Vis Comput
DOI 10.1007/s00371-014-1027-1

ORIGINAL ARTICLE

On spatio-temporal feature point detection for animated meshes

Vasyl Mykhalchuk · Hyewon Seo · Frederic Cordier

Abstract Although automatic feature detection has been a long-sought subject by researchers in computer graphics and computer vision, feature extraction on deforming models remains a relatively unexplored area. In this paper, we develop a new method for automatic detection of spatio-temporal feature points on animated meshes. Our algorithm consists of three main parts. We first define local deformation characteristics, based on strain and curvature values computed for each point at each frame. Next, we construct multi-resolution space–time Gaussians and difference-of-Gaussian (DoG) pyramids on the deformation characteristics representing the input animated mesh, where each level contains 3D smoothed and subsampled representation of the previous level. Finally, we estimate locations and scales of spatio-temporal feature points by using a scale-normalized differential operator. A new, precise approximation of spatio-temporal scale-normalized Laplacian has been introduced, based on the space–time DoG. We have experimentally verified our algorithm on a number of examples and conclude that our technique allows to detect spatio and temporal feature points in a reliable manner.

Keywords Feature detection · Animated mesh · Multi-scale representation · Difference of Gaussian

Electronic supplementary material The online version of this article (doi:10.1007/s00371-014-1027-1) contains supplementary material, which is available to authorized users.

V. Mykhalchuk · H. Seo (✉)
University of Strasbourg, Strasbourg, France
e-mail: seo@unistra.fr

F. Cordier
University of Haute Alsace, Mulhouse, France

1 Introduction

With the increasing advances in animation techniques and motion capture devices, animation data have become more and more available today. Coupled with this, almost all geometry processing techniques (alignment, reconstruction, indexing, compression, segmentation, etc.) began to evolve around the new, time-varying data, which is an active research area in computer graphics. Many applications in medicine and engineering benefit from the increased availability and usability of animation data.

Since such data have considerably large sizes, it often becomes indispensable to be able to select distinctive features from it, so as to maintain efficiency in its representation and in the process applied to it. Consequently, the need for robust, repeatable, and consistent detection of meaningful features from animation data cannot be overemphasized. However, the feature detection in animated mesh remains as a much less explored domain, despite the proliferation of feature detectors developed by many researchers in computer graphics and computer vision.

In this paper, we develop a spatio-temporal feature detection framework on animated meshes (an ordered sequence of static mesh frames with a fixed number of vertices and connectivity), based on the scale space approaches. Our algorithm, which we call AniM-DoG, extends the spatial IP (interest point) detectors on static meshes [4,5,17,25] to animated meshes, so as to detect spatio-temporal feature points on them. Based on a deformation characteristic computed at each vertex in each frame, we build the scale space by computing various smoothed versions of the given animation data. At the heart of our algorithm is a new space–time difference of Gaussian (DoG) operator, which is an approximation of the spatio-temporal, scale-normalized Laplacian. By computing the local extrema of the new operator in space–

Published online: 20 September 2014

time and scale, we obtain repeatable sets of spatio-temporal feature points over different deforming surfaces modeled as triangle mesh animations. We then validate the proposed AniM-DoG algorithm for its robustness and consistency. To the best of our knowledge, our work is the first that addresses the spatio-temporal feature detector in animated meshes.

The remainder of the paper is organized as follows. In Sect. 2, we survey related works on local feature extraction in videos and (static) meshes. After recapitulating some basic terminologies and notions in Sect. 3, we present an overview of the method's pipeline overview in Sect. 4. Next, we describe the scale space representation and our AniM-DoG algorithm in Sect. 5. In Sect. 6 we show the results of the proposed feature point extraction algorithm and evaluate the robustness of the method. Finally, we present some useful applications of the spatio-temporal feature detection in Sect. 7 and conclude in Sect. 8.

2 Previous works

Feature extraction is essential in different domains of computer graphics and is frequently used for numerous tasks including registration, object query, object recognition, etc. Scale-space representation has been widely used for feature extraction in image, video and triangle mesh data sets [12]. However, almost no research has been done on the feature extraction of deforming surfaces, such as animated meshes.

Interest point detection in images and videos Perhaps, one of the most popular algorithms of feature extraction on images is Harris–Stephens detector [7], which uses second moment matrix and its eigenvalues to choose points of interest. However, Harris method is not invariant to scale. Lindeberg [12] tackled that problem and introduced automatic scale selection technique, which allows feature point detection at their characteristic scales. As Lindeberg has shown, local scale estimation using the normalized Laplace operator allows to robustly detect interest point of different extents. Mikolajczyk and Schmid [16] further developed Lindeberg's idea. As an improvement to the work of Lindeberg, the authors proposed using simultaneously Harris and Laplacian operators to detect interest points in scale-space representation of an image. First, feature point candidates are detected as local maxima of Harris function in the image plane. Further, to obtain a more compact representation, only those points are retained where Laplacian reaches maxima over scale space. This approach, however, requires dense sampling over the scale parameters and is therefore computationally expensive. As Lowe [13] proposed, difference of Gaussians (DoG) is a good approximation of Laplacian and hence could be used to reduce the computational complexity.

More recently, Laptev et al. [10] investigated how the notion of scale space could be generalized to the detection of feature points in space–time data such as image sequences or videos. Interest points are identified as simultaneous maxima of the spatio-temporal Harris corner function as well as extrema of the normalized spatio-temporal Laplace operator. To avoid computational burden, the authors proposed capturing interest points in only sparse scale pyramid and then track these points in spatio-temporal scale–time–space toward the extrema of scale-normalized Laplacian. However, in their method there is no guarantee of convergence. In the work of [3], a novel detector-descriptor scheme SURF (speeded up robust features) has been proposed. The authors extend existing Hessian-based approaches and introduce 'Fast-Hessian' detector that employs integral images for fast Hessian approximation.

Feature description and feature point (FP) extraction on static meshes There have been several approaches proposed for detecting feature points on 3D meshes. Most of them extend the detectors proposed for images. Pauly et al. [17] have used 'surface variation' to measure the saliency of vertices on the mesh, from which they build multi-scale representation. After extracting points with high feature response values, they construct minimum spanning tree of the edge points to extract feature lines.

Lee et al. [14] proposed an algorithm to compute the saliency of mesh points based on the center–surround operator of Gaussian-weighted mean curvatures. First, the mean surface curvatures are computed. Then for each vertex, they estimate saliency as an absolute value of the difference between mean curvatures filtered with Gaussians of smaller and larger variances. They repeat the procedure at different scales by increasing Gaussian variance. Non-linearly normalized aggregate of saliency at all scales is defined as the final vertex saliency.

Castellani et al. [4] build scale space over vertices in a mesh with successive decimations of the original shape. The displacements of a vertex throughout the decimation are used as a measure of saliency. Then vertices with high response in its DoG operator (inter-octave local maxima) and with high saliency in the neighborhood (intra-octave local maxima) are selected as feature points.

Zaharescu et al. [25] use photometric properties associated with each vertex as a scalar function defined on a 3D mesh. A discrete operator named 'MeshDoG' is applied on this function, on which they apply Hessian operator to detect corner-like feature points. They extend MeshDOG to what they call MeshHOG, a feature descriptor, which essentially is a histogram of gradients in the neighborhood. The extracted features along with their descriptors were used for matching 3D model sequences they obtained from multi-view images.

Sipiran and Bustos [19] have used 3D Harris operator which is essentially an extension of the Harris corner detector for images. After fitting a quadratic patch to the

neighborhood, a vertex is treated as an image, on which the Harris corner detector can be been applied.

Darom and Keller [5] propose a scale-invariant local feature descriptor for the repeatable feature point extraction on a 3D mesh. Each point is characterized by its coordinates, and a scale space is built by successive smoothing of each vertex with its one-ring neighbors. Local maxima both in scale and location are chosen as features.

All these methods, however, have been concerned with mesh data defined on the spatial domain only. In this work, we propose a new feature detection technique in animated meshes which extends existing methods based on linear scale-space theory to spatio-temporal domain.

3 Preliminaries

At the heart of our algorithm is a scale-space representation. In this section we briefly recapitulate some basic notions that have been previous studied. Later, we develop its extensions to animated mesh data, which are described in Sects. 5.2 and 5.3.

Scale-space representations have been studied extensively in feature detection for images and, more recently, for videos. The basic idea is to represent an original image $f : R^{\mathrm{d}} \to R$ at different scales as L: $R^{\mathrm{d}} \times R_+ \to R$ by convolution of f with a Gaussian kernel with variance σ:

$$L(\mathbf{x};\sigma) = G(\mathbf{x};\sigma) * f(\mathbf{x}), \tag{1}$$

where

$$G(\mathbf{x};\sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^{\mathrm{d}}} \exp\left(-\frac{x_1^2 + \ldots x_{\mathrm{d}}^2}{2\sigma^2}\right). \tag{2}$$

One of the most successful feature detectors is based on DoG (difference of Gaussians). To efficiently detect feature points in scale space, Lowe [13] proposed using convolution of the input image with the DoG functions. It is computed from the difference of two nearby scales:

$$\begin{aligned} D(\mathbf{x};\sigma) &= (G(\mathbf{x};k\sigma) - G(\mathbf{x};\sigma)) * f(\mathbf{x}) \\ &= L(\mathbf{x};k\sigma) - L(\mathbf{x};\sigma), \end{aligned} \tag{3}$$

where k is a constant multiplicative factor separating the two nearby scales. Note that DoG is particularly efficient to compute, as the smoothed images L need to be computed in any case for the scale space feature description, and D can therefore be computed simply by image subtraction.

The DoG provides a close approximation to the scale-normalized Laplacian of Gaussian [11], $\sigma^2\nabla^2 G$, which has been proven to produce the most stable, scale-invariant image features [16]. The DoG and scale-normalized LoG are related through the heat-diffusion equation:

$$\frac{\partial G(\mathbf{x})}{\partial \sigma} = \sigma \nabla^2 G(\mathbf{x}), \tag{4}$$

where the Laplacian on the right side is taken only with respect to the $\mathbf{x}$ variables. From this, we see that $\nabla^2 G(\mathbf{x})$ can be computed from the finite difference approximation to $\partial G(\mathbf{x})/\partial\sigma$, using the difference of nearby scales at $k\sigma$ and σ:

$$\frac{\partial G(\mathbf{x})}{\partial \sigma} = \lim_{k\to 1} \frac{G(\mathbf{x};k\sigma) - G(\mathbf{x};\sigma)}{k\sigma - \sigma} = \sigma.\nabla^2 G(\mathbf{x}), \tag{5}$$

and therefore

$$G(\mathbf{x};k\sigma) - G(\mathbf{x};\sigma) \approx (k-1)\cdot\sigma^2\cdot\nabla^2 G. \tag{6}$$

4 Overview

Our goal is to develop a feature detector on animated mesh based on space–time DoG, which has been reported to be an efficient approximation of robust Laplacian blob detector in the space domain. Note that animated meshes that we are dealing with are assumed to have no clutters or holes, and maintain fixed topology over time, without tearing or changing genus. The spatial samplings can vary from one mesh to another, but it is desirable to have uniform sampling across one surface. The temporal sampling rate can also vary (∼30 Hz in our experiments), depending on how the animation has been obtained. In any case, the temporal sampling is considered uniform.

The features we want to extract are the corners/blob-like structures, which are located in regions that exhibit a high variation of deformation spatially and temporally. We first define local deformation attributes on the animated mesh, from which we build a multi-scale representation of it. One of the main motivations to base our method on local surface deformation rather than vertex trajectories can be explained by the fact that (1) local deformation on a surface can be effectively measured by some well-defined principles, and that (2) the domain has intrinsic dimension of 2D+time (rather than 3D+time) with some reasonable assumption on the data, i.e., differentiable two-manifold with time-varying embedding.

We then compute the deformation characteristics at different scales, by defining an appropriate spatio-temporal Gaussian-like smoothing method. However, real Gaussian smoothing on mesh animation is problematic and expensive. Therefore, we follow the other alternative and approximate Gaussian low-pass filter by a sequence of spatio-temporal box average filters of fixed width. We obtain different space and time scales of deformation field over animation by varying the number box filtering applied in space and in time.

To estimate positions and scales of mesh animation feature points, we define a scale-normalized differential operator that assumes simultaneous extrema over space–time and scale neighborhood. Theoretically, it is possible to compute

spatio-temporal, scale-normalized Laplacian on every vertex of the animated mesh. For example, one could extend the work by Zaharescu et al. and compute the 3D gradient and Laplacian on the animated mesh. However, it would be too costly as it requires computing the normal plane, on which the principal direction should be determined. Therefore, we introduce a new precise approximation of spatio-temporal scale-normalized Laplacian based on space–time difference of Gaussian. Then local extrema of the space–time DoG operator are captured as feature points. Space–time DoG operator is cheap to compute and allows to robustly detect feature points over mesh animation in a repetitive and consistent manner.

5 Dynamic feature detector (AniM-DoG)

5.1 Deformation characteristics definition

We are interested in quantities that are related to local deformation characteristics associated with each point of the mesh, at each frame. Thus, we base our algorithm on locally computed strain and curvature values computed as follows.

Strain computation We first consider the degree of deformation associated with each triangle on the mesh at each frame. Our method requires specifying the reference pose, the rest shape of the mesh before deformation (Fig. 1). In certain cases the reference pose can be found in one of the frames of the animation. If none of the given frames is appropriate as the rest pose, some prior works [9] could be adopted to compute a canonical mesh frame by taking the average of all frames.

Let $\mathbf{v}_i$ and $\tilde{\mathbf{v}}_i$ be the vertices of a triangle before and after the deformation, respectively. A 3 by 3 affine matrix $\mathbf{F}$ and displacement vector $\mathbf{d}$ transform $\mathbf{v}_\mathrm{i}$ into $\tilde{\mathbf{v}}_\mathrm{i}$ as follows:

$$\mathbf{F} \cdot \mathbf{v}_i + \mathbf{d} = \tilde{\mathbf{v}}_i, \quad \mathrm{i} = 1, \ldots 3.$$

Similarly to Sumner et al. [23], we add a fourth vertex in the direction of the normal vector of the triangle and subtract the first equation from the others to eliminate $\mathbf{d}$. Then, we get $\mathbf{F} = \tilde{\mathbf{V}} \cdot \mathbf{V}^{-1}$ where

Fig. 1 Rest shapes are chosen as the reference frame for defining the deformation characteristics

$$\mathbf{V} = [\mathbf{v}_2 - \mathbf{v}_1 \ \mathbf{v}_3 - \mathbf{v}_1 \ \mathbf{v}_4 - \mathbf{v}_1],$$

and

$$\tilde{\mathbf{V}} = [\tilde{\mathbf{v}}_2 - \tilde{\mathbf{v}}_1 \ \tilde{\mathbf{v}}_3 - \tilde{\mathbf{v}}_1 \ \tilde{\mathbf{v}}_4 - \tilde{\mathbf{v}}_1].$$

Non-translational component of $\mathbf{F}$ encodes the change in orientation, scale, and skew induced by the deformation. Note that this representation specifies the deformation on per-triangle basis, so that it will be independent of the specific position and orientation of the mesh in world coordinates. Without loss of generality, we assume that the triangle is stretched first and then rotated. Then we have $\mathbf{F} = \mathbf{RU}$, where $\mathbf{R}$ denotes the rotation tensor and $\mathbf{U}$ the stretch tensor. Since we want to describe the triangle only with its degree of stretch, we eliminate the rotation component of $\mathbf{F}$ by computing the right Cauchy deformation tensor $\mathbf{C}$ as defined by:

$$\mathbf{C} = \mathbf{F}^\mathrm{T}\mathbf{F} = (\mathbf{RU})^\mathbf{T}(\mathbf{RU}) = \mathbf{U}^\mathrm{T}\mathbf{U}.$$

It can be shown that $\mathbf{C}$ is equal to the square of the right stretch tensor. We obtain principal stretches by the Eigen-analysis on $\mathbf{C}$, and use the largest eigenvalue λ_1 (maximum principal strain) as the in-plane deformation of the triangle.

Curvature computation Computing the curvature at the vertices of a mesh is known to be non-trivial because of the piecewise linear nature of meshes. One simple way of computing the curvature would be to compute the angle between two neighboring triangles along an edge. However, such curvature measurement is too sensitive to the noise on the surface of the mesh, because its computation relies on two triangles only. Instead, we compute the curvature over a set of edges as described in [2]. Given a vertex v_i, we first compute the set of edges E_i whose two vertices are within a user-defined geodesic distance to v_i. Next, we compute the curvature at each of the edges of E_i. The curvature at v_i is then calculated as the average of the curvatures at the edges of E_i.

Deformation measure Let $\mathcal{M}$ with M frames and N vertices be a given deforming mesh. For each vertex $v_i^f \in \mathcal{M}$ ($f = 1, \ldots, M, i = 1, \ldots, N$) on which we have computed strain $s(v_i^f)$ and curvature $c(v_i^f)$, we define the deformation characteristics $d(v_i^f)$ as follows:

$$d(v_i^f) = s(v_i^f) + \ \alpha \cdot |c(v_i^f) - c(v_i^1)|.$$

The first term is obtained by transferring the above-described per-triangle strain values to per-vertex ones, computed at each frame. At each vertex, we take the average strain values of its adjacent triangles as its strain.

The second term encodes the curvature change with respect to the initial, reference frame. Note that $d(v_i^f) \geq 0$ for $\forall v_i^f$, which we use later for the feature detection (Sect. 5.3). We set α typically to 7 in our experiments. Color-coded defor-

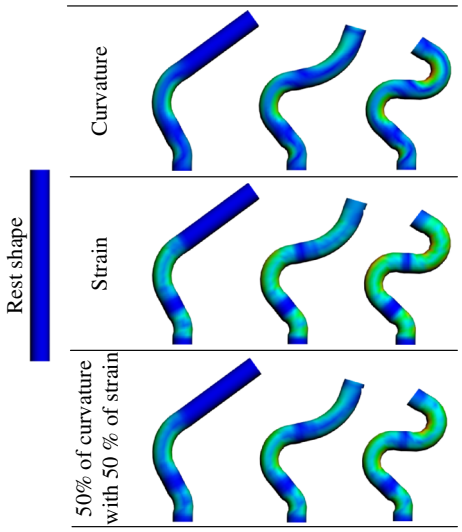

Fig. 2 Local deformation characteristics are shown on a bending cylinder mesh

mation characteristics on a bending cylinder data are shown in Fig. 2.

5.2 Scale space construction (Multi-scale representation)

Given the deformation measures d for all vertices of the input animated mesh $\mathcal{M}$, we re-compute d at K·L different scale representations, obtaining octaves $O_{kl}(k \in \Sigma = 0, \ldots \mathrm{K},\ l \in \mathrm{T} = 0, \ldots \mathrm{L})$ of deformation characteristics at different spatio-temporal resolutions. Theoretically, the octaves are obtained by applying an approximated Gaussian filter for meshes. In practice, the approximation consists of subsequent convolutions of the given mesh with a box (average) filter [5]. In our work, we define a spatio-temporal average filter on the deformation characteristics of the animated mesh and compute a set of filtered deformation scalar fields, which we call as anim-octaves. As shown in Fig. 3, we define spatio-temporal neighborhood $\mathcal{N}_{st}$ of a vertex in animation as a union of its spatial and temporal neighborhoods. A spatio-temporal average smoothing over $\mathcal{N}_{st}$ is obtained by applying a local spatial filter followed by a local temporal one.

More specifically, for each vertex v_i^f at an anim-octave of scale (σ_k, τ_l), we compute deformation measures at the next spatial octave (σ_{k+1}, τ_l), by averaging deformation measurements in the current vertex of the current octave $d(v_i^f, \sigma_k, \tau_l)$ and its one-ring's spatial neighborhood $d(\mathcal{N}_s^1(v_i^f), \sigma_k, \tau_l)$, i.e., at adjacent vertices. For the next temporal octave (σ_k, τ_{l+1}), we repeat a similar procedure, but this time averaging deformation values in one-ring temporal neighborhood $\mathcal{N}_t^1(v_i^f)$ as in Fig. 3. For the next spatio-temporal octave, we start from deformations in octave (σ_{k+1}, τ_l) and apply temporal average filter again in the way described above, which yields $d(v_i^f, \sigma_{k+1}, \tau_{l+1})$. We continue this procedure until

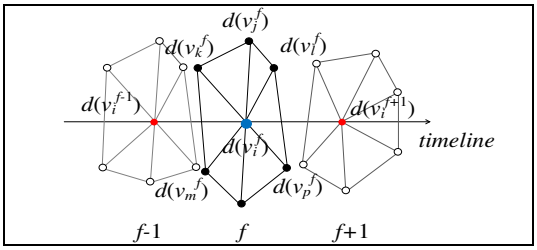

Fig. 3 The smallest possible spatio-temporal neighborhood $\mathcal{N}_{st}$ of a vertex v_i^f (*blue dot*) is composed of one-ring spatial neighbors in frame f (*black vertices*) and one-ring temporal neighbors (*red vertices*). Note that considering the temporal neighbors implies considering their spatial neighbors (*white vertices*) as well

octave scale	σ_1	σ_2	…	σ_k
τ_1	O_{11}	O_{12}	…	O_{1k}
τ_2	O_{21}	O_{22}	…	O_{2k}
⋮	⋮	⋮	⋱	⋮
τ_l	O_{l1}	O_{l2}	…	O_{lk}

Fig. 4 Scale space is built by computing a set of octaves of an input animated mesh

we build the desired number of spatio-temporal octaves. Figure 4 illustrates our anim-octaves structure. We denote an anim-octave as $O_{kl} = d(\mathcal{M}, \sigma_k, \tau_l)$, where $O_{00} = d(\mathcal{M})$. We note that although the term octave is widely used to refer to a discrete interval in the scale space, it may be misleading since in a strict sense, our octaves do not represent the interval of half or double the frequency. In Fig. 5, we illustrate multi-scale deformation characteristics that we computed on an animated mesh. The horizontal axis represents the spatial scale σ_k, and the vertical axis the temporal scale τ_l.

Widths of the average filters We set the width of the spatial filter as the average edge length of the mesh taken at the initial frame, assuming that spatial sampling of the mesh is moderately regular, and that the edge lengths in the initial frame represent well those in other frames of animation. Note that it can be done in a per-vertex manner, by computing for each vertex the average distance to its one-ring neighbors, as it has been proposed by Darom and Keller [5]. However, since this will drastically increase the computation time for the octave construction stage, we have chosen to use the same filter width for all vertices.

Determining the width of the temporal filter is simpler than the spatial one, as almost all data have regular temporal sampling rate (fps) throughout the duration of animation. Similarly to the spatial case, the inter-frame time is used to set the width of the temporal filter. Instead of averaging over immediate neighbors, however, we consider larger number of

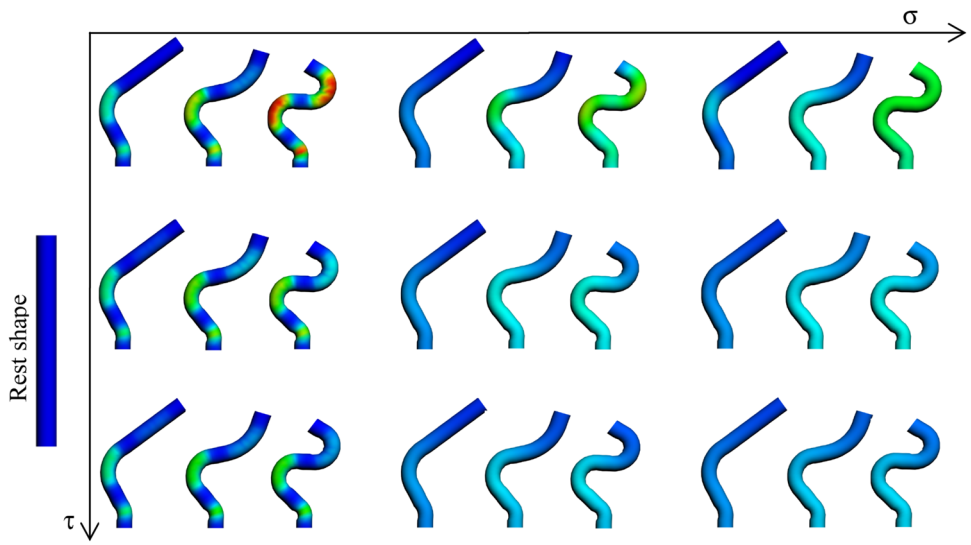

Fig. 5 Multi-scale deformation characteristics on an animated mesh. From left to right, spatial scale σ_j increases, and from top to bottom, temporal scale τ_i increases

Table 1 The animated meshes used in our experiments

Name	No.vertices/ triangles	No. frames	Filter widths (space/time)	Max. no. smoothings (space/time)
Cylinder	587/1170	40	10.0/0.83	50/100
Face1 (happy)	608/1,174	139	8.96/8.45	118/113
Face1 (surprise)	608/1,174	169	9.39/13.2	96/107
Face2 (happy)	662/1,272	159	9.31/13.2	112/94
Face2 (surprise)	662/1,272	99	8.95/8.45	109/57
Horse	5,000/9,984	48	3.48/5.33	77/54
Camel	4,999/10,000	48	2.62/5.33	102/54
Woman 1	4,250/8,476	100	5.12/5.2	72/150
Woman 2	4,250/8,476	100	4.44/5.2	82/150
Woman 3	4,250/8,476	100	4.54/5.2	99/150
Face 3	5,192/9,999	71	5.18/3.65	90/114
Head	7,966/15,809	71	9.06/3.65	28/114

frame neighbors, in most cases. This is especially true when the animated mesh is densely sampled in time. The filter widths we used for each data set are summarized in Table 1.

Maximum number of smoothings Since an animated mesh can be highly redundant and heavy in size, the memory space occupied by the anim-octaves can be large as the number of scales increases. This becomes problematic in practice. With an insufficient number of smoothings, on the other hand, features of large characteristic scale will not be detected. Indeed, when the variance of the Gaussian filter is not sufficiently large, only boundary features will be extracted. Figure 6 illustrates the principle behind the characteristic scale and the maximum required scale level. Given a spatio-temporal location on the mesh, we can evaluate the DoG response function and plot the resulting value as a function of the scale (number of smoothings). Here, the spatial scale has been chosen as a parameter for the simplicity. The characteristic scales of the chosen vertices are shown as vertical lines, which can be determined by searching for scale-space extrema of the response function. To detect the feature points on the bending region in the middle (v_1), for instance, the octaves should be built up to 12 level. This means the maximum number of smoothings must be carefully set to be able to extract feature points of all scales while maintaining a moderate number of maximum smoothing.

To make sure that the features representing blobs of large scale are detected, we start by an average filter. Multiple applications of a box (average) filter approximates a Gaussian filter [1]. More precisely, n averagings with a box filter of width w produce overall filtering effect equivalent to the Gaussian filter with a standard deviation of:

$$\sigma = \sqrt{\frac{n(w^2-1)}{12}}. \tag{7}$$

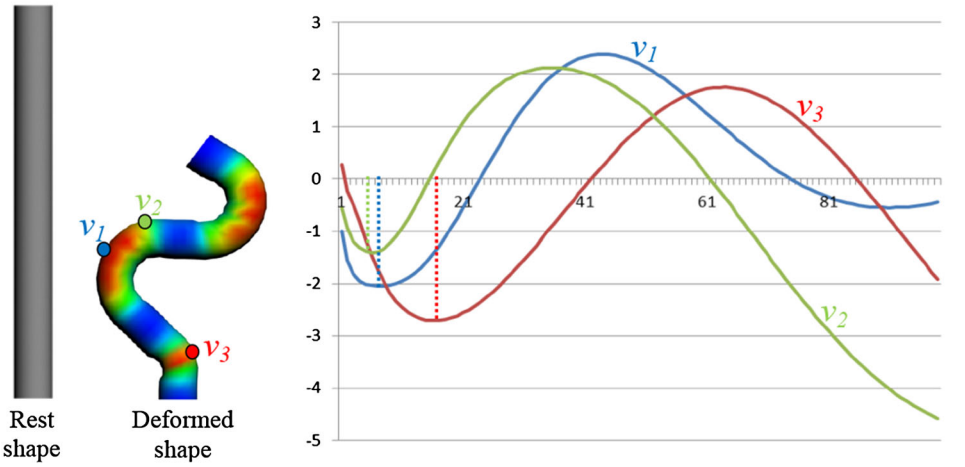

Fig. 6 The DoG response function has been evaluated as a function of the spatial scale (number of smoothings). The characteristic scales of the chosen vertices are shown as *vertical lines*

When the Laplacian of Gaussian is used for detecting blob centers (rather than boundaries), the Laplacian achieves a maximum response with

$$\sigma = \frac{r}{\sqrt{2}}, \tag{8}$$

where r is the radius of the blob we want to detect.

Now, assuming that the maximum radius $r_{\max}$ of the blob we want to detect is known, we can compute the required number of average smoothing that is sufficient to detect blob centers from Eqs. (7) and (8):

$$r_{\max}^2 = \frac{n(w^2-1)}{6} \tag{9}$$

$$\Leftrightarrow n = \frac{6r^2}{w^2-1}. \tag{10}$$

The maximum number of application of box filter for each data set is listed in Table 1.

Maximum radius of all possible blobs Along the spatial scale space, we consider the average edge length of the initial shape as the width of the average filter w, as described above. For the maximum possible radius of a blob, we compare the axis length change of the tight bounding box of the mesh during animation, with respect to its initial shape. The half of the largest change in axis length is taken as $r_{\max}$.

Along the temporal scale space, we assume that the maximum radius $r_{\max}$ of all possible blobs is the half of the total time of duration of animation. By fixing the maximum number of smoothing to some moderate value, we obtain the desirable box filter width from Eqs. (9) or (10).

5.3 FP detection by DoG

In this section, we extend the idea of scale representation in spatial domain to spatio-temporal domain and adopt it to the case of animated mesh. Next, we propose our feature point detector and discuss some of its implementation related issues.

Spatio-temporal scale space principles Given time-varying input signal $f(\mathbf{x}, t)$, $f : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}$, one could build its scale-space representation $L(\mathbf{x}, t; \sigma, \tau)$ by convoluting f with anisotropic Gaussian

$$L(\mathbf{x}, t; \sigma, \tau) : \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}_+^2 \rightarrow \mathbb{R}.$$

The motivation behind the introduction of separate scale parameters in space σ and time τ is that the space and the time extents of feature points are independent in general [10].

Alternatively, another useful formulation of spatio-temporal scale space was reported in the work of Salden et al. [22]. The spatio-temporal scale space $L(\mathbf{x}, t; \sigma, \tau)$ for a signal $f(\mathbf{x}, t)$ could be defined as a solution of two diffusion equations:

$$\frac{\partial L}{\partial \sigma} = \sum\nolimits_i \frac{\partial^2 L}{\partial x^i \partial x^i}, \tag{11a}$$

$$\frac{\partial L}{\partial \tau} = \frac{\partial^2 L}{\partial^2 t}, \tag{11b}$$

with an initial condition

$$\lim_{\sigma \rightarrow 0^+ \tau \rightarrow 0^+} L(x, t; \sigma, \tau) = f(x, t).$$

In our case, the input animated mesh $\mathcal{M}$ can be considered as two-manifold with time-varying embedding, i.e., $m(u, v, t)$ in 3D Euclidean space. Measuring deformation scalar field $d(\mathcal{M})$ in two-manifold over space and time yields a 3D input signal of the form $d(u, v, t)$, $d : \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}$, and its scale space of the form $L_d(u, v, t; \sigma, \tau) : \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R}_+^2 \rightarrow \mathbb{R}$. Given the scale space representation $L_d(\mathbf{x}, t; \sigma, \tau)$ of the input animated mesh, we proceed with the construction of the DoG feature response pyramid, which we describe below.

Computing DoG pyramid To achieve the invariance in both space and time, we introduce a spatio-temporal DoG operator, which is a new contribution. Our idea is to combine the

spatial and the temporal parts of Laplacian and difference-of-Gaussians. Given the property of DoG (Eq. 6) and Eqs. 11a, 11b, we obtain the following:

$$D(\mathbf{x};\sigma) = L(\mathbf{x};k\sigma) - L(\mathbf{x};\sigma) = \sum\nolimits_i \frac{\partial^2 L}{\partial x^i \partial x^i}, \tag{12a}$$

$$D(t;\tau) = L(t;k\tau\ ) - L(t;\tau\ ) = \frac{\partial^2 L}{\partial^2 t}. \tag{12b}$$

Then we propose to define the spatio-temporal Laplacian by adding (12a) and (12b):

$$D(\mathbf{x};\sigma) + D(\mathbf{t};\tau) = \sum\nolimits_i \frac{\partial^2 L}{\partial x^i \partial x^i} + \frac{\partial^2 L}{\partial^2 t} = \nabla^2 L. \tag{13}$$

The new spatio-temporal Laplacian operator is just a sum of DoG in space scale and DoG in time scale, which is computationally very efficient.

To be able to extract features of all scales correctly, we need to scale-normalize the DoG response function. Choosing the exponent coefficients for the spatial Eq. 12a (rightmost term) and the temporal Eq. 12b (rightmost term) parts of Laplacian [10], we have:

$$\nabla^2_{norm} L = \sigma^2 \tau^{1/2} \sum_i \frac{\partial^2 L}{\partial x^i \partial x^i} + \sigma \tau^{3/2} \frac{\partial^2 L}{\partial^2 t}. \tag{14}$$

Therefore, to achieve scale-normalized approximation of Laplacian through DoG, we multiply both sides of (12a) with $\sigma^2\tau^{1/2}$ and both sides of (12b) with $\sigma\tau^{3/2}$ obtaining

$$\sigma^2 \tau^{1/2} D(\mathbf{x};\sigma) = \sigma^2 \tau^{1/2} \sum\nolimits_i \frac{\partial^2 L}{\partial x^i \partial x^i}, \tag{15a}$$

$$\sigma \tau^{3/2} D(t;\tau) = \sigma \tau^{3/2} \frac{\partial^2 L}{\partial^2 t}. \tag{15b}$$

From (15a–15b) we see that

$$\nabla^2_{\text{norm}} L = \sigma^2 \tau^{1/2} \mathrm{D}(\mathbf{x};\sigma) + \sigma \tau^{3/2} D(t;\tau).$$

On the other hand, we can get a formulation of spatio-temporal DoG that approximates scale-normalized Laplacian

$$D_{st}(\mathbf{x},t;\sigma,\tau) = \sigma^2 \tau^{1/2} D(\mathbf{x};\sigma) + \sigma \tau^{3/2} D(t;\tau).$$

Thus, given the definition of spatio-temporal difference of Gaussians, we can compute feature response pyramid in the following way. For each vertex (u, v, t) in the animated mesh $\mathcal{M}$, and for every scale $(\sigma_k, \tau_l) \in \Sigma \times \mathrm{T}$ of the surface deformation pyramid, we compute $D_{\text{st}}(u, v, t; \sigma_k, \tau_l)$.

FP detection Once the spatio-temporal DoG pyramid $\{D_{\text{st}}(u, v, t; \sigma_k, \tau_l) | (\sigma_k, \tau_l) \in \Sigma \times \mathrm{T}\}$ is constructed, we extract feature points by identifying local extrema of the adjacent regions in space, time, and scales. In contrast to Mikolajczyk and Schmid [16] who computes Harris and Laplacian operators, our method requires only DoG, which makes itself computationally efficient. This is particularly interesting for the animated mesh data which are generally much heavier than the image. Considering that our surface deformation function is always non-negative (and consequently its scale-space representation), it is worth mentioning that Laplacian of Gaussian and its DoG approximation reach local minima at the centers of blobs. Such specific LoG behavior is illustrated in Fig. 7.

For each scale $(\sigma_k, \tau_l) \in \Sigma \times \mathrm{T}$ of 2D DoG pyramid, we first detect vertices in animation that are local minima in DoG response over their spatio-temporal neighborhood $\mathcal{N}_{st}$:

$$P_{kl} = \{p \in \mathcal{M} \,|\, \forall p_i \in \mathcal{N}_{st}(p),\ D_{kl}(p) < D_{kl}(p_k) \text{ and } D_{kl}(p) < \varepsilon_{st}\},$$

where $\mathcal{N}_{st}(p)$ is a spatio-temporal neighborhood of vertex p in the animation $\mathcal{M}$ (Fig. 3).

Next, out of preselected feature candidates P_{kl}, we retain only those vertices which are simultaneous minima over neighboring spatio-temporal scales of DoG pyramid:

$$P = \{p \in P_{kl} | \forall (i,j) \in \mathcal{N}_{\sigma\tau}(k,\mathrm{l}),\ D_{ij}(p) > D_{kl}(p) \text{ and } D_{kl}(p) < \varepsilon_{\sigma\tau}\},$$

where $\mathcal{N}_{\sigma\tau}(k,\mathrm{l})$ is a set of eight neighboring scales $\mathcal{N}_{\sigma\tau}(k,\mathrm{l}) = \{D_{(k+1)l}, D_{(k+1)(l-1)}, D_{(k+1)(l+1)}, D_{k(l-1)}, D_{k(l+1)}, D_{(k-1)l}, D_{(k-1)(l-1)}, D_{(k-1)(l+1)}\}$, and ε_{st}, $\varepsilon_{\sigma\tau}$ are user-controlled thresholds. The spatial scale of a feature point corresponds to the size of the neighborhood where some distinctive deformation is exhibited. Similarly, the temporal scale corresponds to the duration (or speed) of the deformation.

Dealing with secondary (border) blobs However, in case we consider local maxima of DoG (LoG) magnitude, we may detect artifacts. Undesirable secondary blobs are caused by the shape of Laplacian of Gaussian which yields peaks around the border of the real blob (Fig. 7a). Consider a perfect Gaussian blob as an input signal. If we assume the magnitude (i.e., absolute value) of LoG to be feature response, we get a strong peak at the center of the blob and two other secondary peaks around the edges, and that is troublesome. In contrast, dealing with signed LoG (not absolute) we observe valleys (local minima) at blob centers and peaks (local maxima) on borders. Hence, searching for local minima of LoG, rather than local maxima of LoG magnitude, prevents the detection of false secondary features (Fig. 7b–f). The other way around could be to use LoG magnitude, but discard local maxima which are not strong enough in the initial signal and therefore are false findings. Note that previous works on feature extraction on images/video/static meshes [10,16,25] often adopt Hessian detector, which does not detect secondary blobs. However, in contrast to the DoG detector, estimation of Hessian on a mesh surface is significantly heavier. It is

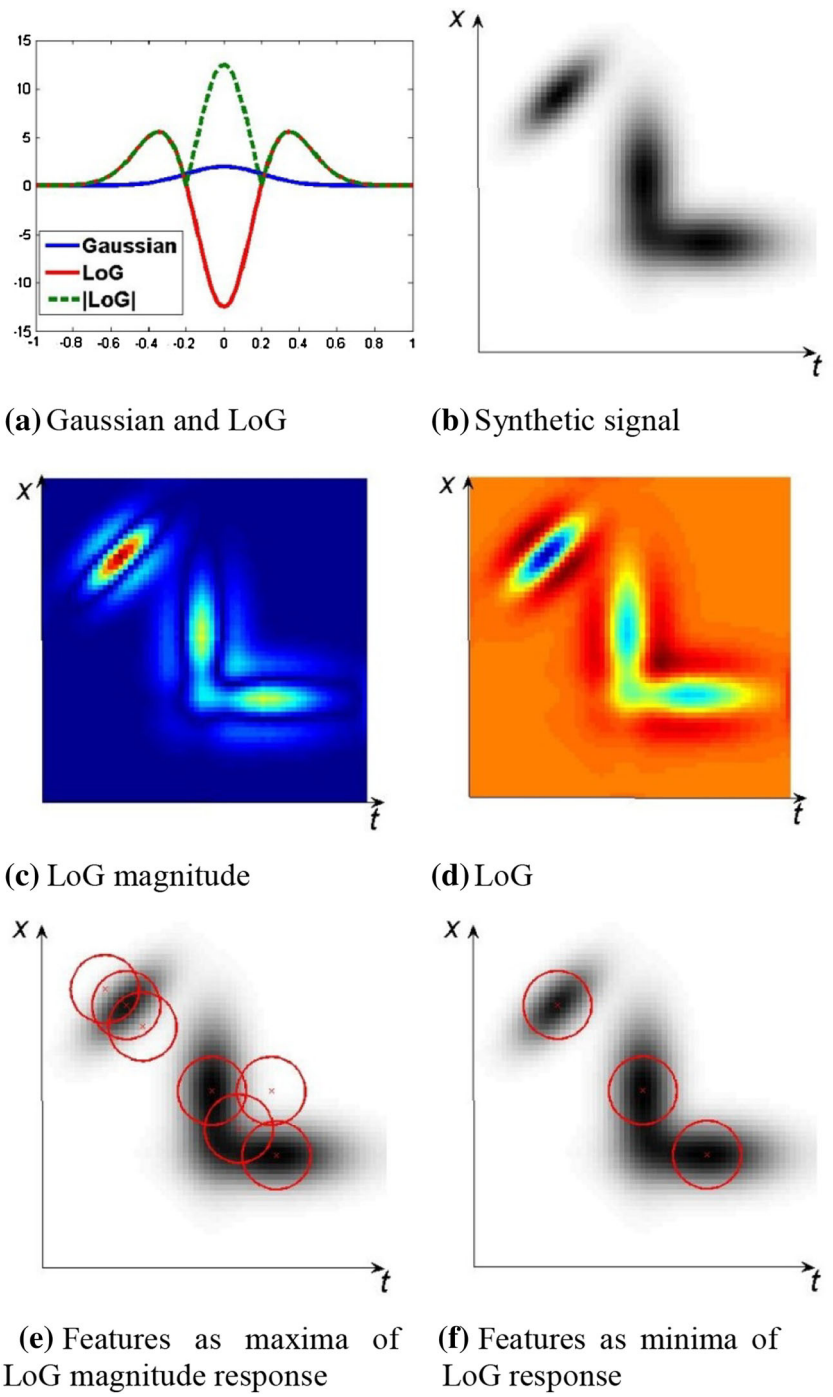

Fig. 7 A 2D illustration of our feature detection method. **a** LoG yields the valley at the blob's center and peaks around the boundary, while the magnitude of LoG has peaks in both cases. **b** Synthetic input signal consisting of three Gaussian blobs in 2d. **c** Response of synthetic 2d signal as the absolute value of LoG. **d** Response of the 2d signal computed as LoG. **e** Working with LoG magnitude response we observe several false secondary blobs. **f** Features captured as the local minima of LoG response are reliable

even more problematic and challenging to estimate Hessian matrix on animated mesh.

Implementation notes Often, animated meshes are rather heavy data. As we increase the number of anim-octaves in the pyramid, we can easily run out of memory, since each octave is essentially a full animation in itself, but at a different scale. Consequently, we have to address that issue in the implementation stage. To minimize memory footprint, we compute pyramids and detect feature points progressively. We fully load into main memory only space scale dimension of Gaussian and DoG pyramids. As for time scale, we keep only two neighboring time octaves simultaneously, which are required for DoG computation. Then we construct the pyramid from bottom to top by iteratively increasing the time scale. On each iteration of Gausian/DoG pyramid movement along the time scale, we apply our feature detection method to capture interest points (if any on current layer).

We repeat the procedure until all pyramid scales have been processed.

6 Experiments

Deforming meshes used in our experiments include both synthetic animations and motion capture sequences, which are summarized in Table 1. We synthesized a simple deforming cylinder animation by rigging and used it for initial tests. Also, we captured two person's facial expressions using Vicon system [24] and then transferred the expressions to the scanned faces of the two persons, Face1 and Face2 (Table 1).

Figure 8 shows selected frames of several animated meshes we used in our experiments. Spatio-temporal feature points we have extracted using our algorithm are illustrated as spheres. For the complete sequences along with the extracted feature points, please take a look at our accompanying demo video.

Face1 (happy), Face1 (surprise), Face2 (happy) and Face2 (surprise) contain facial expressions of happiness and surprise of those scanned subjects. The horse and the camel were obtained from the results of Sumner and Jovan Popović's work [20] that are available online [15]. Furthermore, we produced two mesh animations of 70 frames, Face3 and Head, by using nine facial expressions of the face and the head from [20]. More specifically, given an ordered sequence of nine facial expressions, we smoothly morphed each mesh to the next one through a linear interpolation of their vertex coordinates. We also used "walk and whirl" skeletal animations of three women models. Those models share the same mesh topology and were obtained by deforming a template mesh onto three body scans of different subjects. Note that there is high semantic similarity between animation pairs of Face1/Face2, horse/camel, and Face3/Head. It is also the case for three women models.

The color of a sphere represents the temporal scale (red color corresponds to more fast deformations) of the feature point, and the radius of the sphere indicates the spatial scale. Vertex color on surfaces corresponds to amount of deformation (strain and curvature change) observed in each of the animation frame. During experiments we have discovered that our method captures spatio-temporal scales in a robust manner. For example, surface patches around joints of cylinder (Fig. 9, 1a–1e) exhibit different amounts of deformation that occurs at different speeds. The top joint moves fast and consequently the corresponding feature was detected at low temporal scale (red color). However, the mid-joint deforms for a long time and we identify it at high temporal scale (blue color). Moreover, large radii of deforming spheres for both joints make sense and indicate large deforming regions around the features, rather than very local deformation (Fig. 9, 1c). The second row in (Fig. 9, 2a–2e) depicts some of the feature points in horse mesh animation, and the third row (Fig. 10, 3a–3e) corresponds to camel animation. Those two meshes deform in a coherent manner [20], and eventually we detect their spatio-temporal features quite consistently. In the last two rows (Fig. 9, 4a–4e, 5a–5e), we present feature points in mocap-driven face animations of two different subjects. Our subjects were asked to mimic slightly exaggerated emotions during the mocap session. Notice that people normally use different sets of muscles when they show up facial expressions, and therefore naturally we observe some variations in the way their skin deforms.

Our algorithm is implemented in C++. All our tests have been conducted on an Intel Core i7–2600 3.4 GHz machine,

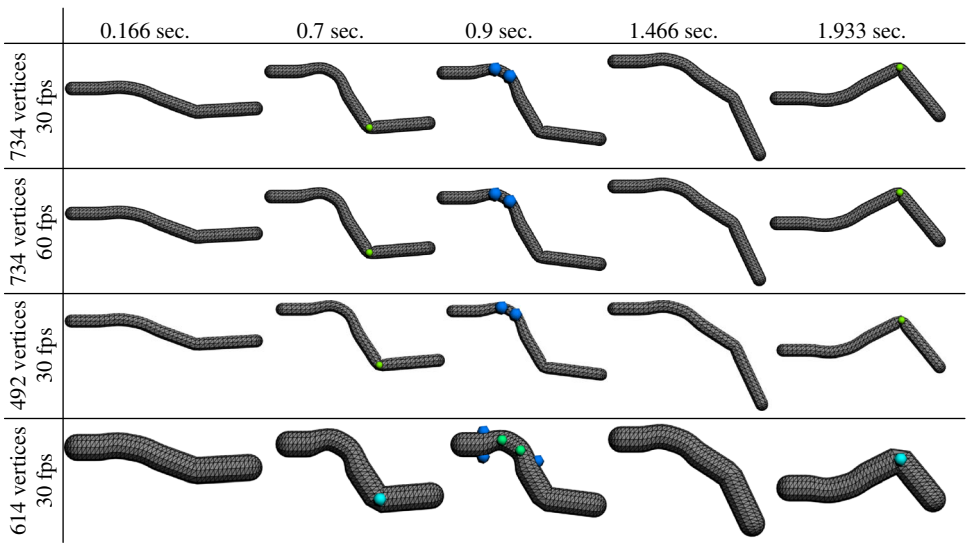

Fig. 8 Results we obtained on varying data sets of bending cylinder animations demonstrate the consistent behavior of our feature detector

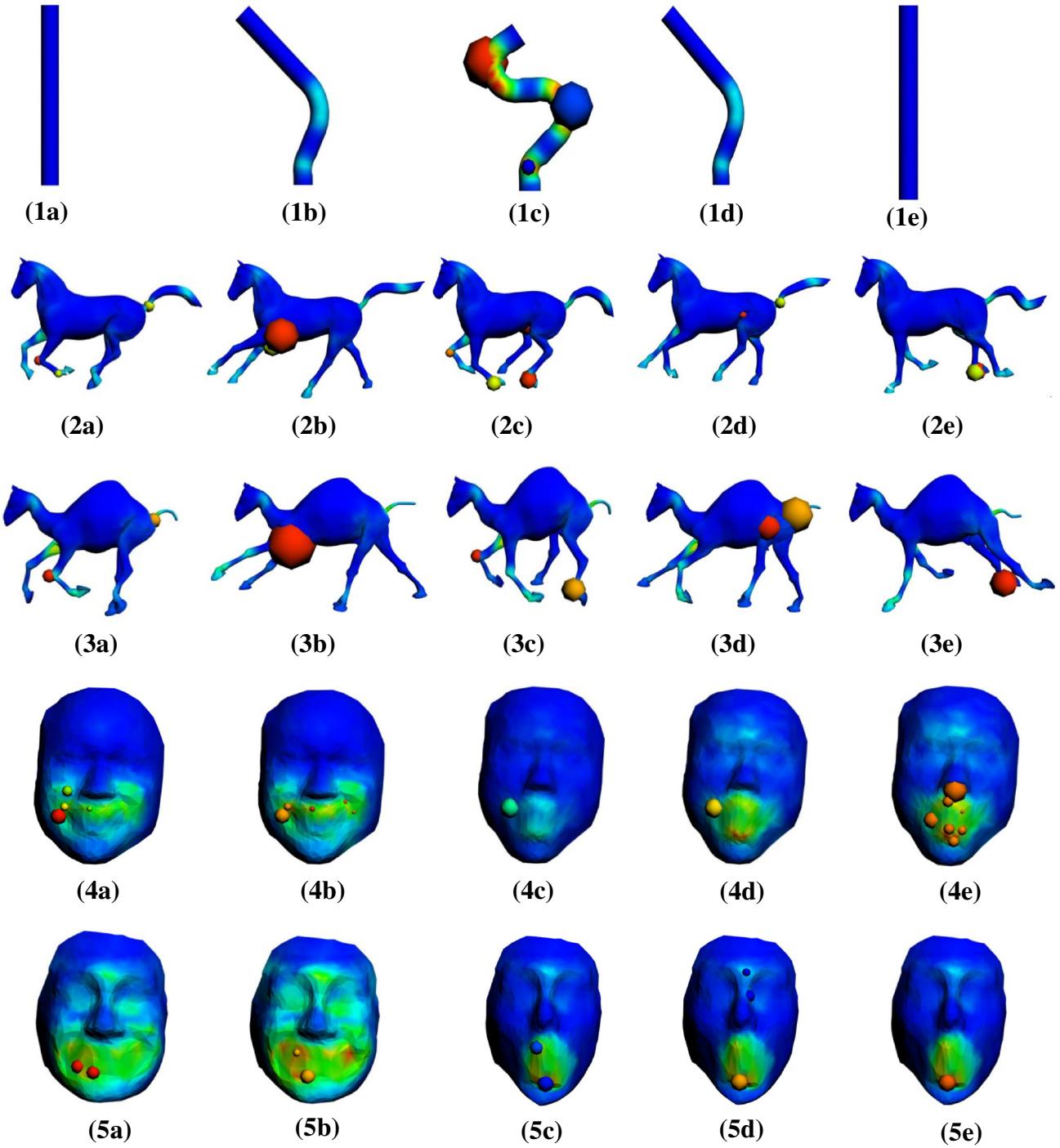

Fig. 9 Dynamic feature points detected by our AniM-DoG framework are illustrated on a number of selected frames of animated meshes. The *color* of a sphere represents the temporal scale (from *blue* to *red*) of the feature point, and the radius of the sphere indicates the spacial scale

with 8GB of RAM. The computation time devoted to full pipeline of the algorithm is approximately 2 min for most of our example data.

Invariance to rotation and scale Invariance of our detector to rotation as well as scale is evident from the definition of our deformation characteristics. Both the strain and the curvature measure we use are invariant to rotation and scale of the animated mesh.

Robustness to changes in spatial and temporal sampling Robustness of our feature detector to changes in spatial sampling is obtained by the adaptive setting of the widths of the box filters. As described in Sect. 5.2, we set the width of the

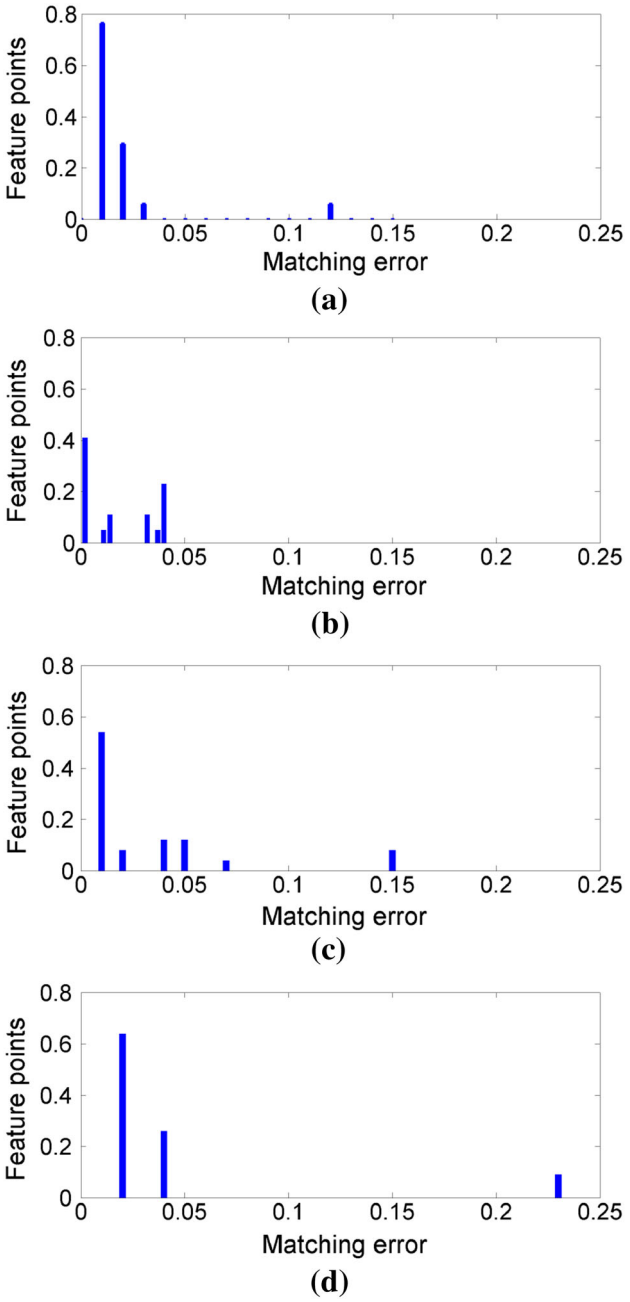

Fig. 10 The error plots of feature points for pairs: **a** Woman1–Woman2, **b** Woman1–Woman3, **c** Woman2–Woman3, **d** camel–horse. We depict the feature matching error on the x-axis as the error (percentage of the error with respect to the diagonal length of the mesh bounding box). The percentage of features with prescribed matching error is depicted on the y-axis. For all four pairs of animations, more than 90 % of features have a matching error less than 0.05

spatial filter as the average edge length of the mesh taken at the initial frame. To demonstrate the invariance to spatial density of the input mesh, we have conducted comparative experiments on two bending cylinders. These two cylinders have identical shape and deformation; they only differ by the number of vertices and the inter-vertex distance. As shown in the first and third rows of Fig. 8, the features are extracted at the same spatio-temporal locations.

Robustness to changes in temporal sampling is obtained similarly as above, i.e., by the adaptive setting of the widths of the box filters. Similar experiments have been conducted by using the two bending cylinders as shown in the first and second rows of Fig. 8. They are perfectly identical except that the temporal sampling of the first one is twice higher than that of the first one. Once again, the extracted feature points are identical in their locations in space and time.

We have further experimented with data sets of similar animations, but with different shape, and spatial and temporal samplings (4th row of Fig. 9, galloping animals and two face models in Fig. 9). Although the extracted features show a good level of consistency, they are not always identical. For example, feature points for the galloping horse and camel do not have the same properties (location, time, tau and sigma). Similar results have been observed for the "face" models. This can be explained by the following facts. Firstly, although the two meshes have deformations that are semantically identical, the level of deformation (curvature and strain) might differ greatly. Secondly, most of these models have irregular vertex sampling, whereas in our computation of the spatial filter width, we assume that the vertex sampling is regular.

6.1 Consistency

Since our method is based on deformation characteristics, it has an advantage of consistent feature point extraction across mesh animations with similar motions. To demonstrate mutual consistency among feature points in different animations, we used animation data that exhibit semantically similar motions. Our technique captures similarity of surface deformations and therefore ensures feature point detection consistency (Fig. 11). In most cases, our method demonstrates high coherency not only in space and time locations of extracted features, but also in their space–time scales σ and τ. The only data sets for which we observed relatively lower consistency of feature detection are the two face mocap sequences. The reason for this lies in inherent difference of people's facial expressions and underlying muscle anatomy.

Additionally, we have performed the quantitative evaluation of the feature extraction consistency as follows. For all feature points we consider only their spatial locations disregarding the time coordinates. Then, using a pair of similarly deforming meshes $\mathcal{M}_S$ and $\mathcal{M}_T$ whose full correspondence $\mathrm{f} : \mathcal{M}_S \rightarrow \mathcal{M}_T$ is known, we find the matching between their feature points F_S and F_T based on the spatial proximity. More precisely, for each feature point $p_S^i \in F_S$, the feature point $p_T^j \in F_T$ that minimizes $d_T^i = \left\| \mathrm{f}(p_S^i) - p_T^j \right\|$ is considered to be the matching one. The distance d_T^i is what we call feature matching error. Histogram plots of

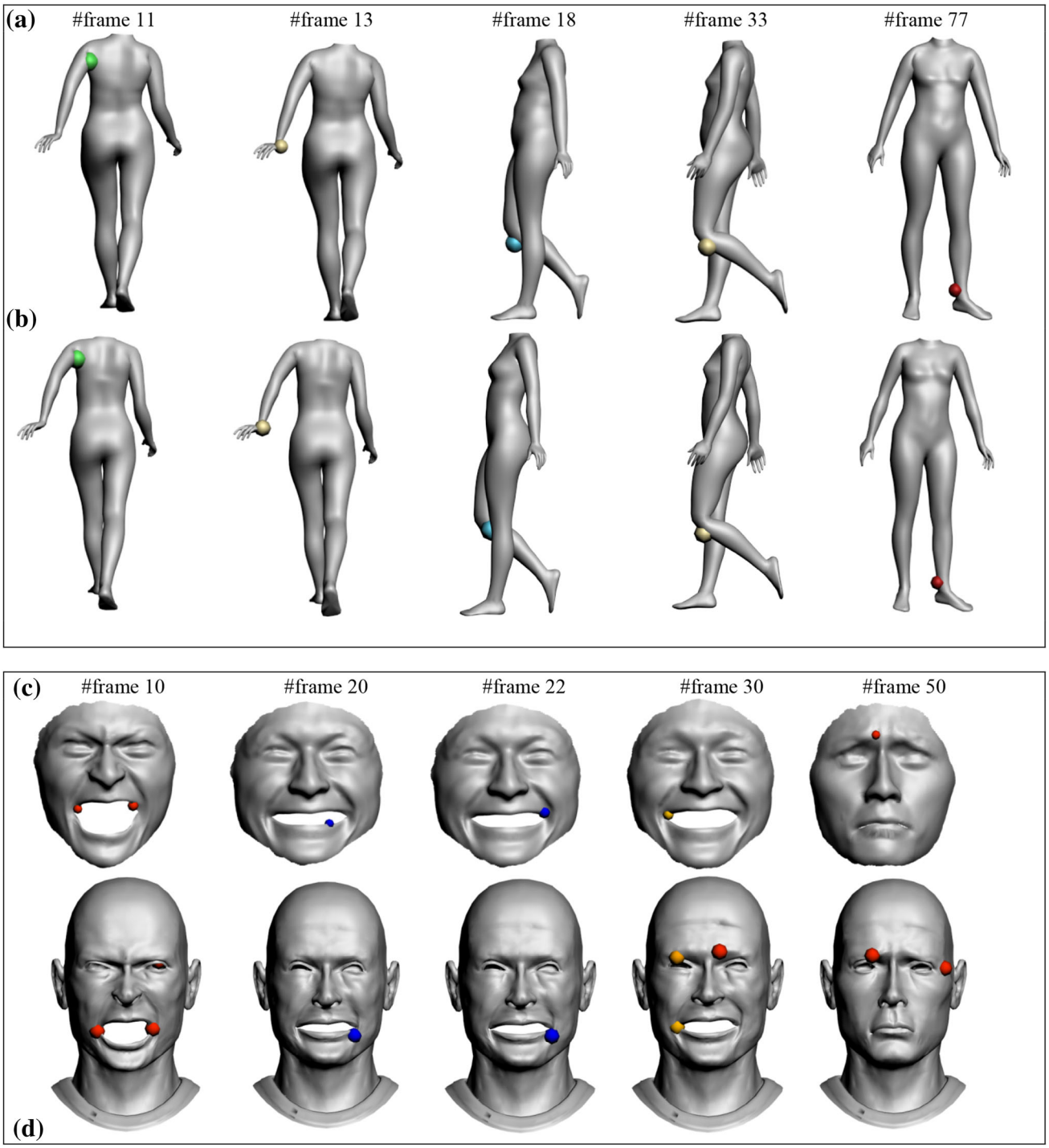

Fig. 11 Inter-subject consistency of feature points extracted from semantically similar mesh animations. Rows **a**, **b** depict subset of feature points extracted from walking subject sequences and **c**, **d** from face animations. Note that each column corresponds to an identical frame of animations

feature matching errors are depicted in Fig. 10. Obtaining the full correspondence for walking women models was straightforward, because they share the same mesh topology. For horse/camel, we obtained a full per-triangle correspondence from [20], which we converted to a per-vertex correspondence.

6.2 Comparison to the ground truth

We have validated our method by comparing the feature points to the manually defined ground truth. We asked six volunteers to paint feature regions on the animated meshes using an interactive tool. The task was to mark locations at

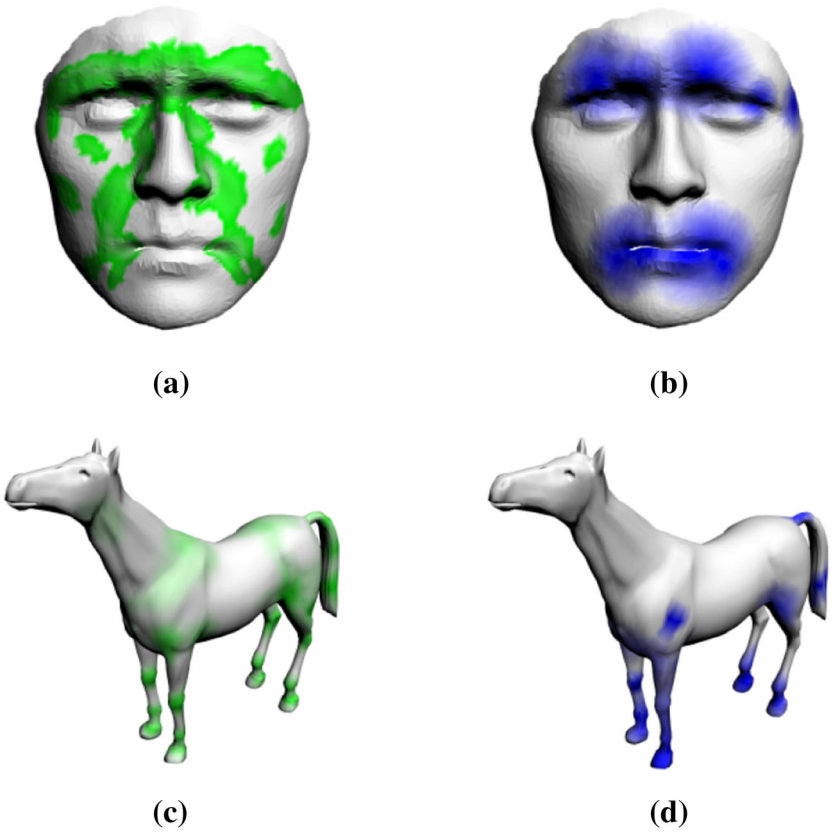

Fig. 12 Comparison of the ground truth (**a**, **c**) to the feature regions computed by our method (**b**, **d**). For each vertex, *color* intensity depicts the accumulative number of its appearances during the animation. *Green* and *blue colors* were used for the ground truth and our computed feature regions, respectively

which salient surface deformation behavior can be observed from the user point of view. Each of them could play and pause the animation at any moment and mark feature regions by a color. To simplify the task, the time duration of each feature region was not considered. Since the per-vertex selection can be error prone, we deliberately allow users to select a region on the surface instead of a single vertex. By aggregating the feature regions from all volunteers, we generated a color map of feature regions. More specifically, for each vertex we summed up and averaged the number of times it has been included in the user-selected regions. The aggregated ground truth was then converted into a surface color map, as depicted in Fig. 12a, c. Note, that eyes do not belong to feature regions of face animations, since the user's task was to define features based on the surface deformation rather than geometric saliency or naturally eye-catching regions.

To compare our results with respect to the ground truth, we compute for every feature point p its feature region of neighboring vertices q such that $\mathcal{L} = \{q : d_{\mathcal{M}}(q, p) < \sigma_k\}$, where $d_{\mathcal{M}}(\cdot, \cdot)$ is a within-surface geodesic distance and σ_k is the corresponding scale value at which the feature was detected. Similarly to the ground truth, for each vertex of the mesh we count the number of occurrences in feature regions during the animation and convert the numbers to the surface color map as shown in Fig. 12b, d. We observe a good level of positive correlation between the computed feature regions and the ground truth.

7 Discussion

Our method and results could be extended and applied to a number of useful applications. We describe some of the ideas below while leaving their developments as future works.

Animated mesh simplification As it has been noted in earlier works on simplification of dynamic (deforming) meshes [8,21], it is preferable to allocate bigger triangle budget for regions of high surface deformation while simplifying mostly rigid regions. Our algorithm could be adopted in these as it detects feature points that are exactly in deforming regions. Their spatial scales σ_k can be used to define regions around features where the mesh must keep denser sampling during simplification. For instance, the spatial scale of the feature points can be used to define regions where the mesh must be densely sampled during simplification. The temporal scale can also be used to dynamically determine the triangle budget around the feature point, when designing a time-varying simplification technique. A very small temporal scale implies either a short duration or a high speed of the animation; thus, one may assign low priority to the feature point. In the same

way, the region around a feature point with large temporal scale will be prioritized when allocating the triangle budget. Another use of the temporal scale is in the maintenance of the hierarchy. When transferring the previous frame's hierarchy to one better suited for the current frame in a time-critical fashion, the algorithm can use the temporal scale of an FP as a "counter" to determine whether to update or reuse the node corresponding to the region around the FP. By processing the nodes corresponding to the spatio-temporal feature points in an order of decreasing temporal scale, one can economize the time for the per-frame maintenance of the hierarchy while keeping the animation quality as much as possible.

Viewpoint selection With increasing advances in scanning and motion capture technologies, animated mesh data have become more and more available today. Thus, it is very practical to have a tool for automatic viewpoint selection for the preview of the motion in animation repositories. The idea behind that is to let a user to quickly browse the animation data from the point that maximizes the visibility of mesh deformations. With such viewpoint selection, the user benefits from a better perception of the animation. One equally handy and straightforward way to automatically select optimal viewpoint is to compute the one which maximizes the number of visible feature points through the optimization. We note that our spatio-temporal feature points can simplify the selection of good viewpoint(s). For instance, the quality of a viewpoint could be defined as a function of the visibility of the spatio-temporal feature points in terms of the total number, temporal variability and the concavity of the projected feature region (as defined by the spatial and temporal scales), etc. Interested reader may refer to an optimization technique proposed in [14] on saliency-based viewpoint selection for static meshes.

Animation alignment Another interesting application could be animated mesh alignment. Considering the consistency of the extracted feature points, their scale values can be employed for the temporal alignment. Given sets of features P and $P^{'}$ extracted from a pair of similar animations, we consider corresponding sequences $\{(\sigma_1, \tau_1), \ldots, (\sigma_n, \tau_n)\}$, $\{(\sigma_1^{'}, \tau_1^{'}), \ldots, (\sigma_m^{'}, \tau_m^{'})\}$ of spatio-temporal feature scales aligned along the time they were detected. Existing algorithms of sequence alignment such as [6] can then be used to compute the temporal alignment between them. In addition to the spatial and temporal scales, more sophisticated feature descriptors can also be used to compose the sequences.

Animation similarity We can also think of extending the above-mentioned animation alignment algorithm toward a measurement of animation similarity. From the feature sequence alignment map, we can sum up all penalty gaps, i.e., some predefined costs for all features for which no match can be found. That cost function could serve as a distance metric between the animations and hence be a measure of dissimilarity/similarity. Note that an important by-product of the animation similarity is the animated mesh retrieval, which is particularly beneficial in emerging dynamic data repositories.

8 Conclusion

We have presented a new feature detection technique on triangle mesh animations based on linear scale-space theory. We introduced a new spatio-temporal scale representation of surface deformation in mesh animations. Furthermore, we developed extension of classical DoG filter to the spatio-temporal case. The latter allows our method to robustly extract repeatable sets of feature points over different deforming surfaces modeled as triangle mesh animations. We carried out experimental validation of detected features on various types of data sets and observed consistent results. Our approach has shown robustness to spatial and temporal sampling of mesh animation. In our future research, we intend to focus on feature point descriptor that could be useful for applications such as matching between animations.

Descriptors Our feature detector could be extended to detector descriptor. One straightforward idea of the feature point descriptor could be the following. Suppose that we are given a space–time neighborhood around feature point v_i^f which consists of its k-ring space neighborhood over a range of $[f - l, \ldots, f + l]$ frames. Note that the values of k and l can be adjusted to reflect the characteristic scale of the feature. Flattening of k-ring regions for each frame of the range produces a volumetric stack of planar mesh patches. Then, proceeding in the spirit of 3D SIFT descriptor [18], we can estimate histograms of DoG gradients computed inside the spatio-temporal volume around the feature point. Further, Euclidean or Earth Mover's distance between the histograms could be used for measuring similarities of features within mesh animation or between different animations.

Acknowledgments We acknowledge Robert W. Sumner for providing triangle correspondences of the horse and camel models. We also thank Frederic Larue and Olivier Génevaux for their assistance with the facial motion capture. This work has been supported by the French national project SHARED (Shape Analysis and Registration of People Using Dynamic Data, No.10-CHEX-014-01).

References

1. Andonie, R., Carai, E.: Gaussian smoothing by optimal iterated uniform convolutions. Comput. Artif. Intell. **11**(4), 363–373 (1992)
2. Alliez, P., Cohen-Steiner, D., Devillers, O., Levy, B., Desbrun, M.: Anisotropic Polygonal Remeshing. ACM Trans. Graph. **22**(3), 485–495 (2003)

3. Bay, H., Ess, A., Tuytelaars, T., van Gool, L.: Speeded-up robust features (SURF). Comput. Vis. Image Underst. (CVIU) **110**(3), 346–359 (2008)
4. Castellani, U., Cristani, M., Fantoni, S., Murino, V.: Sparse points matching by combining 3D mesh saliency with statistical descriptors. Comput. Graph. Forum **27**(2), 643–652 (2008)
5. Darom, T., Keller, Y.: Scale-Invariant features for 3-D mesh models. IEEE Trans. Image Process. **21**(5), 2758–2769 (2012)
6. Gotoh, O.: Optimal sequence alignment allowing for long gaps. Bull. Math. Biol. **52**, 359–373 (1990)
7. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of 4th Alvey Vision Conference, pp. 147–151 (1988)
8. Kircher, S., Garland, M.: Progressive multiresolution meshes for deforming surfaces. In: ACM SIGGRAPH Symposium on Computer, Animation, pp. 191–200 (2005)
9. Lian, Z., Godil, A., Xiao, J.: Feature-preserved 3D canonical form. Int. J. Comput. Vis. **102**, 221–238 (2013)
10. Laptev, I., Lindeberg, T.: Interest point detection and scale selection in space-time. In: Griffin, L.D., Lilholm, M. (eds.) Scale Space Methods in Computer Vision. Lecture Notes in Computer Science, vol. 2695, pp. 372–387. Springer, Berlin, Heidelberg (2003)
11. Lindeberg, T.: Scale-space theory: a basic tool for analyzing structures at different scales. J. Appl. Stat. **21**(2), 224–270 (1994)
12. Lindeberg, T.: Feature detection with automatic scale selection. Int. J. Comput. Vis. **30**, 79–116 (1998)
13. Lowe, D.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
14. Lee, C.-H., Varshney, A., Jacobs, D.W.: Mesh Saliency, ACM Transactions on Graphics. Proc, SIGGRAPH (2005)
15. Mesh data, http://people.csail.mit.edu/sumner/research/deftransfer/data.html
16. Mikolajczyk, K., Schmid, C.: Indexing based on scale invariant interest points. IEEE Int Conf. Comput. Vis. (ICCV), pp. 525–531 (2001)
17. Pauly, M., Keiser, R., Gross, M.: Multi-scale feature extraction on point-sampled surfaces. Comput. Graph. Forum **22**(3), 281–289 (2003)
18. P. Scovanner, S. Ali, and M. Shah: A 3-Dimensional SIFT Descriptor and Its Application to Action Recognition. Proceedings of the 15th International Conference on Multimedia, pp. 357–360, (2007)
19. Sipiran, I., Bustos, B.: A robust 3D interest points detector based on harris operator. Eurographics workshop on 3D Object Retrieval (3DOR), pp. 7–14 (2010)
20. Sumner, R., Popovic, J.: Deformation transfer for triangle meshes. ACM Trans. Grap. **23**, 3 (2004)
21. Shamir, A., Pascucci, V., Bajaj, C.: Multi-resolution dynamic meshes with arbitrary deformations. In: Proceedings of IEEE Visualization, pp. 423–430 (2000)
22. Salden, A.H., ter Haar Romeny, B.M., Viergever, M.A.: Linear scale-space theory from physical principles. J. Math. Imaging Vis. **9**, 103–139 (1998)
23. Sumner, R.W., Zwicker, M., Gotsman, C., Popović, J.: Mesh-based inverse kinematics. ACM Trans. Graph. (TOG) **24**(3), 488–495 (2005)
24. Vicon motion capture system, http://vicon.com
25. Zaharescu, A., Boyer, E., Varanasi, K., Horaud, R.: Surface feature detection and description with applications to mesh matching. IEEE Comput. Vis. Pattern Recognit. (CVPR), pp. 373–380 (2009)

Vasyl Mykhalchuk received his B.Sc. degree in computer science from the National University of Kyiv-Mohyla Academy, Kyiv, Ukraine, in 2007 and M.Sc. degree in computer science from the National University of Kyiv-Mohyla Academy, Kyiv, Ukraine, in 2009. During master program he was working on a problem of matching of image stereo pairs. From 2009 to 2010, he held a software engineering position at a video game development company. He is currently a Ph.D. candidate in Computer science and research assistant in computer graphics group at ICube laboratory, University of Strasbourg, France. His research interests include shape analysis, and registration of static and dynamic geometry.

Hyewon Seo is a CNRS (Centre National de la Recherche Scientifique) research fellow at the University of Strasbourg, France, since 2009. She has B.Sc. and M.Sc. degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST). After obtaining Ph.D. from the University of Geneva in 2004, she became an assistant professor and supervisor of the Computer Graphics Laboratory in the Computer Science and Engineering Department at the Chungnam National University, South Korea. Her research interests include human body modeling, virtual prototyping, and shape analysis.

Frederic Cordier is an associate professor at the University of Haute-Alsace. His research interests include 3D modeling and texturing, human–computer interaction and physics-based simulation. Frederic has a Ph.D. in computer science from the University of Geneva, Switzerland.