Deformable Mesh Animation from Single-View Video

Richard Stebbing *
University of Oxford

Aaron Hertzmann †
Adobe Systems, Inc.

Andrew Fitzgibbon ‡
Microsoft Research

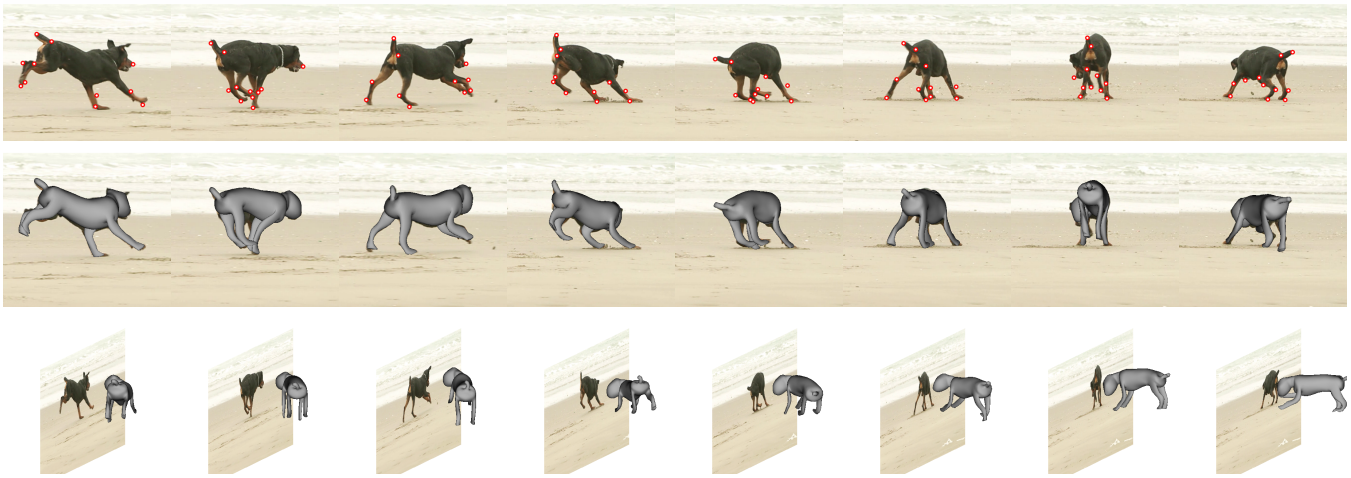

Figure 1: *Our method extracts time-varying 3D geometry from a single 2D video using silhouettes and sparse point tracks.* ***Top row:*** *Selected frames from 196-frame video of a dog running on the beach with sparse point tracks overlaid.* ***Middle row:*** *3D reconstruction overlaid on the input frames.* ***Bottom row:*** *3D reconstructions seen from an oblique view.*

Abstract

We present an animation capture technique which derives dense 3D shape and motion from a monocular 2D video sequence, using the object silhouette as the primary shape cue, augmented with a small number of user-specified keypoints. A template mesh is deformed to fit the image data. The deformations are controlled by regularizers based on the As-Rigid-As-Possible (ARAP) principle, which provides effective constraints on the depth ambiguities inherent in single-view reconstruction while allowing a wide range of shapes to be reconstructed. The use of monocular video allows library footage to be used as the source material, and examples on several library sequences are included. We require user annotation of the video, and the models we recover do not show the level of detail or accuracy of those captured in a studio setting, but conversely they exhibit animals and movements that would be challenging to capture in a studio.

1 Introduction

> *The first duty of the cartoon is not to picture or duplicate real action or things as they actually happen, but to give a caricature of life and action [...] I definitely feel that we cannot do the fantastic things based on the real, unless we first know the real.*
>
> *Walt Disney, 1935 [Girveau 2007, page 90]*

Computer graphics, unfettered by physical laws, allows us to experience alternative realities limited only by the CG artist's imagination. Even outlandish realities, however, are made more believable by including objects which behave according to our experience of the real world. To do so requires convincing 3D animations of humans, animals and natural phenomena. Generating such animations demands great skill from the animator, and often realism can be achieved only by capturing animation data from the real world. The goal of this paper is to capture 3D animations from 2D video sequences captured from a single viewpoint using a standard video camera. This allows the animator to select from millions of hours of existing library footage to find the ideal reference or input motion. Such a system would fundamentally change the feasibility of incorporating realistic animations into users' projects, even for amateur animators. The approach we describe in this paper is a step towards such a system, and even though the models we recover are not of the fidelity that one could expect from a multi-camera capture studio, the range of motions and scenarios that can be captured is considerably more. Because the system acquires dense time-varying geometry, the captured motions might also be useful as a basis for effects such as relighting and retexturing.

1.1 Background

Animation capture is a subject with a long history, from rotoscoping at Disney in the 1930s, to marker-based motion capture, and matchmoving of camera trajectories from live-action footage. Recent research has made significant progress in the capture of humans and domesticated animals. However, most existing systems for capturing the motion of the nonrigid 3D world suffer one or more significant limitations. The first such limitation is to require **multiple views** of the 3D scene to be captured simultaneously [Vlasic et al. 2008; de Aguiar et al. 2008; Gall et al. 2009]. While this is easy to arrange in studio or laboratory environments, it is rather more difficult in the outside world. In many cases, obtaining even one good viewpoint of a rare or wild animal is a challenge; to obtain another with sufficient baseline for 3D reconstruction may be impossible. And of course, stock footage is almost exclusively monocular.

Considering just the monocular case, researchers have developed many methods to extract 3D motion from video footage. We dis-

*e-mail:richard.stebbing@eng.ox.ac.uk
†email:hertzman@adobe.com
‡email:awf@microsoft.com

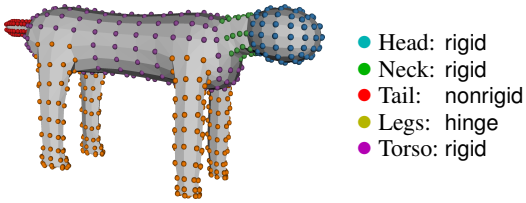

Figure 2: ***User-supplied template mesh for the dog sequence****. Note that the mesh shape need not be accurate. The user may paint the mesh to indicate regions which share common motion characteristics. Several markup types are illustrated here. The head and neck are each hinted to have globally near-rigid motion, that is to say the ARAP rotations for all the same-coloured vertices are shared. The tail has generic ARAP-regularized motion: each vertex has 3 degrees of rotational freedom. The legs and torso each have 1-dof ARAP, so the rotation angles are estimated per-vertex, but the rotation axis is globally estimated.*

Figure 3: ***User-supplied inputs to our algorithm. Left:*** *point correspondences. Any easily identifiable keypoints on the object surface may be tracked using standard tracking software. In our examples, we use a simple homegrown interactive rotoscoping interface to specify point tracks.* ***Right:*** *silhouette mask, extracted using Adobe After Effects Roto Brush.*

tinguish two main strands of research: model-based adaptation and tracking, and general nonrigid structure from motion. Model-based tracking proceeds from a **parameterized 3D model**, and an optimization procedure adapts the model parameters to fit image data such as silhouettes or point tracks. Parameterized models include rigged skeletons as fitted to silhouette data (recent examples are [Shih et al. 2009; de la Gorce et al. 2011; Pons-Moll et al. 2011; Salzmann and Urtasun 2010]), or low-dimensional shape models of classes such as faces [Blanz et al. 2003; Vlasic et al. 2005] and the human body [Balan et al. 2007]. However the building of such parameterized models is a significant effort, either having an artist build a rigged model, or obtaining multiple laser scans to compute low-dimensional shape bases. While such effort is justified for the special case of human capture, it is more difficult to foresee for arbitrary object classes. Certainly we know of no system that has been shown to track animals in sequences of the complexity of our examples. Additionally, if the goal is to recover 3D shape models from video, model-based tracking provides only a partial solution: the parameterized model will typically fit the input data only approximately, ignoring clothing for example [Balan et al. 2007]. While this is an advantage if the model parameters are themselves the desired output, it is insufficient if dense shape recovery is also a goal of the capture.

Without a predefined parameterized model, recovering general 3D shape from each frame of a monocular sequence is the problem of nonrigid structure from motion (NRSFM). One approach poses the problem as the joint recovery of a parameterized shape model as well as the per-frame instance parameters. Bregler et al. [2000] represented the parameterized model with a linear basis, a representation that works well for simple objects like faces, but does not represent articulation accurately. Yan and Pollefeys [2008] treat articulated, non-rigid motion. Ross et al. [2010] estimate articulated structure from point tracks by alternating between kinematic chain moves and parameter updates. An alternative casting of the problem is to impose a smoothness prior on the recovered spacetime model. The early work of Ullman [1984] proposed that the per-frame reconstructions should be related as closely as possible by a rigidity constraint, measured by comparing the distances between 3D point pairs in the reconstructions. Taylor et al. [2010] use a local rigidity constraint and demonstrate impressive results on a range of objects. Akhter et al. [2010] use a temporal basis that enforces smoothness of point trajectories, without modeling relationships between points. The major disadvantage of all these methods is the requirement for **dense feature tracks** to be extracted from the footage. Unfortunately, for almost all animals one might want to capture, dense tracks cannot be extracted—few animals have spots, and even giraffes have featureless lower legs and tails. Notice that some point tracks are generally obtainable, for example near eyes, or at sharp joints, but are far too sparse to employ the above methods, which require tens of tracks as a bare minimum. Recently, Cashman and Fitzgibbon [2013] recovered a linear basis model from a combination of silhouettes and sparse point correspondences. They show successful examples on birds and dolphins, but dependence on a linear model means that articulated classes such as polar bears are presented as a failure case.

1.2 Our approach

Our approach combines aspects of the above methods to extract animated meshes from monocular video with sparse point tracks. Like Vlasic et al. [2008], we start from a rigid template, which is deformed to fit each input image. Like Ullman [1984] we use rigidity between 3D models as a regularizer, although we employ a novel definition based on the As-Rigid-As-Possible (ARAP) formulation [Alexa et al. 2000; Sorkine and Alexa 2007] which has been successfully used in a variety of mesh deformation applications. Like Cashman and Fitzgibbon [2013] we combine silhouette information with sparse point tracks to render the estimation tractable and stable. The main **technical contributions** of this work are:

1. Our model comprises a novel hierarchical decomposition that captures the object shape in a *core* geometry, and individual video frames in *instance* geometries. ARAP regularizers describe both the deformation from the core geometry to each instance (frame), and the deformations from frame-to-frame.

2. The use of ARAP as a prior shape model for nonrigid structure from motion. Previous work has proposed the use of basis shapes and/or temporal smoothness, which do not accurately constrain the motion of articulated, non-rigid motion.

3. A new "Hinted ARAP" scheme (§3.2.4) which allows the user to use simple markup on the template mesh to indicate collections of model vertices which have similar motion characteristics, providing finer control over the ARAP regularizer without requiring explicit specification of a skeleton or rig.

We demonstrate reconstruction on a variety of animals of various sizes and performing various motions, including walking, running, leaps and turns. We show, for the first time, dense 3D shape and motion estimation from raw footage of articulated, non-rigid animals. No specialized equipment is required: all our examples are demonstrated on stock footage obtained from the web.

The major **limitations** of the work are as follows. Because we work from video data, and primarily rely on silhouette constraints, we do

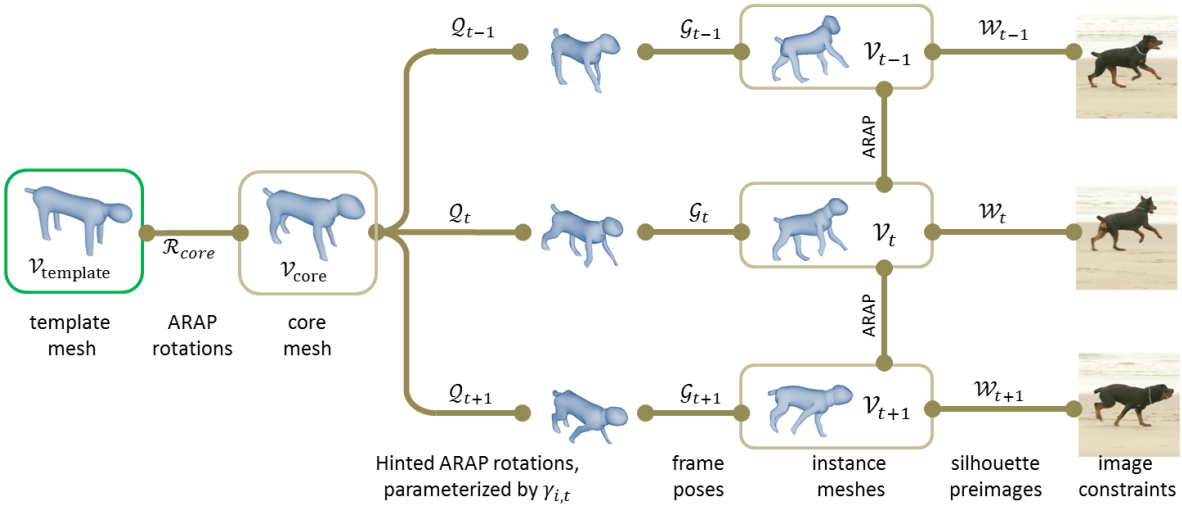

Figure 4: ***Model structure.*** *Each image corresponds to a 3D surface called an* instance mesh $\mathcal{V}_t$*. Silhouette points in the images correspond to preimage points $\mathcal{W}_t$ on the meshes, and point tracks $\mathbf{c}_t$ correspond to mesh vertices (not shown). The instance meshes are deformations of a single* core mesh $\mathcal{V}_{core}$*. Each core-to-instance deformation comprises an As-Rigid-As-Possible (ARAP) deformation (with rotations $\mathcal{Q}_t$), followed by a rigid+scale transformation $\mathcal{G}_t$. The frame-to-frame deformations are also assumed to be ARAP. The core is regularized according to the template mesh $\mathcal{V}_{template}$, provided by a user. All other variables shown in the diagram are unknowns in the optimization.*

not capture detailed geometry, and our results fall short of the quality obtained by laboratory-based capture systems. We require some user input for matting and point tracking, and our optimization currently takes several hours. The number of parameters in our system is large: eleven energy terms, each with its own weight. However many have natural interpretations in terms of image noise, so the number of parameters that might need to be tweaked in practice is relatively few, and a small number of parallel runs should be sufficient to achieve a good result without further interaction.

2 Model and Inputs

Given an input video sequence with T frames of a moving animal, we seek to recover the animal's time-varying 3D geometry. We represent the geometry as a triangle mesh with fixed topology and time-varying vertex locations $\mathbf{v}_i^t \in \mathbb{R}^3$, indexed by time t and vertex index i. We will write the model at frame t as the list

$$\mathcal{V}_t = \{\mathbf{v}_i^t\}_{i=1}^N \tag{1}$$

where N is the number of vertices in the mesh. The mesh topology is represented by the neighbourhood function $\mathcal{N}(i)$ which returns the list of vertices adjacent to (sharing an edge with) vertex i.

The algorithm takes several inputs from a user. First, the user supplies a template mesh $\mathcal{V}_{\text{template}}$ (Fig. 2) that specifies the topology of the surface, anchors the core geometry, and is annotated with ARAP hints (see below). Second, the user segments the animal from the video using an off-the-shelf tool. Finally, the user specifies sparse point correspondences between image pixels and the base mesh vertices.

3 Energy Function

We express the problem as optimizing an energy function, expressed as a weighted sum of terms:

$$E_{full} = \sum_\ell w_\ell E_\ell \tag{2}$$

The optimization includes data terms for the animal silhouettes and point correspondences, ARAP terms that penalize non-rigid deformation from a *core geometry* that represents the animal in a default pose, and spatial and temporal regularization terms. The free variables in the optimization are the per-frame *instance geometries* $\mathcal{V}_1, \ldots, \mathcal{V}_T$ and the core geometry $\mathcal{V}_{\text{core}}$, as well as a number of auxiliary variables: the camera motion, ARAP rotations, rotation bases, and silhouette point correspondences. Before explaining the energy terms and unknowns in more detail we offer a step-by-step development of the overall structure of the energy. Each stage may introduce additional unknowns, which are indicated in the argument lists of the E_ℓ as they are introduced.

The first term is the data term, dependent only on the instance geometries, which measures fit to the silhouette and user-supplied point constraints. Its calculation is independent from frame to frame, so is a sum over frames:

$$E_{data}(\mathcal{V}_{1..T}) = \sum_{t=1}^{T} E_{data}^t(\mathcal{V}_t) \tag{3}$$

If one were to fit the model using only data terms, the problem would be massively underconstrained: hundreds of vertices per frame, only a very small subset of which are attached to image data. Thus regularization is crucial to obtain reasonable results. Following Ullman [1984], we define a motion smoothness term which encodes the prior assumption that the object's shape changes smoothly over time. We assume the existence of a deformation measure D which computes the difference between a pair of meshes $\mathcal{V}$ and $\mathcal{U}$, returning zero if the meshes are related by a rigid transform, and then define the motion energy as

$$E_{motion}(\mathcal{V}_{1..T}) = \sum_{t=2}^{T} D(\mathcal{V}_{t-1}, \mathcal{V}_t) \tag{4}$$

This term encourages the instance shapes to change slowly over time. However, the motion penalty should not be too strongly imposed if we want to model interesting real-world motions, whereas a weak motion penalty does not offer enough regularization to constrain the shape.

In order to ensure that the shape and motion are consistent across the whole sequence, we require that each model instance be a smooth deformation of some a priori unknown *core geometry* $\mathcal{V}_{\text{core}}$ by defining the energy term

$$E_{\textit{instance-core}}(\mathcal{V}_{1..T}, \mathcal{V}_{\text{core}}) = \sum_{t=1}^{T} D(\mathcal{V}_t, \mathcal{V}_{\text{core}}) \tag{5}$$

where here we will use a more constrained version of D.

We also measure deviation of the core geometry to the user-supplied template $\mathcal{V}_{\text{template}}$:

$$E_{\textit{core-template}}(\mathcal{V}_{\text{core}}) = D(\mathcal{V}_{\text{core}}, \mathcal{V}_{\text{template}}) \tag{6}$$

The weighting of this term proves to be one of the important parameters of our system. If the sequence provides rich geometric cues—the subject moves around so that a range of viewpoints are exercised—it can be downweighted. On the other hand, in the limiting case that only a single frame is available, the problem is reduced to single-view reconstruction, and a stronger shape prior is needed.

Our energy is the weighted sum of the terms *data, motion, instance-core, core-template*, as well as terms *core-template-reg*, *instance-core-motion*, *camera-R-motion*, *camera-scale-motion*, which will be defined below. The model is displayed pictorially in Fig. 4.

3.1 Data terms

The components of the energy that cause the model to align to image data are presented now, comprising the silhouette term and the point correspondence term.

Silhouette matching. During preprocessing, the user segments the animal from the background (Fig. 3), using an off-the-shelf algorithm. We use the Adobe After Effects Roto Brush [Bai et al. 2009] for all examples in this paper. The algorithm then extracts a set of 2D silhouette points $\mathbf{s}_j^t$ for each frame t, by traversing the silhouette. These silhouette points form a set of one or more ordered loops, and are subsampled. A world space normal $\mathbf{n}_j^t$ for each silhouette point is estimated as follows. The silhouette tangent is computed by central differences $[x, y]^T = \mathbf{s}_{j+1}^t - \mathbf{s}_{j-1}^t$. The world space normal $\mathbf{n}_j^t$ is approximated as $[y, -x, 0]^T / \sqrt{x^2 + y^2}$ [Cashman and Fitzgibbon 2013], by incorporating the constraint that the normal lies on the contour generator. We assume that the geometry is viewed under weak-perspective projection. As the geometry for each frame is represented in camera coordinates, image projection is given by a canonical 2×3 orthographic projection matrix $\mathbf{\Pi} = [I_{2\times 2}|\mathbf{0}]$.

Each silhouette point $\mathbf{s}_j^t$ corresponds to an unknown *preimage point* $\mathbf{w}_j^t$ that lies on the triangle mesh $\mathcal{V}_t$. This introduces a new set of unknowns $\mathcal{W}_t = \{\mathbf{w}_j^t\}$ per image. A preimage point is represented as a triangle index, together with barycentric coordinates within that triangle. The 3D position of the point is computed by the standard barycentric interpolation, written $\mathbf{b}(\mathbf{w}_j^t; \mathcal{V}_t)$. The silhouette term sums across all silhouette points in frame t:

$$E_{\textit{sil}}^t(\mathcal{V}_t, \mathcal{W}_t) = \sum_j ||\mathbf{s}_j^t - \mathbf{\Pi}\mathbf{b}(\mathbf{w}_j^t; \mathcal{V}_t)||^2 \tag{7}$$

where $\mathbf{b}(\mathbf{w}; \mathcal{V}_t)$ performs barycentric interpolation within the triangle containing $\mathbf{w}$ (Fig. 5). Furthermore, the surface normals should match the estimates:

$$E_{\textit{nrm}}^t(\mathcal{V}_t, \mathcal{W}_t) = \sum_j w_j^t(\mathbf{w}_j^t; \mathcal{V}_t)\, ||\mathbf{n}_j^t - \mathbf{N}(\mathbf{w}_j^t; \mathcal{V}_t)||^2 \tag{8}$$

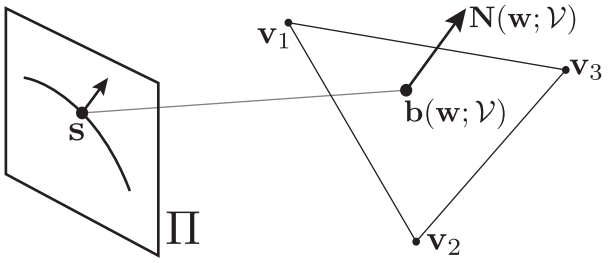

Figure 5: ***Silhouette projection terms.*** *An observed silhouette point* $\mathbf{s}$ *must correspond to a 3D point* $\mathbf{b}(\mathbf{w}, \mathcal{V})$ *on the surface, parameterized by preimage coordinates* $\mathbf{w}$*, which include a triangle index and barycentric coordinates. A "smooth" normal* $\mathbf{N}$ *is computed at a surface point by Phong interpolation of vertex normals. This normal must lie on the contour generator for silhouette points, and the observed normal must be the projection of this normal.*

where $\mathbf{N}(\mathbf{w}; \mathcal{V}_t)$ computes an approximate surface normal at $\mathbf{w}$ by Phong interpolation using the barycentric coordinates. In particular, per-vertex normals are defined by face-averaging, and then normals are defined by barycentric interpolation, followed by normalization to unit length. The weighting w_j^t is used to prevent problems with very small triangles where the derivatives can be extremely large, and thus the optimizer can get stuck. In particular, the algorithm computes the surface area of each vertex's one-ring; these areas are interpolated according to the barycentric coordinates of $\mathbf{w}_j^t$ to get w_j^t.

Figure 5 illustrates these quantities. It is perhaps worth noting that our model is not an explicitly parameterized smooth surface, but a triangle mesh with interpolated normals. This means that the contour generator (the set of silhouette preimages) passes through faces, rather than being on edges as it would generically be in a polygonal model.

Contour generator smoothness. Following Cashman and Fitzgibbon [2013], the silhouette preimages are encouraged to be piecewise smooth, using an energy of the form

$$E_{\textit{sil-smooth}}^t(\mathcal{W}_t) = \sum_j d(\mathbf{w}_j^t, \mathbf{w}_{j+1}^t) \tag{9}$$

where $d(\cdot, \cdot)$ is a robust distance measure (we use a truncated quadratic on geodesic distance) between preimage points. The summation is applied over adjacent pairs on each silhouette loop.

Point correspondences. The user also provides a few point correspondences on the animal, by clicking image points in a simple UI. A correspondence is a tuple $(i, t, \mathbf{c})$ indicating that vertex $\mathbf{v}_i^t$ projects to a 2D point $\mathbf{c}$ in frame t, and the set of all correspondences is denoted $\mathcal{C}$, with the restriction to frame t being denoted $\mathcal{C}_t = \{(i, \mathbf{c}) | (i, t, \mathbf{c}) \in \mathcal{C}\}$. This gives an energy

$$E_{\textit{point}}^t(\mathcal{V}_t) = \sum_{(i,,\mathbf{c})\in\mathcal{C}_t} ||\mathbf{c} - \mathbf{\Pi}\mathbf{v}_i^t||^2 \tag{10}$$

These correspondences could also be detected automatically by a feature matching algorithm. For the animals that we consider, these correspondences are necessarily very sparse, as there are few good corner features on them; they provide guidance for overall rigid motion and for rotation.

Summing the above terms (*sil, nrm, sil-smooth, point*) yields the data term $E_{\textit{data}}^t(\mathcal{V}_t, \mathcal{W}_t)$.

3.2 Regularization terms

The key to our regularization terms is in the choice of the deformation measures D. We first describe the general form of the ARAP model, and then describe how it is applied to individual energy terms.

3.2.1 As-rigid-as-possible (ARAP) energy.

At various stages in the formulation, D will take the form of an ARAP energy [Sorkine and Alexa 2007]. Let the pair of meshes to be compared be $\mathcal{V} = \{\mathbf{v}_i\}_{i=1}^n$ and $\mathcal{U} = \{\mathbf{u}_i\}_{i=1}^n$ and recall that all meshes have the same topology. Then the ARAP energy is a measure of how "different" the two meshes are. It is expressed in terms of per-vertex rotations R_i applied to each vertex of $\mathcal{U}$ in order to most closely align the one-ring of each vertex in $\mathcal{V}$ to the corresponding one-ring of $\mathcal{U}$, and measures the residual error of this alignment. The energy is zero if $\mathcal{U}$ and $\mathcal{V}$ are related by a rigid transformation, and increases as the transformation between the meshes becomes more nonrigid. Formally, it is defined as the minimization

$$E^{\text{ARAP}}(\mathcal{V},\mathcal{U};A) = \min_{R_1,\ldots,R_N} \sum_{i=1}^{N} \sum_{j\in\mathcal{N}(i)} \left\|(\mathbf{v}_i-\mathbf{v}_j) - AR_i\,(\mathbf{u}_i-\mathbf{u}_j)\right\|^2 \quad (11)$$

Each vertex has an associated 3×3 rotation R_i that are included in the optimization variables, parameterization using the exponential map as $R_i = \exp[\mathbf{r}_i]$ where $\exp[\cdot]$ converts from axis-angle form to a 3×3 matrix using the exponential map [Grassia 2000]. The parameter A is a 3×3 matrix which is the identity for standard ARAP, but will allow us to incorporate explicit global rotations and scales in our formulation. One convention we will need to introduce is a means of referring to the R_i outside the optimization. We can write a version of E^{ARAP} which names the rotation matrices, and omits the explicit minimization. The rotation optimization variables are then considered to be added to the global list of unknowns. If $\mathcal{R}$ is a set of rotations $\{R_i\}$ then write

$$\hat{E}^{\text{ARAP}}(\mathcal{V},\mathcal{U},\mathcal{R};A) = \sum_{i=1}^{N} \sum_{j\in\mathcal{N}(i)} \left\|(\mathbf{v}_i-\mathbf{v}_j) - AR_i\,(\mathbf{u}_i-\mathbf{u}_j)\right\|^2 \quad (12)$$

We now treat the rotations $\mathcal{R}$ as one of the free variables to be optimized together with the meshes, but this is equivalent, since $E = \min_{\mathcal{R}} \hat{E}$. Note that this does not imply any approximation or change to the minimization problem: for general f_i, we always have $\sum_{i=1}^n \min_{x_i} f_i(x_i) = \min_{x_1,\ldots,x_n} \sum_{i=1}^n f_i(x_i)$.

3.2.2 ARAP motion energy.

We enforce frame-to-frame consistency in an ARAP sense, defining the mesh deformation measure as an ARAP energy. Thus (4) becomes

$$E_{\textit{motion}}(\mathcal{V}_{1..T}) = \sum_{t=2}^{T} \min_{\lambda_t} E^{\text{ARAP}}(\mathcal{V}_{t-1},\mathcal{V}_t;\lambda_t I_{3\times 3}) \quad (13)$$

where the minimization over the λs means that scale change from frame to frame is not penalized (it is managed by the instance-core terms below).

3.2.3 ARAP core-template energy.

The core-template energy is also expressed as ARAP, but with an additional small-rotation penalty. Eq. (6) becomes

$$E_{\textit{core-template}}(\mathcal{V}_{\text{core}},\mathcal{V}_{\text{template}},\mathcal{R}_{\text{core}}) = \hat{E}^{\text{ARAP}}(\mathcal{V}_{\text{core}},\mathcal{V}_{\text{template}},\mathcal{R}_{\text{core}};I_{3\times 3}) \quad (14)$$

introducing variables $\mathcal{R}_{\text{core}} = \{R_i^{\text{core}}\}_{i=1}^N$. Also defining the function $\angle(R)$ as the rotation angle of the matrix R, we introduce a regularizer on the rotation magnitude, given by

$$E_{\textit{core-template-reg}}(\mathcal{R}_{\text{core}}) = \sum_{i=1}^{N} \angle(R_i^{\text{core}})^2 \quad (15)$$

3.2.4 "Hinted ARAP" instance-core energy.

The last of the deformation terms, from (5), penalizes the deformation from core to instance. We will again introduce explicit rotation variables, so (5) becomes

$$E_{\textit{instance-core}}(\mathcal{V}_{\text{core}},\mathcal{V}_{1..T},\mathcal{Q},\mathcal{G}) = \sum_{t=1}^{T} \hat{E}^{\text{ARAP}}(\mathcal{V}_{\text{core}},\mathcal{V}_t,\mathcal{Q}_t;\alpha_t^G R_t^G) \quad (16)$$

with new unknown rotation variables $\mathcal{Q}_t = \{Q_i^t\}_{i=1}^N$, gathered across all frames in the list $\mathcal{Q} = \{\mathcal{Q}_t\}_{t=1}^T$.

This is an ARAP energy, but will now undergo important modifications in order to constrain the range of motion of the animal. We find that allowing completely independent rotations for each vertex provides too many degrees of freedom for the inherent depth ambiguities of single-view fitting. In particular, we observe limbs that twist and splay out of the image plane, while still matching the silhouettes. In order to address this issue, we introduce a novel scheme for restricting limb motion, based on a lightweight, user-provided annotation of the mesh. The motion is separated into per-vertex motion, and global rotation, as described below.

When supplying an initial estimate of the core geometry, the user places the mesh vertices into M groups (Fig. 2). Each group m has a set of group parameters $\boldsymbol{\theta}_m$. Each vertex i is assigned to a group $m(i)$, the rotation Q_i^t at frame t for this vertex is parameterized by a lower-dimensional parameter vector $\boldsymbol{\gamma}_{i,t} \in \mathbb{R}^{d_m}$.

The dimensionality $d_m \in \{0,1,3\}$ controls how freely the vertices in group m may deform. Three different group types are defined, depending on the dimensionality for that group:

• rigid ($d_m = 0$): Parts of the model which move near-rigidly, such as an animal's skull. The group parameters comprise a rotation per frame, $\boldsymbol{\theta}_m = \{\boldsymbol{\omega}_m^1,\ldots,\boldsymbol{\omega}_m^T\} \subset \mathbb{R}^3$, and each vertex rotation is simply a copy of the group rotation $Q_i^t = \exp[\boldsymbol{\omega}_{m(i)}^t]$.

• nonrigid ($d_m = 3$): General non-rigid motion. Each vertex's rotation is unconstrained, so $Q_i^t = \exp[\boldsymbol{\gamma}_{i,t}]$, and the group parameters are empty $\boldsymbol{\theta}_m = \{\}$. This is the normal ARAP case.

• hinge ($d_m = 1$): Components, such as animals' legs, which comprise mostly rigid components linked by coaxial hinges. In this case, the group parameters are a single global (time-independent) rotation axis, $\boldsymbol{\theta}_m = \mathbf{a}_m \in \mathbb{R}^3$, and each vertex is parameterized by a single scalar $\boldsymbol{\gamma}_{i,t} \in \mathbb{R}$, so $Q_i^t = \exp[\boldsymbol{\gamma}_{i,t}\mathbf{a}_{m(i)}]$. To see why this works, consider a mesh which comprises two rigid parts linked by a hinge joint. Then if $\mathcal{U}$ and $\mathcal{V}$ are two instances of this mesh with the hinge in different positions, the minimum of the ARAP energy will be found when all vertices, both at the hinge and elsewhere, are associated with rotations sharing the hinge's axis. Each vertex

in the group therefore rotates about a common axis, but with varying angle. Note that this combats splaying within the legs, but does not inhibit correct splaying at the hips, for example when the dog turns in Fig. 1.

• pivot ($d_m = 2$): For completeness although we have not used them, pivot components (named after pivot joints [Grassia 2000, page 64]), have two degrees of freedom ($\boldsymbol{\gamma}_{i,t} \in \mathbb{R}^2$). The group parameters are $\boldsymbol{\theta}_m = \{\mathbf{a}_m^1, \mathbf{a}_m^2\}$, and the rotations are given by $Q_i^t = \exp[\boldsymbol{\gamma}_{i,t,1}\mathbf{a}_{m(i)}^1]\exp[\boldsymbol{\gamma}_{i,t,2}\mathbf{a}_{m(i)}^2]$.

3.2.5 Instance-core motion energy

Having access to the ARAP rotations $\mathcal{Q}$ in the optimization, we define an additional motion prior, as follows.

$$E_{\textit{instance-core-motion}}(\mathcal{Q}) = \sum_{t=2}^{T}\sum_{i=1}^{N} \angle(Q_i^t (Q_i^{t-1})^\top)^2 \qquad (17)$$

This prior is typically used with a very low weight, but is useful to stabilize the optimization.

3.2.6 Global per-frame rigid/camera transformation

We also incorporate a per-frame rotation matrix and per-frame scaling to account for changes in camera zoom and object distance. These quantities are parametrized by rotations R_t^G and scales α_t^G, gathered into a list of **pose parameters**

$$\mathcal{G} = \{\alpha_t^G, R_t^G\}_{t=1}^T \qquad (18)$$

Including global pose parameters is necessary for the hinted ARAP constraints on the body groups (§3.2.4) to be expressed in the core (local) coordinates. For example, the axis for leg rotations should always be transverse to the knees, even if the animal has turned relative to the camera.

In some cases, we also restrict the global rotation R_t^G, so it is parameterized by a parameter vector γ_t^G with fewer than 3 degrees of freedom. In cases where the animal moves in a straight path without turning, global rotations are around a single user-specified "pitch" axis, so γ_t^G is 1-dimensional. When the animal does turn, we often restrict the global rotation to two axes including yaw (heading) and pitch, but excluding roll. Details on specific examples are given in the results (§5).

3.2.7 Smooth rigid/camera motion.

The velocity of the rigid/camera coordinates is penalized by

$$E_{\textit{camera-R-motion}}(\mathcal{G}) = \sum_{t=2}^{T} \angle(R_t^G (R_{t-1}^G)^\top)^2 \qquad (19)$$

Camera scale velocity is penalized by

$$E_{\textit{camera-scale-motion}}(\mathcal{G}) = \sum_{t=2}^{T} \left(\log(\alpha_t^G) - \log(\alpha_{t-1}^G)\right)^2 \qquad (20)$$

The log domain is used in order to prevent very small scales.

4 Optimization Algorithm

The above energy function is a large-scale non-convex optimization problem, and so both the choice of optimizer and initialization are important. The optimization alternates between different subsets of the unknowns. Unless stated otherwise, each optimization step uses a large-scale Levenberg-Marquardt algorithm. All gradients are computed semi-automatically through symbolic differentiation [Paprocki 2011] with manual fixup. Note that many terms (e.g., w_j and $\mathbf{N}_j$ in Eq. 8) have complex dependencies on the unknowns, but liberal use of the chain rule keeps the code compact with only a small impact on efficiency. Rotations are parameterized using the exponential map [Grassia 2000].

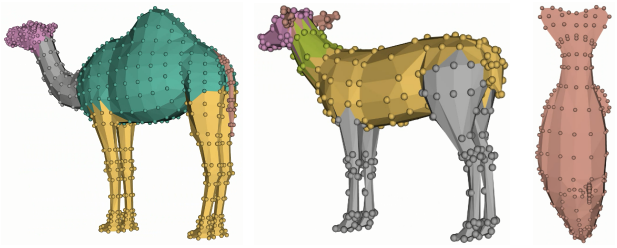

Figure 6: ***Templates for camel, impala, and fish.*** *Motion domains for the camel and impala are as for the dog, but the camel's neck is of type hinge. The fish is all of type nonrigid.*

As there are a large number of variables in the optimization (about 500,000 for the longest sequence) and some of our test sequences approach 200 frames, we have implemented the optimization in "sweeps," where each frame is visited sequentially, and only the energy terms relating to that frame are optimized.

The shape in each frame is initialized as follows. The core geometry $\mathcal{V}_{\text{core}}$ is initialized to the user-provided mesh $\mathcal{V}_{\text{template}}$. For the first sweep, an important task is to ensure reasonable initialization of the preimages $\mathcal{W}$. To this end, several weights are zeroed ($w_{\textit{sil}} = w_{\textit{nrm}} = w_{\textit{sil-smooth}} = 0$) and the core is fixed ($w_{\textit{core-template-reg}} = w_{\textit{core-template}} = \infty$). Then a sequence of optimizations is performed with T increasing from 1, varying only $\mathcal{V}_T$ at each step.

The algorithm then resets the weights to their default values, and each sweep alternates the following updates:

1. Optimize the core geometry $\mathcal{V}_{\text{core}}$ and all local and global pose variables $(\mathcal{G}, \mathcal{R}_{\text{core}}, \mathcal{Q}_t)$, holding all other variables fixed.
2. Optimize all preimage coordinates $\mathcal{W}_t$, holding the other variables fixed, using dynamic programming [Cashman and Fitzgibbon 2013].
3. Jointly optimize instance geometry $(\mathcal{V}_t)$, scale (α_t^G), and preimages $(\mathcal{W}_t)$, holding other variables fixed. In our current implementation $w_{\textit{sil-smooth}}$ is zeroed for this step, due to the expense of recomputing geodesic distances which yields some jitter in the contour generators. It would be interesting future work to look at approximations to ameliorate this.

We typically run for 20 sweeps, although the instance variables are generally close to their final values after 5 sweeps.

[AWF: Note that as many of our energy terms are quadratic, there are "closed form" solutions to some of the subproblems. However, it is almost certainly not worthwhile to exploit those: running the Levenberg Marquardt algorithm on a quadratic problem will not be more than say 2x slower than a quadratic solver, and running LM on a problem which admits quadratic alternations will almost certainly be quicker [?].]

5 Results

We demonstrate our method on video sequences of three mammals (dog, camel, impala) and a fish (sea bass), all obtained from the stock photography site shutterstock.com. Template meshes (Fig. 6)

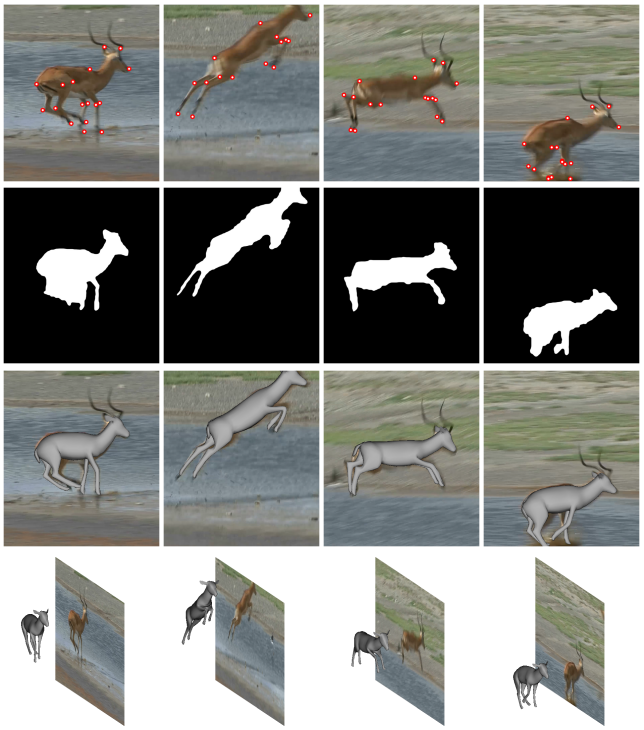

Figure 7: ***Leaping Impala.*** *Four frames from a 100-frame sequence.* ***Top:*** *input frames with 15 point tracks.* ***Row 2:*** *Silhouette masks. Note that these can be imperfect.* ***Row 3:*** *Overlaid 3D reconstruction.* ***Bottom:*** *Oblique view.*

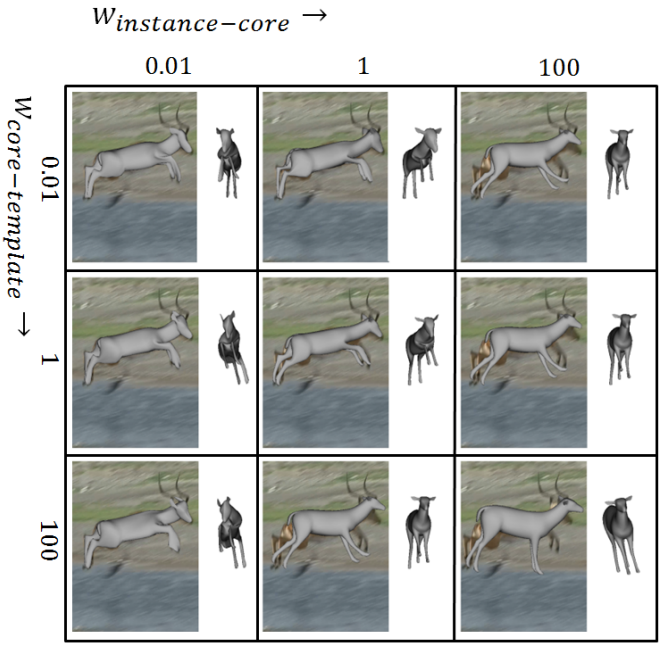

Figure 8: ***Effect of*** $w_{\text{instance-core}}$ ***and*** $w_{\text{core-template}}$ ***parameters.*** *The parameters are varied through four orders of magnitude to investigate the effect of extreme settings. Left column: too small instance-core weight causes collapse along the viewing direction.*

were obtained by downloading appropriate-looking meshes from turbosquid.com, except for the dog, where we deliberately chose an ill-fitting mesh in order to show that the output is not strongly tied to the template.

All runs used the weights $w_{\textit{core-template-reg}} = 8192, w_{\textit{motion}} = 0.5, w_{\textit{point}} = 2$, and all used customized camera motion weights: high for the impala and camel, where the camera is fairly static, low for the dog and fish. Then the impala, fish, and camel shared all but one weight: $w_{\textit{sil-smooth}}$ which was 4 for the fish, 10 for the others. The dog has a different weight vector because it is the only model for which $w_{\textit{instance-core-motion}}$ is nonzero. Computation times were all of the order of 5-20 minutes per sweep, with sequences requiring from 5-20 sweeps.

Visualizations and results are shown in the accompanying video, and in Figures 1, 7, 10. As can be seen in the examples, the method produces models which match the silhouette reasonably closely (although errors of up to 10 pixels do occur), and produces smooth meshes. The video reveals that the mesh motion is mostly smooth, although the first order motion prior does allow some jitter. It also has some difficulty correctly interpolating the motion of the dog's head when it is completely occluded by the body. These effects might be repaired by a second order motion prior, but that is future work.

To investigate the behaviour of the energy weights, we varied two important parameters, $w_{\textit{instance-core}}$ and $w_{\textit{core-template}}$, through four orders of magnitude. A high value of the core-template weight corresponds to ARAP fitting of the template to the sequence, and yields either overly rigid reconstructions, or allows too much variation in the instances. Conversely, moderate values yield higher fidelity reconstructions, evidence of the value of the template-core-instance formulation. Note that in this case, the impala template is visually very close to the subject's shape, but is different in proportion, so adapting the core still gives a more realistic model.

In order to test the benefit of ARAP as a deformation model for vision, we show comparisons with linear basis shapes [Bregler et al. 2000; Cashman and Fitzgibbon 2013]. To avoid implementing a completely different system, we simply fitted to our best impala model using PCA, and projected the recovered geometry onto the PCA basis. As might be expected, large motions are poorly represented under the PCA model, as the side-by-side comparison in Fig. 9 illustrates.

6 Discussion, Limitations, and Future Work

This paper introduced a method for reconstructing time-varying 3D geometry of animal movements. While much previous work in computer graphics has focused on the use of specialized capture equipment or studios, we instead work entirely from monocular video; all our examples come from stock footage on the web.

A core technical contribution of our work is the use of ARAP as a prior for articulated, non-rigid reconstruction. We show that ARAP can effectively model articulation and non-rigid reconstruction. An alternative, that we considered, would be to directly estimate a skinned, skeleton-based rig from video data. We chose the ARAP representation because we believe that estimating a skeleton from data would be much more difficult, requiring a combinatorial optimization, many local minima, and difficult-to-specify user constraints. In contrast, the ARAP formulation works with a straightforward optimization and relatively few user constraints. We believe that it should be possible to learn many of the variables that are currently user-specified, such as the mesh clustering. However, both ARAP and skeleton-based estimation are interesting avenues for future work.

Although our method requires user input, there is no provision

Figure 9: ***Comparison with linear basis shapes.*** *Each pair: left is our method, right is PCA of our solution. The range of motion of the impala means that even with eight PCA bases, as recovered by [Cashman and Fitzgibbon 2013], significant errors remain in the linear model.*

Figure 10: ***Swimming sea bass.*** *Frames from a 114-frame sequence. Seven point tracks were used. The sharp tail fin is recovered without any special modelling.*

made for interactive cleanup of the model: the user is assumed to be able to produce reasonably accurate silhouettes and point tracks. However, for many sequences, one might imagine a user interface to adjust the model, and fast minimization to the new optimum would be necessary.

Because we use silhouettes as our primary constraint, our method does not estimate detailed shape, and often exhibits errors at internal occluding contours. Moreover, we do not perfectly fit all silhouettes. Estimating detailed shape would likely require fitting texture and/or lighting, a challenging problem for future work.

References

AKHTER, I., SHEIKH, Y., KHAN, S., AND KANADE, T. 2010. Trajectory Space: A Dual Representation for Nonrigid Structure from Motion. *IEEE Trans. PAMI*.

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-Rigid-As-Possible Shape Interpolation. In *Proc. SIGGRAPH*.

BAI, X., WANG, J., SIMONS, D., AND SAPIRO, G. 2009. Video SnapCut: Robust Video Object Cutout Using Localized Classifiers. *ACM Trans. Graphics*.

BALAN, A. O., SIGAL, L., BLACK, M. J., DAVIS, J. E., AND HAUSSECKER, H. W. 2007. Detailed human shape and pose from images. In *Proc. CVPR*.

BLANZ, V., BASSO, C., POGGIO, T., AND VETTER, T. 2003. Reanimating Faces in Images and Video. In *Proc. Eurographics*.

BREGLER, C., HERTZMANN, A., AND BIERMANN, H. 2000. Recovering Non-Rigid 3D Shape from Image Streams. In *Proc. CVPR*, 690–696.

CASHMAN, T. J., AND FITZGIBBON, A. W. 2013. What Shape are Dolphins? Building 3D Morphable Models from 2D Images. *IEEE Trans. PAMI 35*, 1.

DE AGUIAR, E., STOLL, C., THEOBALT, C., AHMED, N., SEIDEL, H.-P., AND THRUN, S. 2008. Performance Capture from Sparse Multi-view Video. *ACM Trans. Graphics 27*, 3.

DE LA GORCE, M., FLEET, D. J., AND PARAGIOS, N. 2011. Model-Based 3D Hand Pose Estimation from Monocular Video. *IEEE Trans. PAMI 33*, 9.

GALL, J., STOLL, C., DE AGUIAR, E., THEOBALT, C., ROSENHAHN, B., AND SEIDEL, H.-P. 2009. Motion capture using joint skeleton tracking and surface estimation. In *Proc. CVPR*.

GIRVEAU, B. 2007. *Once Upon a Time—Walt Disney: The Sources of Inspiration for the Disney Studios*. Prestel.

GRASSIA, F. S. 2000. *Believable Automatically Synthesized Motion by Knowledge-Enhanced Motion Transformation*. PhD thesis, Carnegie Mellon University CMU-CS-00-163.

PAPROCKI, M. 2011. Symbolic mathematics in pure python. In *Proc. SIAM Comp. Sci. and Eng, Reno, NV*.

PONS-MOLL, G., LEAL-TAIXÉ, L., TRUONG, T., AND ROSENHAHN, B. 2011. Efficient and robust shape matching for model based human motion capture. In *Proc. DAGM*.

ROSS, D., TARLOW, D., AND ZEMEL, R. S. 2010. Learning Articulated Structure and Motion. *IJCV 88*, 2.

SALZMANN, M., AND URTASUN, R. 2010. Combining discriminative and generative methods for 3D deformable surface and articulated pose reconstruction. In *Proc. CVPR*.

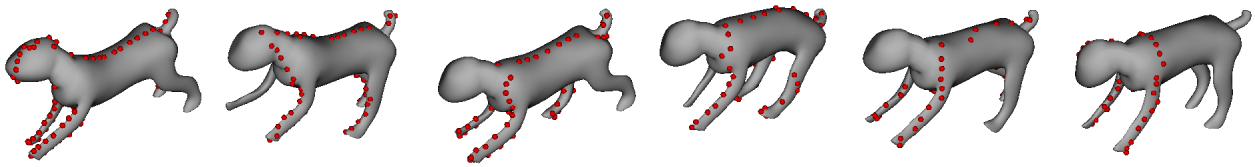

Figure 11: ***Contour generator.*** *The recovered preimage points $\mathcal{W}_t$ at several frames of the dog sequence. The contour generator moves across the mesh, indicating which parts of the shape are constrained by the data.*

SHIH, L.-Y., CHEN, B.-Y., AND WU, J.-L. 2009. Video-based motion capturing for skeleton-based 3d models. In *Proc. Pacific-Rim Symposium on Image and Video Tech. (PSIVT)*, 748–758.

SORKINE, O., AND ALEXA, M. 2007. As-Rigid-As-Possible Surface Modeling. In *Proc. SGP*.

TAYLOR, J., JEPSON, A. D., AND KUTULAKOS, K. N. 2010. Non-Rigid Structure from Locally-Rigid Motion. In *Proc. CVPR*.

ULLMAN, S. 1984. Maximizing rigidity: the incremental recovery of 3-D structure from rigid and nonrigid motion. *Perception 13*, 3, 255–274.

VLASIC, D., BRAND, M., PFISTER, H., AND POPOVIĆ, J. 2005. Face transfer with multilinear models. *ACM Trans. Graphics*.

VLASIC, D., BARAN, I., MATUSIK, W., AND POPOVIĆ, J. 2008. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graphics*.

YAN, J., AND POLLEFEYS, M. 2008. A Factorization-Based Approach for Articulated Nonrigid Shape, Motion and Kinematic Chain Recovery from Video. *IEEE Trans. PAMI 30*, 5.