ_{i+1}. Given N input positions, the total number of positions F for the resulting motion after the m-th subdivision step is given by

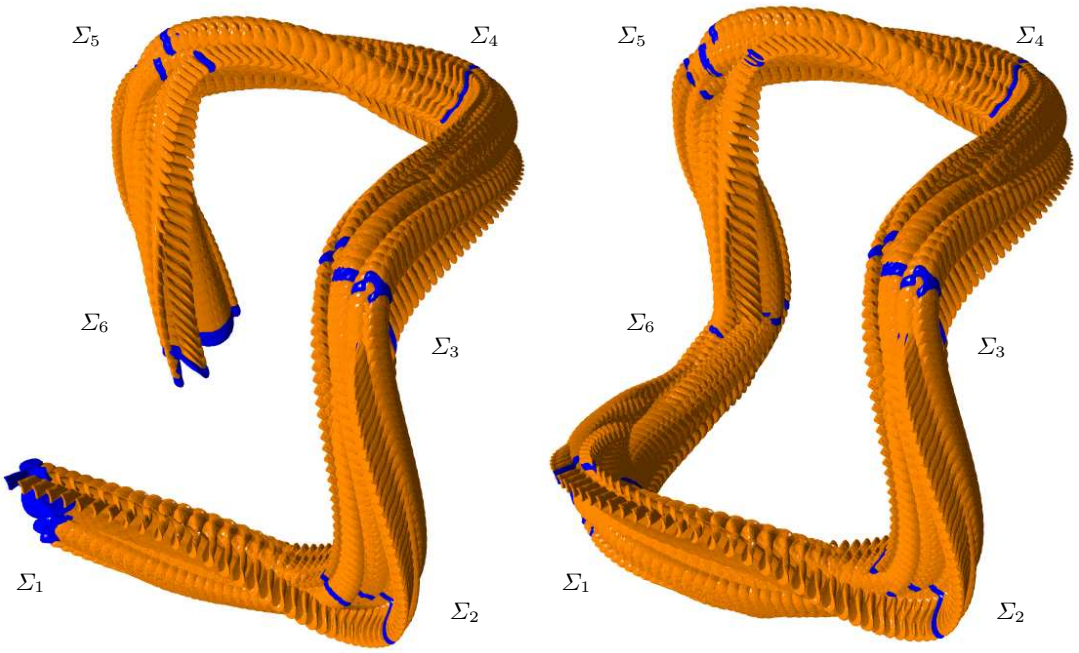
- open motion: $F = 2^m(N-1) + 1$
- closed motion: $F = 2^m N$.

If the time instances t_i corresponding to the input positions Σ_i are chosen uniformly as $t_i = i$, then, after the m-th subdivision step, we know a position of the final motion every $1/2^m s$. Fig. 7 shows for $m = 5$ a C^2 uniform open and closed rigid body motion of the Stanford bunny interpolating 6 given positions. The employed curve design algorithm is the uniform variational subdivision scheme in the open (22) and closed (24) case.

If we want to minimize the change in velocity of our motion we apply the variational subdivision scheme in the non-uniform parameter setting. The parametrization we use is a centripetal parametrization (18) derived with help of the image points s_i in A^{12}. Fig. 8 presents a non-uniform C^2 rigid body motion of the Utah teapot interpolating 5 input positions. Fig. 9a shows a Euclidean body motion of a rigid body computed with interpolatory variational subdivision in the uniform and non-uniform parameter setting. The acceleration of these two motions has been estimated by the arithmetic mean of the squared acceleration of the feature curves used in the computation. In Fig. 9b we compare the acceleration of the two motions and see that the acceleration varies less in the non-uniform parameter setting.

Using the approximating variational subdivision algorithm for curves introduced in Sect. 4.2, we construct motions that interpolate or approximate the given input positions according to the choice of the parameters λ_i. Fig. 10 shows four different choices of the smoothing parameters λ_i to the same 6 input positions Σ_i of a robot gripper arm. If an input position Σ_i is approximated, then the corresponding position in the final motion is denoted by Σ_i^*. Note that the same values of λ_i are used for all feature curves.

Remark 5 The present approach at first constructs feature paths with a variational scheme and then corrects the resulting affine distortions of the moving body in order to get a rigid body motion. In our geometric model in A^{12}, we can say that we are computing via variational subdivision a curve in A^{12} which interpolates the given points s_i on M^6. The variational method yields a curve $\mathbf{c}'$ in A^{12} and not in M^6. Projecting that curve $\mathbf{c}'$ onto M^6 results in a curve $\mathbf{c}$, which is usually not far away



a **b**

Fig. 7 **a** Open and **b** closed uniform C^2 rigid body motion of the Stanford bunny interpolating 6 input positions (dark).

from $\mathbf{c}'$. It is not the minimizer of the chosen functional under the constraint that the curve lies on M^6. In a recent paper by Pottmann et al. 2002b an active contour model which is capable of handling the variational design of $\mathbf{c}$ on M^6 has been outlined. A closely related subdivision method which simultaneously inserts new positions so that the change of velocity at the feature points is minimized, has been presented by Hofer et al. 2002.

4.4 Numerical results and discussion

All numerical results have been obtained with a prototypic Matlab 6.5 implementation on a 1.8GHZ PC with 1GB RAM running under the Windows 2000 operating system. We have performed 7 subdivision steps of the motion design algorithm described in Sect. 4.3 to get the numerical results shown in Table 1.

Table 1 Numerical results for $m = 7$ subdivision steps of the motion design algorithm.

	Bunny	**Teapot**	**Gripper**	**Window**
N/F	6/768	5/513	6/641	4/385
A_c	CUI	ONI	OUA	OUI
T_c	0.16s	0.20s	0.38s	0.06s
T_r	0.55s	0.34s	0.43s	0.29s

We explain the obtained results at hand of the bunny example: given $N = 6$ input positions the computation of 7 subdivision steps in the closed case results in $F = 768$ final positions, including the 6 input positions, of the moving body. The curve design algorithms A_c we use are the variational subdivision schemes, abbreviated in the following way:

- CUI: Closed Uniform Interpolatory
- ONI: Open Non-uniform Interpolatory
- OUA: Open Uniform Approximating
- OUI: Open Uniform Interpolatory.

In the bunny example CUI has been applied to $\hat{\mathbf{f}}_1^k, \ldots, \hat{\mathbf{f}}_6^k, k = 1, \ldots, 4$, resulting in a total computation time T_c for the curve design algorithm of 0.16s. The total computation time T_r for the registration of Σ to the computed $6 \cdot 127 = 762$ positions is 0.55s. Note that registration has to be performed only for the computed *intermediate* positions in the interpolatory setting, but for *all* positions in the approximating case. The motions corresponding to the numerical results shown in Table 1 are illustrated for $m < 7$,

- Stanford bunny: Fig. 7b with $m = 5$,
- Utah teapot: Fig. 8 with $m = 3$,
- robot gripper arm: Fig. 10c with $m = 4$
- window: Fig. 11f with $m = 6$.

If the time instances t_i corresponding to the input positions Σ_i are chosen uniformly as $t_i = i$, then after 7, 6, 5, and 4 subdivision steps we have 127, 63, 31, and 15 intermediate positions which corresponds to a position every $1/128 \approx 0.0078s$, $1/64 \approx 0.0156s$, $1/32 \approx 0.0312s$, and $1/16 = 0.0625s$, respectively. To generate a computer animation of the moving body, 5 resp. 4 subdivi-

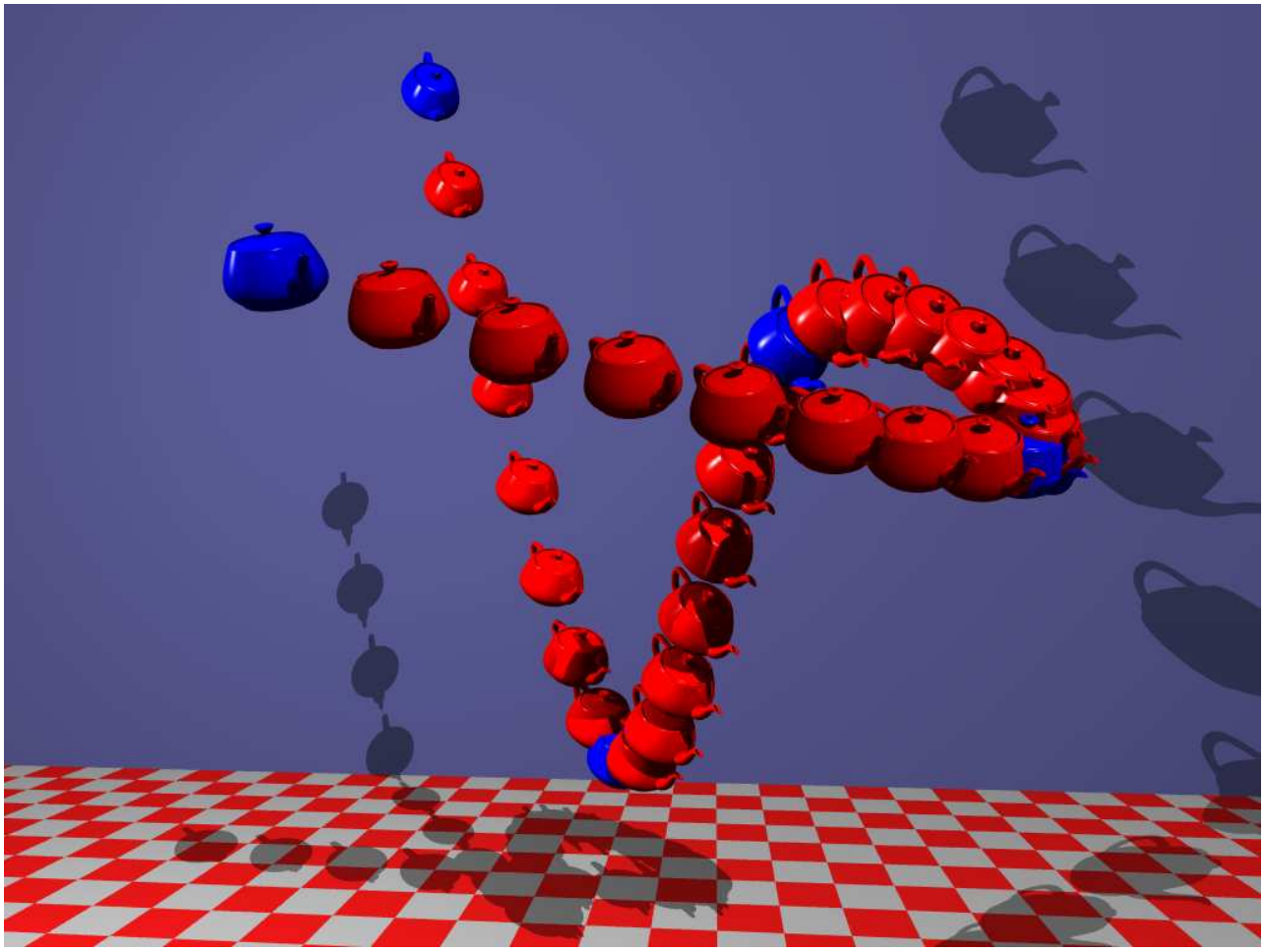
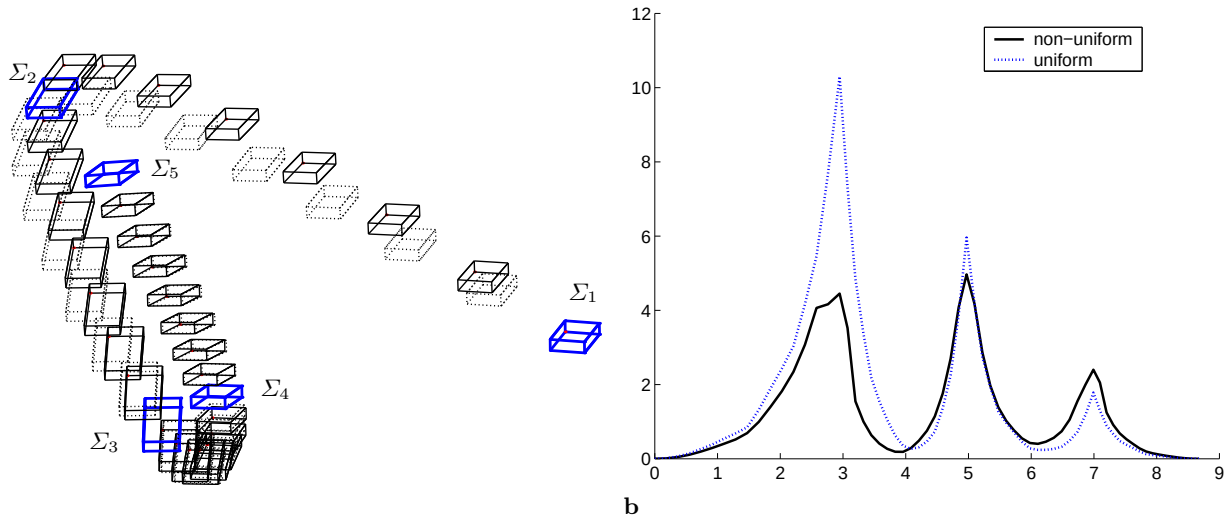


Fig. 8 Open non-uniform C^2 rigid body motion of the Utah teapot interpolating 5 input positions (blue).



a **b**

Fig. 9 **a** Open C^2 rigid body motion in the uniform (dotted) and non-uniform (solid) parameter setting applied to the same input positions (solid bold). **b** Average squared acceleration as a function of time for the uniform and non-uniform motion. The peaks correspond to the input positions Σ_2, Σ_3, and Σ_4, respectively.

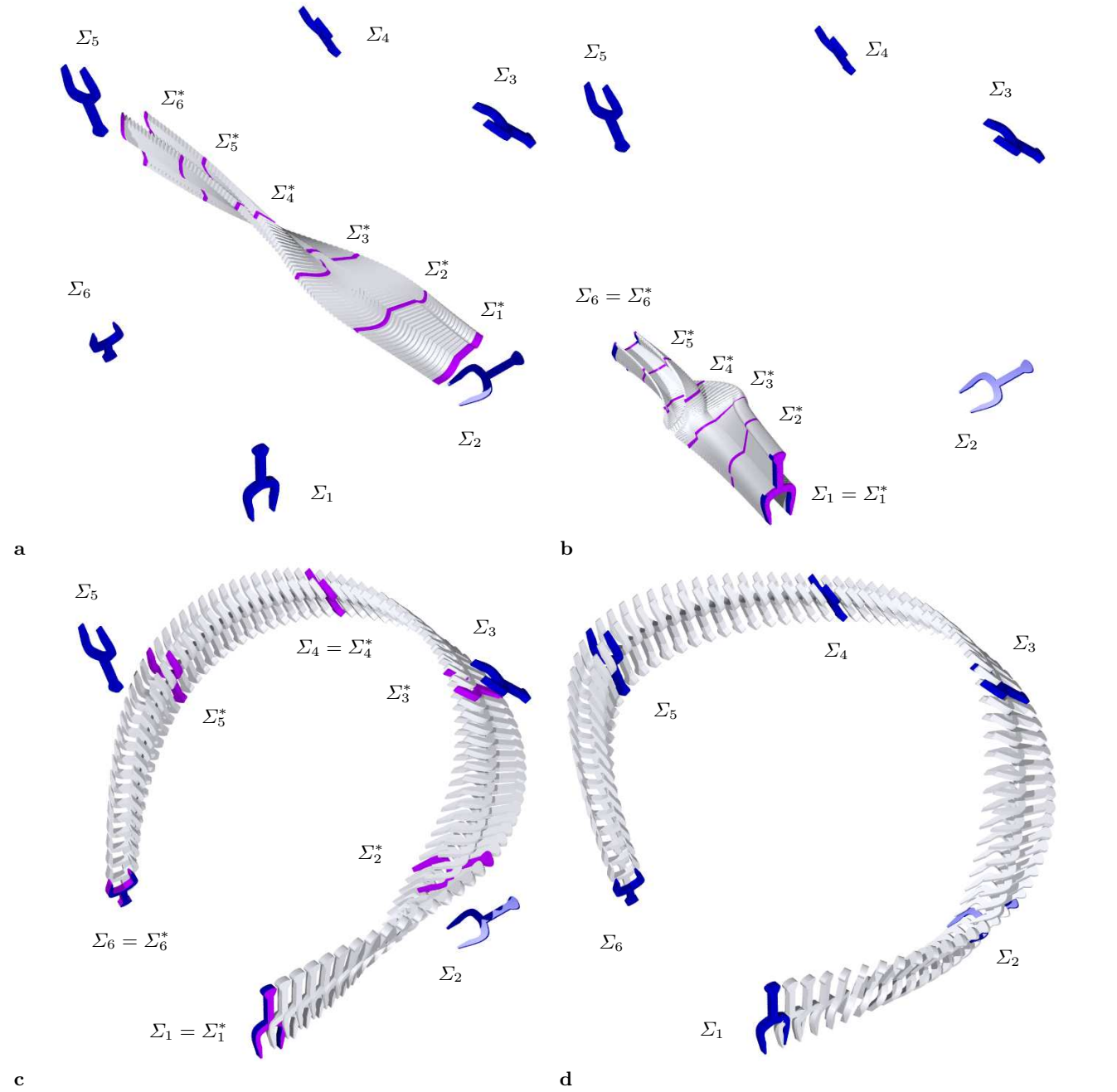


Fig. 10 Some intermediate positions (silver) of a uniform rigid body motion of a robot gripper arm interpolating or approximating given input positions (blue). Intermediate positions Σ_i^* in light blue correspond to the input positions Σ_i that are approximated by the final motion. **a** $\lambda_i = 0.001$ for all i. **b** $\lambda_1 = \lambda_{N_m} = 10^8$, $\lambda_i = 0.001$ for all other i. **c** $\lambda_1 = \lambda_{3\cdot 2^m+1} = \lambda_{N_m} = 10^8$, $\lambda_{1\cdot 2^m+1} = 0.5$, $\lambda_{2\cdot 2^m+1} = 1.5$, $\lambda_{4\cdot 2^m+1} = 0.9$, and $\lambda_i = 1$ for all other i. **d** $\lambda_i = 10^8$ for all i; $m = 1, 2, 3, 4$.

sion steps are sufficient, since these already allow us to generate 32 resp. 16 frames per second.

All employed curve design algorithms lead to the solution of a linear system of equations. Since in the non-uniform interpolatory and the uniform approximating curve case the coefficient matrix of the linear system is slightly more complicated (but it is still sparse and has a band structure), the computation takes slightly longer than in the uniform interpolatory case. Nevertheless, even with our prototypic Matlab implementation our algorithm for uniform or non-uniform, interpolating or approximating motion design is fast enough for interactive design of a rigid body motion.

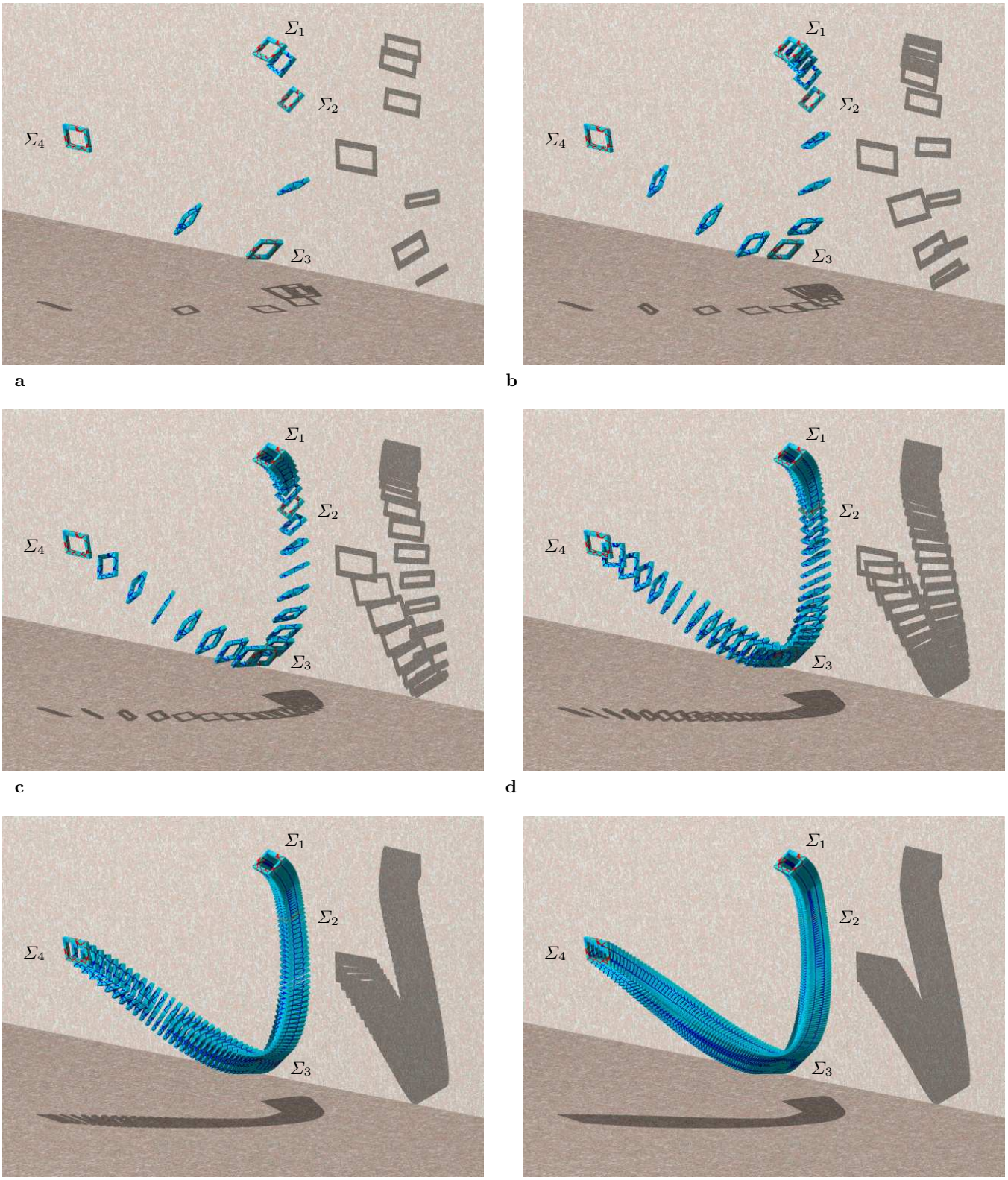


Fig. 11 Open C^2 rigid body motion of the 'window' shown in Fig. 1: **a-f** show 1 to 6 steps of uniform interpolatory variational subdivision for motion design applied to 4 input positions $\Sigma_1, \ldots, \Sigma_4$.

4.5 Local and global shape modifications of the designed motion

The present approach is well suited for modifications of the designed motion. This is so since there is a large variety of shape modification techniques for curves which have been developed in CAGD in the past decades. It would lead too far to discuss the possibilities in detail. For example, if we would like to increase *tension*, then we may use splines in tension or any other curve design scheme with tension parameters. The definition of tension for a motion which is based on a set of feature paths is appropriate for our purpose. Another example is *motion fairing* by employing a *curve fairing* algorithm.

We would like to add another remark. Here we mainly discussed a subdivision strategy based on insertion of new positions, see Fig. 11 for an illustration of the first to sixth subdivision step of uniform interpolatory variational motion design applied to the rigid body shown in Fig. 1. One can combine the presented *global* variational algorithms with *local* subdivision schemes (cf. Warren, Weimer 2001). This allows the following strategy: Perform a few iterations with the global variational subdivision algorithm and then switch to a local subdivision scheme in the following steps, which will yield a good result.

5 Conclusions and future research

We have presented a transfer principle from curve design algorithms to motion design and proved that the such generated motion is of the same smoothness as the employed curve design algorithm. In several examples we have demonstrated the effectiveness of our algorithm for the design of a rigid body motion that interpolates or approximates given input positions. The implementation of the algorithm is straightforward. Since it is a transfer principle from curve design algorithms to motion design, it can be applied with the wide variety of curve design algorithms that have been developed over the past decades to generate a wide variety of rigid body motions in a computationally efficient way.

It is an interesting topic for future research to extend the presented method to the animation of *articulated bodies* (kinematic chains of rigid bodies linked with certain joints). One idea is to add constraints associated with joints, another method is to animate multiple rigid bodies and coordinate their relative motions. Another direction of our current research is the extension of the present algorithm to the *design of smooth rigid body motions in the presence of obstacles*.

We will also investigate *splines on manifolds* in more detail. An outline of the idea has recently been given in Pottmann et al. 2002b, but much more work still needs to be done. The approach we are taking is a so-called active contour model, see Blake, Isard 1998, and relies on the squared distance function to the manifold M^6 in A^{12}. The singular set of this function has been investigated by Wallner 2002. We need more work on local quadratic approximants of the squared distance function of M^6 and on the differential geometry of M^6 in A^{12} since this governs the active contour models we are dealing with.

Acknowledgements This research has been carried out, in part, in connection with project P13938MAT and P16002-N05, supported by the Austrian Science Fund (FWF). This work was supported, in part, by the California Department of Transportation through the basic research component of the AHMCT research center at UC Davis. We would like to thank Johannes Wallner for valuable comments during our work, and the anonymous reviewers for helpful suggestions. We acknowledge the use of the Jinn Blinn version of the Utah teapot data, which has originally been designed by Martin Newell in 1974, and the Stanford Bunny data, scanned in 1993-94 at the Stanford Computer Graphics Laboratory. Last but not least we would like to thank the POV-Ray team for providing such an excellent raytracing software for free. We have used it with great joy for all rendered images in this paper.

References

[1] O. Arikan, D.A. Forsyth (2002) Interactive motion generation from examples. Proceedings SIGGRAPH '02, pp 483–490

[2] A.H. Barr, B. Currin, S. Gabriel, J.F. Hughes (1992) Smooth interpolation of orientations with angular velocity constraints using quaternions. Computer Graphics 26 (SIGGRAPH '92), pp 313–320

[3] O.B. Bayazit, J.-M. Lien, N.M. Amato (2002) Probabilistic roadmap motion planning for deformable objects. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02), pp 2126–2133

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

[5] F. Bernardini, H. Rushmeier (2002) The 3D model acquisition pipeline. Computer Graphics Forum 21(2):149–172

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

[7] A. Blake, M. Isard (1998) Active Contours. Springer

[8] Y. Chen, G. Medioni (1992) Object modeling by registration of multiple range images. Image and Vision Computing 10(3):145-155

[9] D. W. Eggert, A. Larusso, R. B. Fisher (1997) Estimating 3-D rigid body transformations: a comparising of four major algorithms. Machine Vision and Applications 9(5/06):272–290

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

[11] D. Halperin, L. Kavraki, J. C. Latombe (1997) Robotics. In: Handbook of Discrete and Computational Geometry. J. E. Goodman and J. O'Rourke, eds., CRC Press, New York, pp 775–778

[12] A.J. Hanson (1998). Constrained optimal framings of curves and surfaces using quaternion gauss maps. In Proceedings of Visualization '98, IEEE Computer Society Press, pp 375–382

[13] M. Hofer, H. Pottmann, B. Ravani (2002) Subdivision algorithms for motion design based on homologous points. In: J. Lenarcic, F. Thomas, eds., Advances in Robot Kinematics – Theory and Applications, Kluwer Academic Publishers, pp 235–244

[14] M. Hofer, H. Pottmann, B. Ravani (2003) Geometric design of motions constraint by a contacting surface pair. Computer Aided Geometric Design 20(8-9), 523–547

[15] B.K.P. Horn (1987) Closed form solution of absolute orientation using unit quaternions. Journal of the Optical Society A 4(4):629–642

[16] C.-C. Hsieh, T.-Y. Chang (2003) Motion fairing using genetic algorithms. Computer-Aided Design 35(8):739–749

[17] D. E. Hyun, B. Jüttler, M. S. Kim (2001) Minimizing the distortion of affine spline motions. Proc. of Pacific Graphics 01, IEEE Press, pp 50–59

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

[19] B. Jüttler, M. Wagner (1996) Computer Aided Design with spatial rational B-spline motions. ASME J. Mech. Design 118(2):193–201

[20] B. Jüttler, M. Wagner (2002) Kinematics and Animation. In: G. Farin, J. Hoschek, M.S. Kim, eds., Handbook of Computer Aided Geometric Design, Elsevier, pp 723–748

[21] M. Kalisiak, M. van de Panne (2001) A grasp-based motion planning algorithm for character animation. J. Visual Comput. Animat. 12(3):117–129

[22] L. Kobbelt (1996) A variational approach to subdivision. Computer Aided Geometric Design 13(8):743–761

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

[24] J.-C. Latombe (1999) Motion planning: A journey of robots, molecules, digital actors, and other artifacts. International Journal of Robotics Research 18(11):1119–1128

[25] J.-C. Latombe (2001) Robot Motion Planning. 6th printing, Kluwer

[26] É. Marchand, N. Courty (2002) Controlling a camera in a virtual environment. The Visual Computer 18(1):1–19

[27] F. C. Park, B. Ravani (1997) Smooth invariant interpolation of rotations. ACM Trans. Graphics 16(3):277–295

[28] H. Pottmann, S. Leopoldseder, M. Hofer (2002a) Simultaneous registration of multiple views of a 3D object. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXIV, Part 3A, Commission III, pp 265-270

[29] H. Pottmann, S. Leopoldseder, M. Hofer (2002b) Approximation with active B-spline curves and surfaces. Proceedings of Pacific Graphics 02, Beijing, IEEE Computer Society, pp 8–25

[30] H. Pottmann, M. Peternell (2000) On approximation in spaces of geometric objects. In: R. Cipolla and R. Martin, eds., The Mathematics of Surfaces IX, Springer, pp 438–458

[31] H. Pottmann, J. Wallner (2001) Computational Line Geometry. Springer-Verlag

[32] R. Ramamoorthi, A.H. Barr (1997) Fast construction of accurate quaternion splines. Computer Graphics 31 (SIGGRAPH '97), pp 287–292

[33] O. Röschel (1998) Rational motion design — a survey. Computer-Aided Design 30(3):169–178

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

[35] B. Salomon, M. Garber, M.C. Lin, D. Manocha (2003) Interactive navigation in complex environments using path planning, ACM SIGGRAPH Symposium on Interactive 3D Graphics, Monterey, CA

[36] M. Sharir (1997) Algorithmic motion planning. In: Handbook of Discrete and Computational Geometry. J. E. Goodman and J. O'Rourke, eds., CRC Press, New York, pp 733–754

[37] Shoemake, K. (1985) Animating rotation with quaternion curves. Computer Graphics 19 (SIGGRAPH '85), pp 245–254

[38] G. Song, N.M. Amato (2002) Using motion planning to study protein folding pathways. Journal of Computational Biology 9(2):149–168

[39] G. Wahba (1990) Spline Models for Observational Data. SIAM, Philadelphia.

[40] J. Wallner (2002) L^2 Approximation by Euclidean motions. Technical Report No. 93, Institute of Geometry, Vienna University of Technology

[41] J. Wallner (2004) Gliding spline motions and applications. Computer Aided Geometric Design 21(1), 3–21

[42] J. Warren, H. Weimer (2001) Subdivision Methods for Geometric Design: A Constructive Approach. Morgan Kaufmann Series in Computer Graphics