Unwrap Mosaics: A new representation for video editing[*]

Alex Rav-Acha
Weizmann Institute of Science

Pushmeet Kohli
Microsoft Research

Carsten Rother
Microsoft Research

Andrew Fitzgibbon
Microsoft Research

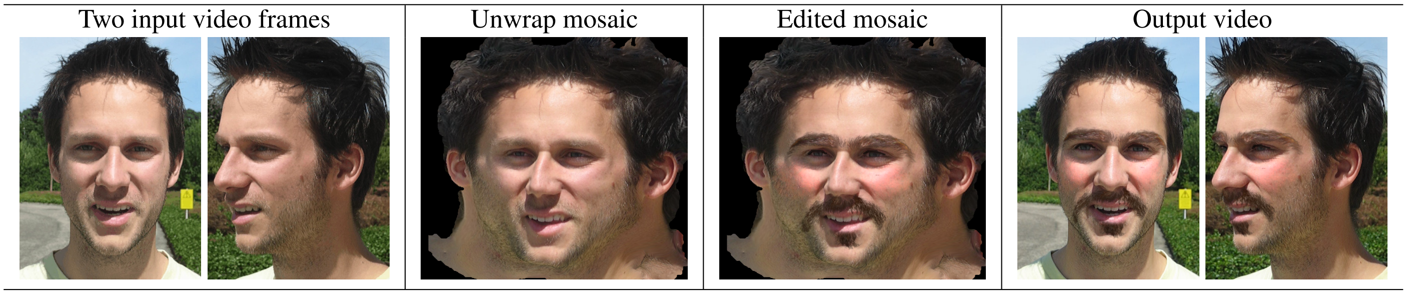

Figure 1: *Editing a video of a talking head. The input images are automatically converted to an "unwrap mosaic" without an intervening 3D reconstruction. Painting the mosaic and re-rendering the video allows us to add virtual make-up (eyebrows, moustache, and rouge on the cheeks) to the actor, as if to a texture map on a deformable 3D surface.*

Abstract

We introduce a new representation for video which facilitates a number of common editing tasks. The representation has some of the power of a full reconstruction of 3D surface models from video, but is designed to be easy to recover from *a priori* unseen and uncalibrated footage. By modelling the image-formation process as a 2D-to-2D transformation from an object's texture map to the image, modulated by an object-space occlusion mask, we can recover a representation which we term the "unwrap mosaic". Many editing operations can be performed on the unwrap mosaic, and then re-composited into the original sequence, for example resizing objects, repainting textures, copying/cutting/pasting objects, and attaching effects layers to deforming objects.

Keywords: motion estimation, video editing, layers, mosaicing

1 Introduction

We show how to recover dense models of deforming 3D surfaces from previously unseen video footage. These models are not conventional 3D surface models—in one sense they are a generalization—but they have some very desirable properties. First, they allow a wide range of editing operations to be performed on the original video. Second, they are easier to recover from the video than 3D models, allowing us to work with deforming surfaces and self-occlusion, two bugbears of conventional computer vision based approaches. Finally, the model itself provides a reference for feature-point tracking, yielding long tracks which are resistant to drift and occlusion, and which could serve as a substrate for many conventional editing techniques.

We are given a video sequence, captured in the real world. Our model of the world is that it is a collection of deforming 3D surfaces, viewed by a moving camera. If we had access to the 3D models and their texture maps, edits such as resizing objects, repainting textures, copying/cutting/pasting objects, magnifying motion, attaching effects layers to deforming objects, and so on, would be easy. However, such models are not readily obtained from *a priori* unseen and uncalibrated footage: the state of the art in the recovery of 3D information from images is briefly as follows. The extraction of *sparse* 3D information from image sequences of rigid scenes is by now well understood, with software packages available which can recover 3D camera trajectories and sparse 3D point clouds from uncalibrated image sequences [2d3 Ltd. 2008; Thormählen and Broszio 2008]. The mathematical extensions to *nonrigid* scenes are understood [Bregler et al. 2000; Brand 2001; Torresani et al. 2008] but their estimation is somewhat less reliable under occlusion. For *dense* reconstruction from video, however, we are firmly restricted to rigid scenes [Seitz et al. 2006]. Some classes of dense models can be built using interactive tools [Debevec et al. 1996; van den Hengel et al. 2007], which aid the recovery of polyhedral surface models from video, but again, these are restricted to rigid scenes. Triangulation of the sparse points from nonrigid structure may be expected to be at least as difficult as from rigid structure, which has proved surprisingly troublesome: although it is easy to form a triangulation of 2D projections of 3D points, zippering these models has not been the simple extension of [Turk and Levoy 1994] that might have been expected.

We introduce a technique which overcomes these difficulties to a large extent, generating a representation which is in some ways equivalent to a deforming 3D surface model, but can be extracted directly from video. To compress the idea into a simple slogan, our primary goal is *to recover the object's texture map, rather than its 3D shape.* Accompanying the recovered texture map will be a 2D-to-2D mapping describing the texture map's projection to the images, and a sequence of binary masks modelling occlusion. The combination of texture map, 2D–2D mapping, and occlusion masks is what we call the *unwrap mosaic*. A video will typically be represented by an assembly of several unwrap mosaics: one per object, and one for the background. Once one has the unwrap mosaic for an object, it is possible in principle to recover the 3D shape, but for many editing tasks this is not necessary: edits can be performed on the mosaic itself and re-rendered without ever converting to a 3D representation.

[*] http://research.microsoft.com/unwrap

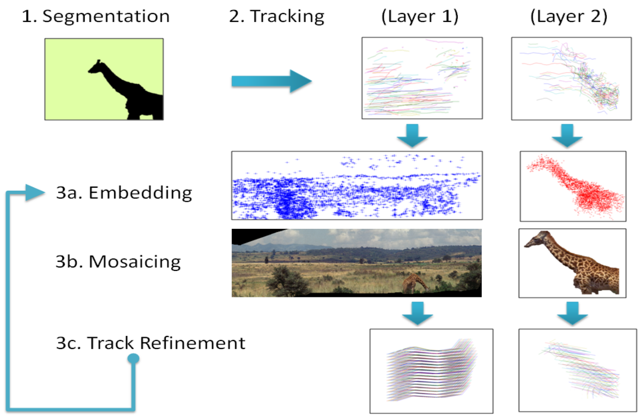

Figure 2: Reconstruction overview. *Steps in constructing an unwrap mosaic representation of a video sequence. Steps 1 to 3a form an initial estimate of the model parameters, and step 3 is an energy minimization procedure which refines the model to subpixel accuracy.*

The main contribution of this paper is the algorithm to recover the unwrap mosaic from images. This algorithm is an energy minimization procedure, and the formal exposition beginning in section 2 shows how each of the algorithmic steps derives naturally from the fitting of the model to image sequences.

The formal description, even in a simplified form, is long. Before embarking on it, it may prove valuable to outline at a high level the steps of the algorithm that will emerge. Despite not having yet set up our notation, it is hoped that the steps will be comprehensible to readers familiar with the computer vision and graphics literature. These steps all correspond to minimization of the model energy via coordinate descent, with different variables held constant at each step.

1. Segmentation

The first stage is to segment the sequence into independently moving objects. This is a subject that has seen recent research, and we use a variant of "video cut and paste" [Li et al. 2005] which allows accurate segmentation maps to be obtained with relatively little effort. Although our system can in some cases compute the segmentation maps automatically, a general-purpose implementation must allow for user interaction at this and certain later stages. These interactions are discussed in section 4.

2. Tracking

We recall that we are trying to recover the texturemap of a deforming 3D object from a sequence of 2D images. The fundamental assumption is that although the model is changing its shape, the texture map may be assumed to be constant, as on skin and cloth. Consider a point on the object (e.g. a surface marking on skin). As this point moves through the video, it generates a 2D *trajectory* or *track*. Conversely, standard computer vision algorithms for interest-point detection and tracking [Sand and Teller 2006, for example] can be used to compute such tracks from the input sequence.

3a. Embedding

The computational cornerstone of our method is the embedding step. We view the sparse point tracks as a high-dimensional projection of the 2D surface parameters, so the point (u, v) in parameter space generates a vector of image positions $(x_1, y_1, x_2, y_2, ..., x_T, y_T)$. We can recover (up to reparametrization) the surface's (u, v) parameter space by computing an *embedding* of the point tracks into 2D, directly yielding (u, v) coordinates for each track. Analogous embeddings (e.g. multidimensional scaling and locally linear embedding) are widely used in visualization, and were used by Zigelman et al. [2002] to create texture coordinates for 3D models. Here there is no 3D model, but a related line of reasoning leads to an algorithm which chooses (u, v) coordinates for each trajectory such that distances in uv space are commensurate with distances in the image frames at which pairs of tracks are maximally separated.

3b. Mosaic stitching

The embedding defines a map from the tracked points in each image to the (u, v) parameter space. Interpolating this mapping allows each image to be warped into the common frame. A variation of the standard *mosaic stitching* technique [Agarwala et al. 2004] emerges naturally from the energy formulation. This is the second crucial stage, and has the very useful property that it often chooses a good texture map, even when some per-frame mappings have large errors.

3c. Track refinement

After the above three steps, the mosaic, although not accurate, is generally good enough to create a reference template to match against the original frames. Because this matching is to a single reference, it reduces any drift that may have been present after the original tracking phase. In essence we are in the situation described in [Gay-Bellile et al. 2007], but where their texture map is provided *a priori*, ours is built automatically from the sequence. Regularization of the mapping defines a dense interpolation, so that track information propagates to occluded areas of the scene, giving a complete description of the object motion.

3d. Iterate

Because track refinement has improved the estimate of inter-track distances, we can hope to compute a better embedding and mosaic by iterating steps 3a-c. Because, as shall be seen, each stage minimizes the same energy function, a consistent global convergence measure can be defined, and iteration can be terminated easily.

Using the model for video editing

Given the unwrap mosaic, one may proceed for editing as if one had a planar mosaic [Wang and Adelson 1994; Irani et al. 1995] with additional occlusion masks. The simplest task is to edit the texture map, for example by drawing on it, warp it via the recovered mapping, and combine with the other layers of the original sequence. This has the effect of drawing on the object's surface, and the accuracy of registration with the pre-existing surface texture is a good test of the algorithm. In practice, the re-rendered mosaic will not exactly match the original sequence, so it is better to represent the edit as an overlay on the texture map, which is warped by the 2D–2D mapping, masked by the occlusion masks, and alpha-blended with the original image. Another possible edit is to remove layers: the removed area is filled in because the mapping is defined even in occluded areas.

Limitations

The main limitations of our approach are that it requires quite well-textured objects in order to work, and that the objects to be recovered must be smooth 3D surfaces which are deforming smoothly over time. In detail:

- Textured surfaces are required for point tracking. Low texture, such as on a face, requires good focus and lighting. One-dimensional textures (stripes, rather than spots) suffer from the aperture problem. In addition, motion blur reduces the efficacy of the point tracking.
- The assumption of a smoothly varying smooth 3D surface means that objects with significant protrusions (the dinosaur in figure 11) do not yield useful mosaics.

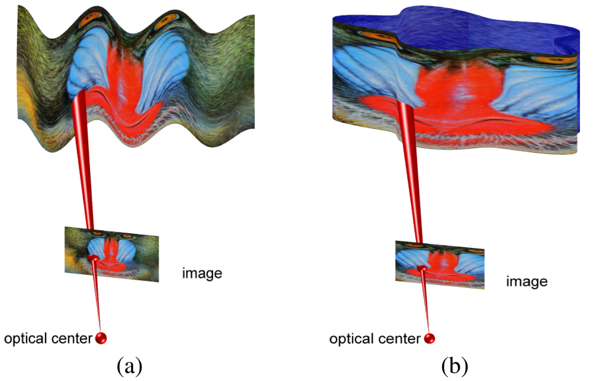

Figure 3: 3D Projection with and without hidden surface points. *A 3D surface* $\mathbf{S}(\mathbf{u})$ *projects to a 2D image* $\mathbf{I}(\mathbf{x})$*. (a) Without self-occlusion, the mapping from points* $\mathbf{u}$ *on the model texture map* $\mathbf{C}(\mathbf{u})$ *to the image is a simple 2D–2D mapping* $\mathbf{w}(\mathbf{u})$*. The red cone represents the point-spread function of the pixel marked by the red sphere: the colour* $\mathbf{I}(\mathbf{x})$ *is a function of the colours of all world points which fall within the cone. (b) With self-occlusion, we view hidden-surface removal as defining a binary mask* $b(\mathbf{u})$*, aligned with the surface parameter space. In this illustration, blue tinted pixels on the model are invisible, corresponding to* $b(\mathbf{u}) = 0$*.*

- The assumption of smoothly varying lighting means that strong shadows will disrupt tracking. Although this can be somewhat mitigated by high-pass filtering as a preprocess, it would be better dealt with explicitly.
- We are currently limited to disc-topology objects, meaning that a rotating cylinder will be reconstructed as a long tapestry, rather than a mosaic with cylindrical topology (see figure 11).

2 The unwrap mosaic model

The above algorithmic steps arise naturally by introducing a new model of image formation, and fitting that model to the image sequence. The next section formally describes the model, and shows how the fitting algorithm corresponds to a sequence of energy minimization processes.

Derivation of the algorithm will comprise three stages: image generation model, energy formulation, and minimization. First, the **image generation model** defines how an image sequence is constructed from a collection of unwrap mosaics. The model is introduced in the continuous domain, and then its discrete analogue is derived. Given such a model, extracting it from a supplied image sequence becomes a fitting problem, albeit a nonlinear and under-constrained one. The bulk of the paper deals with this fitting problem. It is expressed as a problem of **energy minimization**, which can be implemented via nonlinear optimization. The key to the unwrap mosaic model we develop is that a good **initial estimate** for the 2D-2D mapping, and hence for the texture map, can be obtained from sparse 2D tracking data.

In order to explain the model and its recovery from video, the next several paragraphs will assume a greatly simplified scenario. Although this ignores many complexities, such as model topology and lighting, it will be enough to motivate the technique, and later sections will describe the more complex model. Many topics are dealt with in more detail in [Rav-Acha et al. 2008].

Despite the difficulties alluded to in the introduction of recovering 3D models from video, let us proceed as if to solve the general 3D surface reconstruction problem. Let the world be a collection of 3D surfaces, represented as geometry images [Gu et al. 2002]; that is to say each surface is represented as a function $\mathbf{S} : \mathcal{Q} \mapsto \mathbb{R}^3$ where $\mathcal{Q} \subset \mathbb{R}^2$ is the unit square. The surface is also accompanied by a texture map $\mathbf{C} : \mathcal{Q} \mapsto \mathbb{C}$, where $\mathbb{C}$ represents the color space in use, e.g. RGB. Thus, each point $\mathbf{u} = (u, v) \in \mathcal{Q}$ is associated with the 3D point $\mathbf{S}(u, v)$ and the color $\mathbf{C}(u, v)$.

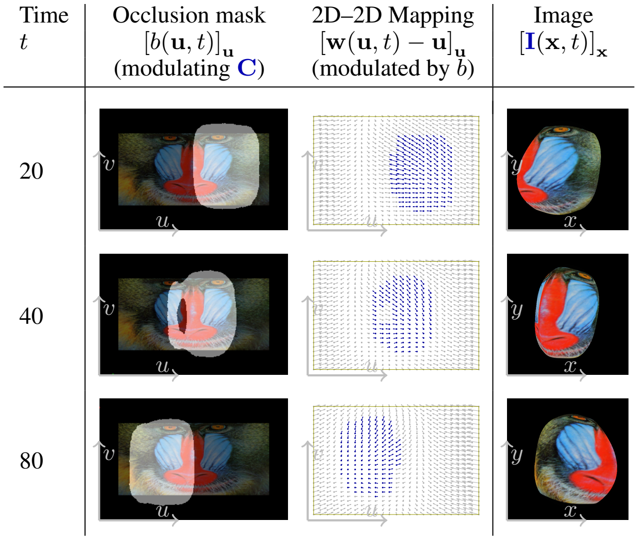
Time
t
Occlusion mask
$[b(\mathbf{u}, t)]_{\mathbf{u}}$
(modulating $\mathbf{C}$)
2D–2D Mapping
$[\mathbf{w}(\mathbf{u}, t) - \mathbf{u}]_{\mathbf{u}}$
(modulated by b)
Image
$[\mathbf{I}(\mathbf{x}, t)]_{\mathbf{x}}$
20
40
80
v
u
y
x

Figure 4: The image generation model. *Three frames from a synthetic sequence of a deforming 3D model. The columns show the evolution of the binary visibility map* $b(\mathbf{u}, t)$ *(in white) and the 2D–2D mapping* $\mathbf{w}(\mathbf{u}, t)$*. Each row may be read as: "the area of* $\mathbf{C}$ *which is visible is warped by* $\mathbf{w}$ *to give* $\mathbf{I}$*". To aid visualization, the visibility map is shown modulating the texture map* $\mathbf{C}(\mathbf{u})$ *(which is constant throughout the sequence). The mapping* $\mathbf{w}$ *is represented by arrows showing how the texture map is warped to produce each image. Although only the values of* $\mathbf{w}$ *where* b *is nonzero are used in order to render each frame, we emphasize that* $\mathbf{w}$ *is defined for* all *values of* $\mathbf{u}$ *and* t*, and it is shown in gray for occluded surface points. When reconstructing* $\mathbf{w}$ *from an image sequence, it will be our goal to recover it at both visible and invisible points.*

A generalized camera is a function $\boldsymbol{\pi} : \mathbb{R}^3 \mapsto \mathbb{R}^2$, which induces the 2D-to-2D mapping $\mathbf{w}(\mathbf{u}) = \boldsymbol{\pi}(\mathbf{S}(\mathbf{u}))$. The function $\mathbf{w}$ now maps from the unit square in the surface's parameter space to the image domain. Let us assume for the moment a trivial lighting model where texture-map colors $\mathbf{C}$ are simply copied to the image plane. Let the point-spread function (PSF) of the imaging camera [Seetzen et al. 2004] be the function $\rho : \mathbb{R}^2 \mapsto \mathbb{R}$.

If every point on the surface were visible, as in fig 3a, the rendered image $\mathbf{I}(\mathbf{x})$ would be defined by

$$\mathbf{I}(\mathbf{x}) = \int \rho(\mathbf{w}(\mathbf{u}) - \mathbf{x})\mathbf{C}(\mathbf{u})\mathrm{J}(\mathbf{u})d\mathbf{u} \qquad (1)$$

where $\mathrm{J}(\mathbf{u})$ is the determinant of the mapping Jacobian.

In practice, of course, parts of the surface are backfacing, or hidden by other surfaces, or self-occluded by the surface itself. We encapsulate all of these processes into an object-space visibility map $b(\mathbf{u})$, defined as

$$b(\mathbf{u}) = \begin{cases} 1 & \text{if } \mathbf{S}(\mathbf{u}) \text{ is visible} \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

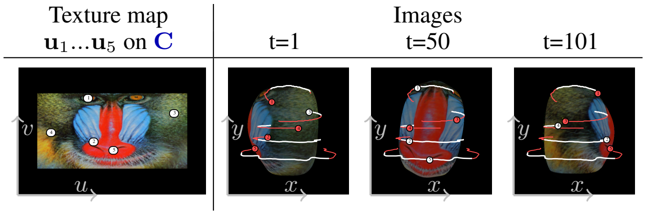

Figure 5: Trajectories. *The unwrap mosaic may also be viewed as defining the* trajectory*, or motion path, associated with each texture coordinate. For some example points $\mathbf{u}_k$, we show the corresponding trajectories $[\mathbf{w}(\mathbf{u}_k, t)]_t$. These are modulated by the point's visibility function $[b(\mathbf{u}_k, t)]_t$, so that white portions of the track are visible ($b = 1$), and red portions are occluded by another part of the object ($b = 0$).*

yielding the complete imaging function

$$\mathbf{I}(\mathbf{x}) = \int \rho(\mathbf{w}(\mathbf{u}) - \mathbf{x})\mathbf{C}(\mathbf{u})b(\mathbf{u})\mathrm{J}(\mathbf{u})\mathrm{d}\mathbf{u} \qquad (3)$$

This is a key step in deriving the new model: the link between b and the 3D geometry is relaxed. Essentially we will not enforce geometrically correct hidden-surface relationships, thus simplifying the recovery of b from images, but the recovered b will nonetheless usefully encapsulate the occlusion geometry. When recovering the model from video, a Markov random field prior on b will replace 3D occlusion reasoning to constrain and regularize the model. Conversely, fitting the model can return a $\mathbf{w}$ and b which are not consistent with any 3D geometry, but if the sequence is correctly re-rendered by the returned parameters, many editing tasks will not suffer.

The first extension of the model is to sequences. The object's colors remain the same for each frame of video, but the mapping $\mathbf{w}$ and the visibility map b will change. Thus a video sequence of T frames $\{\mathbf{I}(\mathbf{x}, t)\}_{t=1}^{T}$ is defined by

$$\mathbf{I}(\mathbf{x}, t) = \int \rho(\mathbf{w}(\mathbf{u}, t) - \mathbf{x})\mathbf{C}(\mathbf{u})b(\mathbf{u}, t)\mathrm{J}(\mathbf{u}, t)\mathrm{d}\mathbf{u} \qquad (4)$$

Now the *trajectory* of the point at parameter-space position $\mathbf{u}$ is the sequence of 2D points $\mathbf{w}(\mathbf{u}, :) = [\mathbf{w}(\mathbf{u}, 1); \mathbf{w}(\mathbf{u}, 2); \ldots; \mathbf{w}(\mathbf{u}, T)]$. Figure 5 illustrates the ground-truth trajectories of the synthetic sequence, while figure 7 shows the sparse trajectories acquired by tracking on the input images.

The final modification is to deal with multiple objects in the sequence. Let the number of surfaces (including the background) be L, with each object represented by the tuple of functions $(\mathbf{C}^l, \mathbf{w}^l, b^l)$. Image formation then becomes

$$\mathbf{I}(\mathbf{x}, t) = \sum_{l=1}^{L} \int \rho(\mathbf{w}^l(\mathbf{u}, t) - \mathbf{x})\mathbf{C}^l(\mathbf{u})b^l(\mathbf{u}, t)\mathrm{J}^l(\mathbf{u}, t)\mathrm{d}\mathbf{u} \qquad (5)$$

where the visibility masks b are now encoding inter-object occlusions as well as self-occlusions.

Figure 4 illustrates the model using a synthetic sequence of a deforming 3D object.

2.1 Discrete energy formulation

The above description is in terms of continous functions $\mathbf{C}, b, \mathbf{w}$ and $\mathbf{I}$. In computation we adopt a discretization onto a regular grid. The images $\mathbf{I}(\mathbf{x}, t)$ are received in discrete form, as grids of size $W \times H$. We choose a discretization of the parameter space $\mathcal{Q}$ into a $w \times h$ grid, where w and h are chosen as described in §3.2. For a sequence of T RGB frames, the variables to be recovered are then:

- the $w \times h \times 3$ texture map $\mathbf{C}(\hat{\mathbf{u}})$,
- the $w \times h \times T \times 2$ mapping $\mathbf{w}(\hat{\mathbf{u}}, t)$,
- the $w \times h \times T$ mask sequence $b(\hat{\mathbf{u}}, t)$.

The caret notation means that the indicated variable can take only integer values. The notation $\tilde{\mathbf{C}}$ will refer to the table of values $\{\mathbf{C}(\hat{u}, \hat{v}), \hat{\mathbf{u}} \in 0..w, \hat{v} \in 0..h\}$. There is a rescaling of axes implicit in the (u, v) discretizations, as the unit cube is mapped to a $w \times h$ rectangle, but we ignore this by redefining $\mathcal{Q}$ to be the rectangle $[0, w] \times [0, h]$ in the continuous domain. Any such reparametrization of the model does not affect the generated sequence.

The goal of this paper is to recover the unknown variables $\tilde{\mathbf{C}}, \tilde{\mathbf{w}}, \tilde{b}$ from the given data $\tilde{\mathbf{I}}$. At first sight, this task seems poorly defined: if we assume w and h are approximately equal to W and H, the number of unknowns to be estimated is of the order of the video size: the images provide $3whT$ measurements, while the unknowns number $3wh + whT$ × (2 scalars + 1 bool). However, by casting the problem as energy minimization, the decomposition into color and motion allows strong regularizers to be placed on $\mathbf{C}$, $\mathbf{w}$ and b. The energy measures the accuracy with which the parameters explain the input frames, as well as the *a priori* plausibility of the parameters.

The energy, like the image generation process above, is naturally described in a continuous formulation, with conversion to discrete form involving a number of simple but tedious integrals. It may appear that some of the terms will be difficult to optimize, but the alternating optimization strategy presented in §3 means that only a subset of the variables appear in each optimization step.

This derivation will assume that the user has provided a good segmentation of the sequence into object layers; the generalization to unknown or imperfectly known segmentations will be discussed in §8.1.

2.2 Data cost

The first term in the energy is the data cost, encouraging the model to predict the input sequence, and in particular to explain every input image pixel. If the input frames are $\tilde{\mathbf{I}}(\hat{\mathbf{x}}, t)$, the basic form of the data cost is the sum

$$E_{\text{data}} = \sum_t \sum_{\hat{\mathbf{x}}} \|\tilde{\mathbf{I}}(\hat{\mathbf{x}}, t) - \mathbf{I}(\hat{\mathbf{x}}, t)\|_\tau \qquad (6)$$

The robust norm $\|e\|_\tau = \min(\|e\|, \tau)$ deals with outlier pixels due to lighting or small unmodelled occlusions. The setting of τ is discussed in §5.

This cost is a discrete sum over the point samples in $\tilde{\mathbf{I}}$, but contains a continuous integral in the evaluation of $\mathbf{I}(\hat{\mathbf{x}}, t)$. Evaluating the integral yields the discrete model

$$\mathbf{I}(\mathbf{x}, t) = \frac{\sum_{\hat{\mathbf{u}}} A(\hat{\mathbf{u}}, \mathbf{x}, t)b(\hat{\mathbf{u}})\mathbf{C}(\hat{\mathbf{u}})}{\sum_{\hat{\mathbf{u}}} A(\hat{\mathbf{u}}, \mathbf{x}, t)b(\hat{\mathbf{u}})} \qquad (7)$$

where the weights $A(\hat{\mathbf{u}}, \mathbf{x}, t)$ are a function of the mapping $\mathbf{w}$, its Jacobian, and the PSF. They measure the contribution of each $\hat{\mathbf{u}}$ point to pixel $\mathbf{x}$, and will be zero at all but a few points $\hat{\mathbf{u}}$. The data cost is then

$$\sum_t \sum_{\hat{\mathbf{x}}} \left\| \tilde{\mathbf{I}}(\hat{\mathbf{x}}, t) - \frac{\sum_{\hat{\mathbf{u}}} A(\hat{\mathbf{u}}, \mathbf{x}, t)b(\hat{\mathbf{u}})\mathbf{C}(\hat{\mathbf{u}})}{\sum_{\hat{\mathbf{u}}} A(\hat{\mathbf{u}}, \mathbf{x}, t)b(\hat{\mathbf{u}})} \right\|_\tau . \qquad (8)$$

At points in the implementation we shall use an approximation to the correct integrals which is given by

$$A(\hat{\mathbf{u}}, \mathbf{x}, t) = \begin{cases} \mathtt{J}(\hat{\mathbf{u}}) & \text{for } \hat{\mathbf{u}} \in U(\mathbf{x}, t) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where $U(\mathbf{x}, t)$ is the set of all texture-map pixels which project to a given image pixel, defined by $U(\mathbf{x}, t) = \{\hat{\mathbf{u}} | \rho(\mathbf{w}(\hat{\mathbf{u}}, t) - \mathbf{x}) > 0\}$, i.e. "the points $\mathbf{u}$ which map to nonzero values of the point-spread function at $\mathbf{x}$".

2.3 Constraints

Simply minimizing the data term can yield a reconstruction which maps every texture pixel to a single image pixel. That is, if there is any colour which appears in every frame, then set $C(\mathbf{u})$ to that colour for all $\mathbf{u}$, and set the mapping $\mathbf{w}(u, v, t) = (\frac{u}{w} - \frac{1}{2} + x(t), \frac{v}{h} - \frac{1}{2} + y(t)) \forall \mathbf{u}$, where $(x(t), y(t))$ is a pixel of that colour in frame t. This gives $E_{\text{data}} = 0$, for any setting of b. Thus we must restrict the search for models to those which explain every pixel. This is imposed as a soft penalty based on a "count of contributing pixels"

$$C(\mathbf{x}, t) = \sum_{\hat{\mathbf{u}} \in U(\mathbf{x}, t)} b(\hat{\mathbf{u}}). \tag{10}$$

This yields an energy term

$$\sum_t \sum_{\hat{\mathbf{x}}} \begin{cases} \tau_c & C(\mathbf{x}, t) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where the threshold τ_c is a parameter of the algorithm. This formulation may then be tractably optimized using graph-cut.

2.4 Mapping smoothness

The mapping $\mathbf{w}$ is a proxy for the projection of a 3D surface, which we assume to be undergoing smooth deformations over time. We might further assume a relatively smooth camera motion. However, these assumptions are not sufficient to ensure that $\mathbf{w}$ is smooth, so conventional smoothness terms, such as thin-plate splines, are not appropriate.

Instead, we wish to encourage the recovered texture map to be sampled such that each texture pixel is taken from the input frame in which it is most fronto-parallel. Equivalently, the mapping is encouraged to be fronto-parallel in at least one frame. Without camera roll or zoom this could be expressed as the energy

$$\sum_{\hat{\mathbf{u}}} \min_t \|\mathtt{J}(\hat{\mathbf{u}}) - \mathtt{I}\|_F^2, \tag{12}$$

saying that, for each $\mathbf{u}$, the mapping Jacobian should be close to the identity (in Frobenius norm) in at least one frame. As written, it does not account for rotation about the camera optical centre or zoom, so in practice we estimate an overall affine transformation for each frame, called $\mathtt{H}_t$, and minimize

$$E_w = \sum_{\hat{\mathbf{u}}} \min_t \|\mathtt{J}(\hat{\mathbf{u}}) - \mathtt{H}_t\|_F^2 \tag{13}$$

Although this appears to offer no spatial smoothing, it can be argued that in combination with a temporal coherence term of the form

$$E_{\text{temporal}} = \sum_{\hat{\mathbf{u}}, t} \|\mathbf{w}_t(\hat{\mathbf{u}}, t)\|^2 \tag{14}$$

it leads to a spatial regularizer akin to a weak membrane [Blake and Zisserman 1987]. The compelling feature of this regularizer is that it leads to an excellent way to initialize the parametrization, as will be discussed in §3.2.

2.5 Visibility smoothness

We recall that the visibility map b is used to represent the effects of hidden surface removal, without explicitly modelling the 3D geometry. Instead, we observe that discontinuities in b are rare, and define a Potts energy which counts discontinuities, as used in image segmentation [Boykov and Jolly 2001]:

$$E_{\text{b}} = \sum_{(\hat{\mathbf{u}}, \hat{\mathbf{u}}') \in \mathcal{N}, t} \text{Potts}(b(\hat{\mathbf{u}}, t), b(\hat{\mathbf{u}}', t)) \tag{15}$$

where $\mathcal{N}$ is the set of 2×1 neighbourhoods, and Potts(b_1, b_2) is 1 if $b_1 \neq b_2$ and zero otherwise. A similar term is applied temporally, taking the mapping into account:

$$E_{\text{btemporal}} = \sum_{\hat{\mathbf{u}}, t} \text{Potts}\big(b(\hat{\mathbf{u}}, t), b(\hat{\mathbf{u}} + \Delta\mathbf{u}(\hat{\mathbf{u}}, t), t)\big) \tag{16}$$

where $\Delta\mathbf{u}(\mathbf{u}, t) = \mathtt{J}(\mathbf{u}, t)^{-1}(\mathbf{w}(\mathbf{u}, t+1) - \mathbf{w}(\mathbf{u}, t))$, using the Jacobian to convert local displacements in the image into displacements on the mosaic.

2.6 Texture prior

A final regularizing term encourages the texture map $\tilde{\mathbf{C}}$ to have the same texture statistics as the input sequence. Following [Woodford et al. 2007], we encourage neighbouring pixels in the texture map to come from the same input image. This energy term is complex to write down, but simple to implement, so we defer its description until the next section.

3 Minimizing the energy

A linear combination of the above terms yields the overall energy. The energy is written as a function of the discrete variables $\tilde{\mathbf{C}}, \tilde{\mathbf{w}}, \tilde{b}$:

$$\begin{aligned} E(\tilde{\mathbf{C}}, \tilde{\mathbf{w}}, \tilde{b}) = & E_{\text{data}}(\tilde{\mathbf{C}}, \tilde{\mathbf{w}}, \tilde{b}) + \\ & \lambda_1 E_{\text{w}}(\mathbf{w}) + \lambda_2 E_{\text{wtemporal}}(\mathbf{w}) + \\ & \lambda_3 E_{\text{b}}(b) + \lambda_4 E_{\text{btemporal}}(b) \end{aligned} \tag{17}$$

Here, and later, several tuning parameters appear in the energy, which must be set. Section 5 discusses the issue of parameter setting in more detail.

The energy is minimized by coordinate descent, optimizing for subsets of the variables in turn. Minimization with respect to each subset of variables yields each of the algorithm steps outlined in the introduction.

3.1 Minimizing over $\mathbf{C}$: stitching

In stage 3b of the algorithm, we are given the mapping $\mathbf{w}$, and the occlusion mask b, and must solve for the texture map $\mathbf{C}$. Notice that only the E_{data} term of the energy depends on $\mathbf{C}$, so for fixed $\mathbf{w}$ and b, the minimization is simply $\mathbf{C} = \text{argmin}_{\mathbf{C}} E_{\text{data}}$. Minimization under the robust norm (8) can be cast as a graph-cut problem by restricting the choice of $\mathbf{C}$. Specifically, an integer label $s(\hat{\mathbf{u}})$ is associated with each texture map pixel $\hat{\mathbf{u}}$, which indicates one of the input frames from which $\mathbf{C}(\hat{\mathbf{u}})$ is to be chosen. The input images are warped by the inverse of $\mathbf{w}$, to generate registered images $\mathbf{I}^w(\hat{\mathbf{u}}, t)$, from which $\mathbf{C}$ is optimized at any pixel $\hat{\mathbf{u}}$ by computing

$$s^* = \underset{s}{\text{argmin}} \sum_t \|\mathbf{I}^w(\hat{\mathbf{u}}, t) - \mathbf{I}^w(\hat{\mathbf{u}}, s)\|_\tau \tag{18}$$

and setting $\mathbf{C} = \mathbf{I}^w(\hat{\mathbf{u}}, s^*)$. At this point it is easy to add a texture prior to the original energy, which encourages adjacent pixels in

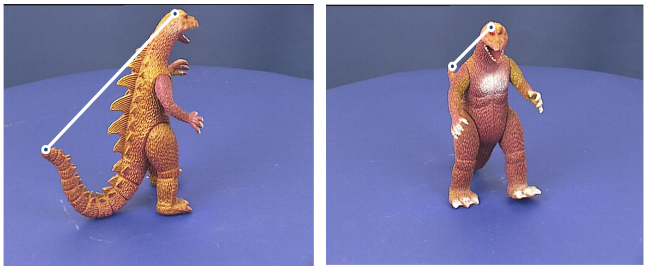

Figure 6: Maximum distance. *Two object points, with (*a priori *unknown) texture coordinates* $\mathbf{u}_1$ *and* $\mathbf{u}_2$. *From tracking, we obtain trajectories* $\mathbf{x}_k(t) = [\mathbf{w}(\mathbf{u}_k, t)]_t$ *where* $k \in \{1, 2\}$. *The embedding energy encourages distances in parameter space to be at least the farthest separation of the trajectories, i.e.* $\|\mathbf{u}_1 - \mathbf{u}_2\| \geq \max_t \|\mathbf{x}_1(t) - \mathbf{x}_2(t)\|$.

the texture map to be taken from the same input images, yielding an energy of the form

$$E(\tilde{s}) = \sum_{\hat{\mathbf{u}}} \sum_t \|\mathbf{I}^w(\hat{\mathbf{u}}, t) - \mathbf{I}^w(\hat{\mathbf{u}}, s(\hat{\mathbf{u}}))\|_\tau + \\ + \lambda_{\text{texture}} \sum_{\{\hat{\mathbf{u}}, \hat{\mathbf{u}}'\} \in \mathcal{N}} \mu(\hat{\mathbf{u}}, \hat{\mathbf{u}}') \text{Potts}(s(\hat{\mathbf{u}}), s(\hat{\mathbf{u}}')) \quad (19)$$

where $\mu(\cdot, \cdot)$ measures patch overlap as in [Agarwala et al. 2004].

3.2 Reparametrization and embedding

An important variable which does not appear as an explicit parameter of the energy functional relates to the parametrization of $\mathbf{u}$ space. The data cost E_{data} is—by construction—invariant to reparametrization, but the warp costs (13), (14) are not. It turns out that addressing this question leads to the crucial step in initializing the energy minimization as we now show.

The initialization of the overall algorithm consists in obtaining sparse point tracks (step 2 above). The i^{th} track is the set $\{\tilde{\mathbf{x}}(\mathbf{u}_i, t) \mid t \in T_i\}$ where T_i is the set of frame indices in which the point is tracked, and $\mathbf{u}_i$ is the *unknown* preimage of the track in parameter space. Finding these—$\mathbf{u}_i$ will anchor all other computations.

Finding the optimal parametrization then consists in assigning the $\mathbf{u}_i$ values such that the warp terms $E_{\text{w}}(\mathbf{w}) + E_{\text{wtemporal}}(\mathbf{w})$ are minimized. For a given pair of tracks, with coordinates $\mathbf{u}_i$ and $\mathbf{u}_j$, we wish to know the energy of the mapping which minimizes the regularizer subject to the mapping being consistent with the tracks. Specifically, we require the value of

$$\begin{aligned} \min_{\mathbf{w}} \quad & E_{\text{w}}(\mathbf{w}) + E_{\text{wtemporal}}(\mathbf{w}) \\ \text{such that} \quad & \mathbf{w}(\mathbf{u}_i, t) = \tilde{\mathbf{x}}(\mathbf{u}_i, t) \quad \forall t \in T_i \\ & \mathbf{w}(\mathbf{u}_j, t) = \tilde{\mathbf{x}}(\mathbf{u}_j, t) \quad \forall t \in T_j. \end{aligned} \quad (20)$$

Note that only the value of the minimizing energy is required, not the mapping itself. It can be shown that the minimal energy in the pairwise case, as a function of $\mathbf{u}_i$ and $\mathbf{u}_j$, is

$$\left(\frac{\|(\tilde{\mathbf{x}}(\mathbf{u}_i, t^*_{ij}) - \tilde{\mathbf{x}}(\mathbf{u}_j, t^*_{ij})) - (\mathbf{u}_i - \mathbf{u}_j)\|}{\|\mathbf{u}_i - \mathbf{u}_j\|} \right)^2 \|\mathbf{u}_i - \mathbf{u}_j\| \\ \text{where } t^*_{ij} = \operatorname*{argmax}_{t \in T_i \cap T_j} \|\tilde{\mathbf{x}}(\mathbf{u}_i, t) - \tilde{\mathbf{x}}(\mathbf{u}_j, t)\| \quad (21)$$

Given several tracks as above, the $\mathbf{u}_i$ are chosen to minimize the sum of weighted distances

$$\sum_{ij} \left\|\mathbf{d}_{ij} - (\mathbf{u}_i - \mathbf{u}_j)\right\|^2 \frac{1}{\|\mathbf{u}_i - \mathbf{u}_j\|} \\ \text{where} \quad \mathbf{d}_{ij} := \tilde{\mathbf{x}}(\mathbf{u}_i, t^*_{ij}) - \tilde{\mathbf{x}}(\mathbf{u}_j, t^*_{ij}) \quad (22)$$

Note that this is analogous to embedding via multi-dimensional scaling [Cox and Cox 2001], but with a distance weighting term $\frac{1}{\|\mathbf{u}_i - \mathbf{u}_j\|}$. The minimization is implemented as an iterated reweighted least squares problem. In practice, to avoid numerical issues when $\mathbf{u}_i$ and $\mathbf{u}_j$ become close during optimization, we use an exponential weighting $\exp(-(\|\mathbf{u}_i - \mathbf{u}_j\|/\tau_3)^2)$. The affine transformation $\mathtt{H}_t$ is estimated from sparse tracks and applied before embedding.

The implementation is as follows. Each pair of tracks is assigned a random weight μ_{ij}, and the $\{\mathbf{u}_k | k = 1..T\}$ which minimize the quadratic form

$$\sum_{ij} \mu_{ij} \left\|\mathbf{d}_{ij} - (\mathbf{u}_i - \mathbf{u}_j)\right\|^2 \quad (23)$$

are found. The μ_{ij} are then recomputed with the new $\mathbf{u}$, using $\mu_{ij} = \exp(-(\|\mathbf{u}_i - \mathbf{u}_j\|/\tau_3)^2)$, and the process is iterated to a fixed point. The embedding is restarted several times, with different random initialization, and the $\mathbf{u}$'s which minimize the original energy are retained.

Since we can have many thousands of tracks, we initialize with a subset of tracks (1000, for example), and solve the embedding for that subset. Then, those $\mathbf{u}$ values are fixed, and the next subset is minimized, *including* the pairwise terms which link from the new set to the set for which we already have a solve. Each subproblem is a quadratic form of size 1000 × 1000, which allows us to compute the embedding on a desktop PC in a few minutes.

The mosaic size is naturally selected by this process: because distances in (u, v) space are measured in pixels, and because pairs of points are encouraged to be as far apart as their longest separation in the input sequence, a simple bounding box of the recovered coordinates is ideally sized to store the model without loss of resolution.

Comparison with nonrigid structure from motion It is instructive to compare this process with the nonrigid structure from motion algorithms of [Torresani et al. 2008]. There, as here, the input is a set of n tracks, $\tilde{\mathbf{x}}(\mathbf{u}_i, t)$. In that algorithm the tracks are concatenated to make a $2T \times n$ measurement matrix, of which a low-rank approximation is formed. Finding this approximation (to rank 6, say) amounts to saying that each track is a linear projection of a point in a $\mathbb{R}^6$, effectively finding an embedding of the tracks into $6D$ space. In our case, the embedding is not constrained to be linear, so it can map from a lower dimensional space—indeed it maps from 2D space, the natural domain of the texture map.

3.3 Minimizing over w: dense mapping

Before step 3b, it is necessary to obtain a dense mapping $\mathbf{w}$, which is obtained given the tracks and their embedding coordinates. In this case, (20) is minimized with one constraint per track, and the resulting $\mathbf{w}$ can be shown in 1D to be linear interpolation of the sparse track data. Although the 2D case has not, to our knowledge, been characterized, we assume an analogous situation and simply use MATLAB's `griddata` to interpolate. Although this unvalidated assumption means that there is no guarantee of minimizing the original energy, it is a simple matter to check that the overall energy has reduced at each iteration, and to reject iterations where it increases. Indeed, to adumbrate the discussion in [Rav-Acha et al. 2008], this is a useful general principle: energy minimization approaches are attractive because all the system tuning parameters are clearly defined and visible in the energy, but it is difficult to find closed-form minimizers for each energy component. However, using *ad-hoc* optimizers, even when they may have their own tuning parameters, will affect only rate of convergence, not correctness, if reduction of the original energy is verified at each stage.

3.4 Minimizing over $\mathbf{w}$ and b: dense mapping with occlusion

Given an initial approximation to $\mathbf{w}$ as above, we may solve simultaneously for b and a refined mapping. By solving for an update $\Delta\mathbf{w}$ to the initial estimate, the problem may be cast as one of optical flow computation. The minimization is now over all energy terms, as all terms depend on $\mathbf{w}$ and b.

The energy for the update is implemented as a variant of robust optical flow [Brox et al. 2004], alternating search for $\Delta\mathbf{w}$ and b on a multiresolution pyramid.

Let $\tilde{\mathbf{C}}$ be the current estimate of the texture map and let $\mathbf{w}^0$ be the current mapping estimate. Then we wish to determine the update $\Delta\mathbf{w}$ which minimizes

$$E_{\text{data}}(\Delta\mathbf{w}) = \sum_{\hat{\mathbf{u}}} b(\hat{\mathbf{u}})\|\tilde{\mathbf{C}}(\hat{\mathbf{u}}) - \tilde{\mathbf{I}}(\mathbf{w}^0(\hat{\mathbf{u}}) + \Delta\mathbf{w}, t)\|^2 \tag{24}$$

under the local regularizers

$$E_{\Delta\mathbf{w}} = \lambda_{\text{wl}} \sum_{\hat{\mathbf{u}}} \|\mathbf{w}^0_{uu} + \Delta\mathbf{w}_{uu}\|^2 + \|\mathbf{w}^0_{vv} + \Delta\mathbf{w}_{vv}\|^2, \tag{25}$$

with E_{b}, and $E_{\text{btemporal}}$ as above. Linearizing (24) gives a linear system in $\Delta\mathbf{w}$ which is readily solved.

Temporal smoothness is imposed via a forward/backward implementation where the mapping $\mathbf{w}$ and the mask b of the previous frame are transformed to the coordinate system of the current frame using the image-domain optic flow between frames, and added as a prior to the current estimate, as follows:

$$E_{\text{temporal}} = \sum_{\hat{\mathbf{u}}} \|\mathbf{w}_{\text{prev}}(\hat{\mathbf{u}}) - \mathbf{w}(\hat{\mathbf{u}})\|^2 + \|b_{\text{prev}}(\hat{\mathbf{u}}) - b(\hat{\mathbf{u}})\|^2.$$

3.5 Lighting

Lighting has been ignored throughout the discussion above. It is addressed in two ways. First, when matching interest points from frame to frame, (or from the mosaic to input frames), we use SIFT descriptors [Brown and Lowe 2007]. Second, when computing dense mappings, energy terms of the form

$$\sum_{\hat{\mathbf{u}}} \|\mathbf{I}(\mathbf{w}(\hat{\mathbf{u}})) - \mathbf{C}(\hat{\mathbf{u}})\|_\tau$$

are extended to include per-pixel intensity scaling terms $\alpha(\mathbf{x}), \beta(\mathbf{x})$ with a strong spatial smoothness prior, so the matching minimizes

$$\sum_{\hat{\mathbf{x}}} \|\alpha(\mathbf{x})\mathbf{I}(\mathbf{x},t) + \beta(\mathbf{x}) - \tilde{\mathbf{I}}(\mathbf{x},t)\|_\tau + \lambda_5 \sum_{\hat{\mathbf{x}}} \|\frac{\partial}{\partial\mathbf{x}}\alpha(\mathbf{x})\| + \|\frac{\partial}{\partial\mathbf{x}}\beta(\mathbf{x})\|.$$

This gives invariance to smoothly changing lighting without allowing any colour to match with any other. In turn, the above is implemented by defining α and β in terms of a coarse set of fixed basis functions whose weights are easily included in the pyramid-based matching scheme of §3.4. For efficiency, α and β are solved for only at coarse pyramid levels, and interpolated at the finest scale.

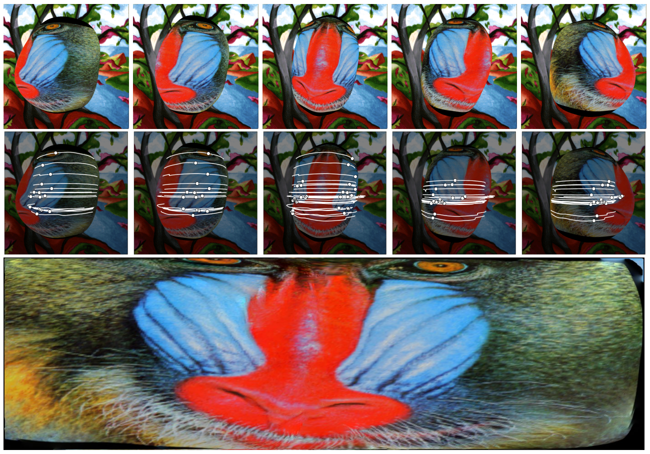

Figure 7: Reconstruction results, synthetic sequence. *(Top) Frames from a 100-frame synthetic sequence of a deforming surface. (Middle) Point tracks extracted fom the sequence, overlaid on the same frames. (Bottom) The texture map recovered by embedding point tracks into 2D.*

4 User interaction and hinting

Although the above algorithm can work completely automatically [examples are in figs 7, 8a, and 11], there will generally be situations where lack of texture, repeated structures, or motion blur, mean that the recovered mosaic does not cover all of the object. In this section we present user interactions that can be used to correct such errors.

The first example interaction is mentioned above: to provide a segmentation of the sequence, the user selects objects in a small number of frames, and the segmentation is propagated using optical flow or sparse tracks. Although automatic segmentation is sometimes possible, a sequence such as the giraffe example has (a) many foreground objects with similar colour statistics and (b) some giraffes are static, or exhibit similar movement to the foreground object. In this case, existing automatic segmentation algorithms will require human assistance.

A second example interaction deals with improving the mosaic coverage. Figure 8 shows a giraffe sequence in which the initial mosaic does not include all of the giraffe's head. By brushing on the head in one (inverse warped) frame $\mathbf{I}^w(\hat{\mathbf{u}}, t)$, the stitching variable $s(\hat{\mathbf{u}})$ is given fixed values for some set of $\hat{\mathbf{u}}$. Incorporating these as hard constraints in the optimization of (19) is trivial—the nodes representing the fixed values are simply removed from the graph—and yields better mosaic coverage which means that the mapping refinement stage can obtain good motion estimates over more of the object's surface.

5 Tuning parameters

For this work, the most important parameters are as follows.

The robust kernel width τ is set to match an estimate of image noise. For all experiments it was set to $5/255$ gray-levels, which means that the visibility masks for the noise-free synthetic sequence, where the appropriate value is lower, are smoother than the best that could have been obtained.

The scale parameter in the embedding distance calculation τ_3 is set to about 40 pixels for all our test sequences (PAL resolution) except the face, which had many outlier tracks, and necessitated a higher

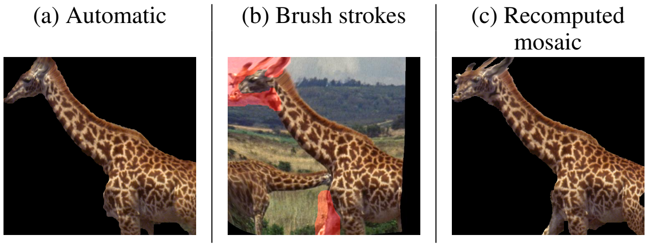

Figure 8: User interaction. *(a) Automatically computed mosaic for giraffe sequence. The far side of the head, and the distal leg, are not represented as they appear in too few frames. (b) User brush strokes on an intermediate frame inverse-warped into parameter space. The strokes indicate parts of this frame which should appear in the mosaic. (c) New mosaic computed with only the brush strokes in (b) as constraints. The head and leg are now present in the mosaic, so can be matched to the original sequence during mapping refinement.*

value. In future work we expect to use this as a convergence control, starting from a high value and reducing.

The spatial smoothness weight λ_{wl} in the proxy regularizer (25) controls the amount of deformation of the mapping. It was constant for all but the "boy" sequence, where it was reduced because the deformations are more vigorous.

The other terms generally define local interpolations whose precise details are less important, or are mostly designed to improve convergence and can be left at initial values at the cost of slower reconstructions.

6 Results

In testing the algorithm, a number of success criteria present themselves, but for video editing, the primary criterion is obviously the range and quality of effects that are enabled by the model. As it is rather difficult to evaluate these on the printed page, the reader is referred to the accompanying video, which should be studied in conjunction with the text below. Computation times for all of these sequences are of the order of a few hours.

6.1 Synthetic sequence

It is interesting to note that there is no concept of a "ground truth" model to be recovered, even with synthetic data. The synthetic sequence is generated by texture-mapping a deforming 3D surface, rendered over a static background. As shown in figure 4, the surface exhibits considerable self-occlusion, with about 30% of mosaic pixels visible in any one frame. There is also some more subtle self occlusion near the nose. The texture coordinates obey a roughly equal-area parametrization, but this does not constitute ground truth, because the minimum-energy parametrization of the mosaic minimizes the rather different metric (13). We can, however, visually evaluate the recovered mosaic (figure 7), confirming that it is approximately a diffeomorphic reparametrization of the model texturemap, with compression at the top and bottom, where the model was seen only obliquely, and some overall twists and warps. This is effectively a perfect result, as one would hope from a noiseless sequence, but illustrates the expected performance of the model fitting.

6.2 Face sequence

The face sequence, used as an example in figure 1, has similar topology and geometry to the synthetic sequence, with the skin surface deforming as the jaw moves. The surface texture, however, is less easy to match, with few high-contrast points coupled with strong lighting. Nevertheless, a good embedding is found, which gives a mosaic which is comparable with that which would be expected from an accurate 3D scan, but is captured here with a single camera and moving head. Rerendering the images with several superimposed edits leads to a believable augmentation which follows facial movements (see video).

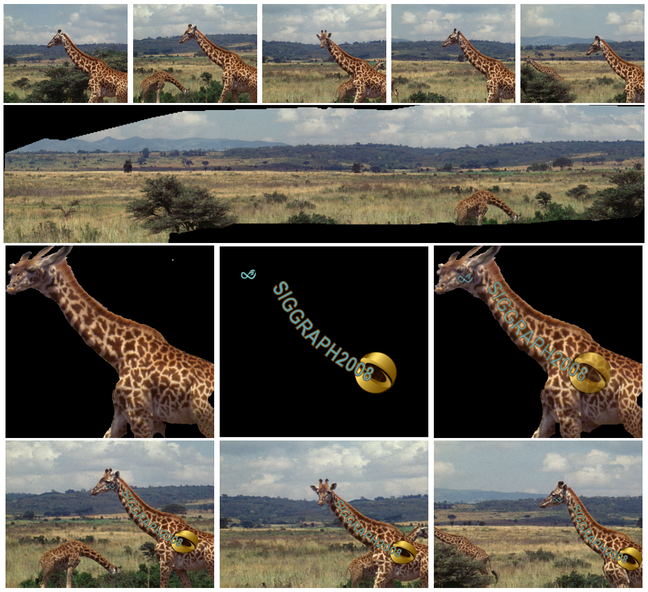

Figure 9: Giraffe sequence. *(Top) Several images from the sequence. (Row 2) The recovered mosaic for the background layer (including a static giraffe). (Row 3) Recovered foreground mosaic and edit layer. (Row 4) Frames from the edited sequence.*

6.3 Giraffe sequence: foreground

The giraffe sequence (figure 9) is archive footage which has been scanned from 16mm film, and exhibits a moderate amount of noise due to the film grain. We concentrate on the foreground giraffe and the far background.

A logo is placed on the foreground giraffe's back and head, and deforms with the skin over time. This task provides an interesting comparison with methods based on optical flow. With optical flow, the annotation drifts by about 10 pixels in 30 frames, while the unwrap mosaic shows no visible drift.

6.4 Giraffe sequence: background

An important and difficult example for our technique is the mosaicing of the background of the giraffe movie. It appears the simplest of the tests, because existing techniques based on homography estimation [Brown and Lowe 2007], would perform excellently. However, consider what the homography supplies: it is a parametric model which correctly describes the entire frame-to-frame motion field; the eight parameters can be robustly estimated given just a few tracks; and importantly the motion field is propagated easily through the sequence with relatively little drift simply by multiplying homography matrices. The unwrap mosaic model, in contrast, has only local track information (average track length is 15 frames, while the sequence is 380 frames), and a much weaker motion model. Despite this, the embedding produces texture coordinates almost as good as would be expected from homography estimation, and stitching the mosaic gives a representation comparable to the state of the art with the much simpler model.

6.5 Boy sequence

This sequence is stock footage of a boy walking in the woods. The protagonist walks through trees, occluding and being occluded, and then turns through 90 degrees so that both sides of his head and torso are shown to the camera, at different times. Matching is complicated by variable focus, motion blur, and considerable foreground occlusion, but again an effective mosaic is produced. The edits on the textureless areas of the cheek show some drift, but this is kept in check by the final matching stage, being evident only on close inspection. Note that the ear is doubled in this mosaic, and could be fixed by editing as in §4, but for these augmentations this was not necessary.

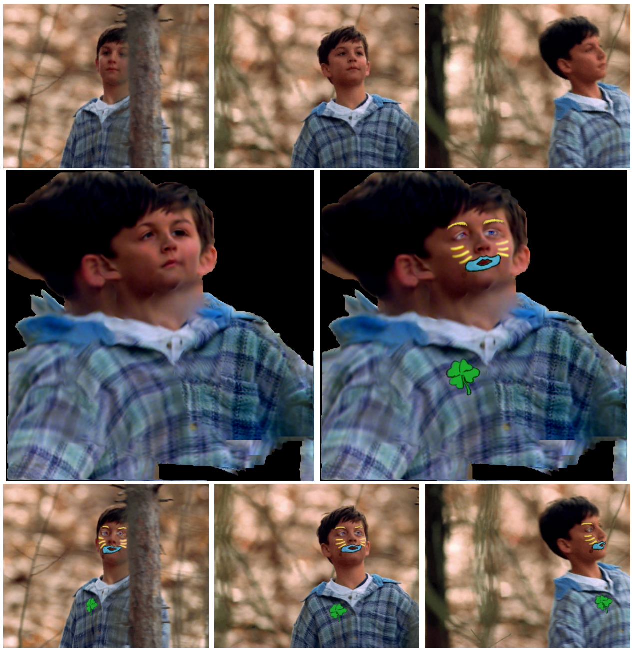

Figure 10: Boy sequence. *(Top) Several images from the sequence. (Middle) The recovered mosaic for the boy, and the edit layer. (Bottom) Edited frames.*

6.6 Dinosaur sequence

The dinosaur sequence (figures 6, 11) is a difficult case. Although not deforming, the depicted object has a complex shape with several self-occlusions. The recovered mosaic gets the gross layout of the object parts correct, but the texture represents only a subset of the model, and is certainly not a smooth reparametrization of the object's true "texture map". This result does not appear to be of particular use for editing, but does give hope that an iterative automatic segmentation of the sequence might yield separate mosaics for each model component: arms, torso, tail. We note as an aside that the unwrap mosaic technique is "deformation agnostic": rigid or nonrigid objects are largely equivalent in difficulty.

7 Related work

Even the name "unwrap mosaics" reveals that some of the ideas in this paper have a long history. Wang and Adelson's paper [1994] on layered representations is a clear progenitor of this work. In fact, it was thinking about how layers might apply to self-occludng objects that inspired the current model. Two important differences exist with our model. First is structural: Wang and Adelson's layers do not allow b to vary over time, a difference which has significant consequences. In particular, a fixed b means that the same set of texture map pixels must be visible throughout the sequence, precluding its application to any 3D object which rotates with respect to the camera. Second, although it is suggested that a dense nonparametric flow field might be associated with the mosaic, in practice the mapping was restricted to an affine transformation per layer. Indeed, almost all developments of layers have used some form of whole-image parametric motion estimate. Irani *et al.* [1995], for example, include more elaborate parametric transformations, registering layers using "6-parameter affine transformations" or "8-parameter quadratic transformations", and also demonstrate many of the editing examples we show in this paper, subject to the limitations imposed by the parametric mapping.

Many authors have reported developments of the ideas in these two papers, for example dealing with non-opaque layers, computing super-resolution mosaics, computing summary mosaics and other representations such as Layered Depth Images [Shade et al. 1998]. An excellent recent overview is included in the work on transferring information from still photos to mosaics [Bhat et al. 2007]. The latter paper shows how, for rigid scenes, dense 3D reconstruction can give motion fields and occlusion masks which extend the homography-based work to scenes containing rigid 3D objects. In parallel to computer vision and graphics research into motion estimation, the effects industry has developed tools and algorithms to compute optical flow, and to apply it to post-production tasks. Seymour [2006] provides a perspective from the viewpoint of a special effects artist, which indicates the many uses to which motion estimates can be put.

The measurement of dense motion fields is also an area which has seen considerable research. A major theme has been the computation of *optic flow*, i.e. a dense motion field between successive pairs of images in a sequence. A recent benchmarking paper [Baker et al. 2007] cites several surveys and other comparison papers, and benchmarks a modern energy-based formulation [Bruhn et al. 2005] ahead of the widely used implementation of Black and Anandan [1993]. Characteristic problems that exist even in modern algorithms are that the estimate is poor at depth discontinuites, and that integrating flow over many frames leads to considerable drift. In our work, we assume that a segmentation of the scene into layers is possible using coarse flow information, so drift is the more important of these two difficulties.

Toklu et al. [2000] is the closest precursor to our work, using optic flow to update a mosaic representation. The disadvantage of their

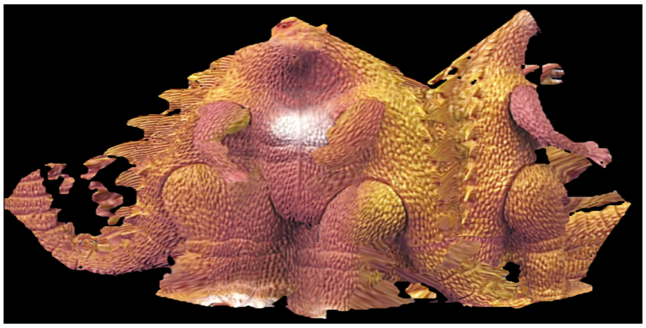

Figure 11: Dinosaur sequence. *The recovered mosaic of the dinosaur. Protrusions such as the snout, arms, and tail are poorly handled by the model. The cylinder topology is not modelled, so the right arm appears twice, but this would be an easy extension.*

algorithm is that it updates the representation in a somewhat ad-hoc manner, at every new frame, and is thus subject to drift, and does not form a mosaic in which errors are consistently distributed over the sequence.

Recently, drift has been addressed by maintaining the "identity" of object points over time, for example by tracking interest points and interpolating dense motion. Sand and Teller [2006] describe a comprehensive system based on point tracking, and their experiments show greatly reduced drift on several example sequences. Dense motion fields may also be obtained from sparse feature tracks by fitting motion models corresponding to multiple 3D motions (Bhat et al. [2007] describe a recent system), or to deformable 3D models [Bregler et al. 2000, et seq.]. The latter techniques, mentioned earlier in this paper, are based on the observation that Tomasi-Kanade factorization extends to multiple motions and deformable models if the factorization rank r is taken to be higher than 3. The difficulty with interpolating these tracks is that some 2D neighbourhood system must be defined either in the r-D space in which the latent coordinates live, or in the image. The latter cannot be propagated through occlusions, and the former introduces considerable difficulties when the density of reconstructed points is not very close to uniform. Because we embed directly into 2D, we avoid both issues.

Related to our embedding technique is the family of methods which discover texture coordinates for predefined 3D surface models, typically represented as polygon meshes. Zigelman *et al.* [2002] represents an example using multi-dimensional scaling and includes a survey of related methods. Zhou *et al.* [2005] show how manual specification of texture coordinates allows a set of images to be texture-mapped onto an arbitrary polygon mesh. They also allude to the problem of recovering texture maps for rigid 3D models obtained from real images. Lempitsky and Ivanov [2007] present a recent survey, noting that even with a 3D model acquired directly from images, the registration of the model will generally not be sufficiently accurate to obtain a seamless texture map. Their method, like that described here, is built on mosaic stitching, but requires a rigid 3D object from which to recover camera positions, silhouettes, and ultimately a 2D–2D mapping between texture space and the images.

An important general principle in related literature is the use of energy formulations to regularize models and to make modelling assumptions coherent and explicit. A consistent energy formulation for layers was described by Fleet *et al.* [2002], while recent optic flow formulations [Brox et al. 2004; Bruhn et al. 2005] also benefit from an energy formulation. Energy formulations may also be derived from a probabilistic viewpoint [Frey et al. 2003], often yielding optimization algorithms with superior convergence characteristics. Frey *et al.*, for example derive an algorithm which converges from a wide range of starting positions, but is practical only for short low-resolution sequences with relatively simple motion models. In contrast, our energy is specifically designed so that a good initialization can be computed, meaning that conventional optimizers can be brought to bear.

8 Discussion

This paper presents a new representation for videos of deforming 3D objects. By making the texture map, rather than the shape, the primary object of reconstruction, many difficulties which have hampered earlier representations are resolved. Even without 3D shape recovery, the representation allows many manipulations of the video which respect the three-dimensionaliy of the scene, using only 2D mappings.

The key step in our algorithm is viewing reconstruction as an embedding of tracks into 2D. This gives us the topology of the underlying 3D points for free, while other algorithms have either had to discover it as a post-process, or were limited to simple shapes such as quadrics and planes. It might be thought that the embedding would be unstable, especially given that more constrained models (such as factorization) have difficulties with the large amounts of missing data that self-occlusion causes. In practice, it is possibly the most robust component of the system, because the linear algorithm at its core scales to large amounts of data. This means that it may use all the information in the input tracks, particularly the information in the very many short tracks.

The other important step is the mosaic stitching. It has the property that even with quite poor estimates of the warp field, yielding poor inverse-warp images $\mathbf{I}^w(\hat{\mathbf{u}}, t)$, a good mosaic is often obtained, and hence re-mapping can correct the original estimate. Combined with the manual interaction described in §4, even quite difficult sequences can yield useful mosaics.

It should be noted that the approximate algorithms we use to minimize each slice of the energy function offer no guarantees of globally minimizing the overall energy [Rav-Acha et al. 2008]. We hope that further research will yield better guarantees.

8.1 Generalizations

A natural generalization is to automatically segment the layers. This is a subject with a long research history (see [Bhat et al. 2007] for one review), and many existing methods could be used as a preprocess here. A class of model that is relevant to our work is that in which sparse point tracks are clustered into independent motions by, for example, factorization-based structure from motion [Costeira and Kanade 1998]. This stage can be replaced by a robost analogue of our embedding, where the 2-norm in (22) is replaced by a truncated quadratic cost, marking as "outliers" some links between track pairs ij. Normalized cuts analysis [Shi and Malik 1997] on these links yields a segmentation into layers which separates objects based on their embedding consistency, rather than any specific motion model. Implementations of this on some of our sequences can produce qualitatively correct results, but the optimization is fragile and requires further investigation. When minimizing the overall energy, the constraints on b must now include contributions from all layers, modifying (10). Optimizing each layer simultaneously would then be rather more computationally expensive, but it might be hoped that alternating minimization over one layer at a time would provide some benefits.

Another natural generalization would appear to be to allow non-boolean masks b, allowing for alpha-blending of the layers. In the continuous case, this is unnecessary, because the PSF correctly describes the colour of mixed pixels. In the discrete case, it may be valuable to allow continuous visibility, but this remains to be tested.

A simple extension would be to apply matrix factorization to the input tracks to recover a deformable 3D shape. The 2D mapping then provides topology which could allow model reconstruction for shapes which rotate about the vertical in front of the camera (i.e. turntable-like sequences). This has not, to our knowledge, ever been demonstrated using nonrigid factorization, due to the missing data it induces. Conversely the factorization can place constraints on the mapping which would allow outlier removal. Forming the 3D model would also allow exact visibility to be computed, providing constraints on b.

A possible difficulty with user interaction is that the recovered mosaic may be an arbitrary warp of the mosaic that the editing artist expects. However, edits may also be performed by painting on a frame of the original video, and then using the mapping to propagate the edit through the sequence. For self-occluding surfaces, the edit can be made on one frame and then updated in others.

Acknowledgements

Discussions with many people have contributed to this work in its long development. To make a long list short, we thank John Winn, Andrew Blake, and the SIGGRAPH reviewers, all of whom made considerable contributions.

References

2D3 LTD., 2008. Boujou 4: The virtual interchangeable with the real. http://www.2d3.com.

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S. M., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. F. 2004. Interactive digital photomontage. *ACM Trans. Graph. (Proc. of* SIGGRAPH*) 23*, 3, 294–302.

BAKER, S., SCHARSTEIN, D., LEWIS, J. P., ROTH, S., BLACK, M., AND SZELISKI, R. 2007. A database and evaluation methodology for optical flow. In *Proc. ICCV*.

BHAT, P., ZITNICK, C. L., SNAVELY, N., AGARWALA, A., AGRAWALA, M., COHEN, M., CURLESS, B., AND KANG, S. B. 2007. Using photographs to enhance videos of a static scene. In *Eurographics Symposium on Rendering*.

BLACK, M. J., AND ANANDAN, P. 1993. A framework for the robust estimation of optical flow. In *Proc. ICCV*, 231–236.

BLAKE, A., AND ZISSERMAN, A. 1987. *Visual Reconstruction*. MIT Press.

BOYKOV, Y., AND JOLLY, M.-P. 2001. Interactive graph cuts for optimal boundary and region segmentation of objects in n-D images. In *Proc. ICCV*, 105–112.

BRAND, M. 2001. Morphable 3D models from video. In *Proc. CVPR*, vol. 2, 456–463.

BREGLER, C., HERTZMANN, A., AND BIERMANN, H. 2000. Recovering non-rigid 3D shape from image streams. In *Proc. CVPR*, 690–696.

BROWN, M., AND LOWE, D. G. 2007. Automatic panoramic image stitching using invariant features. *Intl. J. Comput. Vision 74*, 1, 59–73.

BROX, T., BRUHN, A., PAPENBERG, N., AND WEICKERT, J. 2004. High accuracy optical flow estimation based on a theory for warping. In *Proc. ECCV*, 25–36.

BRUHN, A., WEICKERT, J., AND SCHNÖRR, C. 2005. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *Intl. J. of Computer Vision 61*, 3, 211–231.

COSTEIRA, J. P., AND KANADE, T. 1998. A multibody factorization method for independently moving objects. *Intl. J. of Computer Vision 29*, 3, 159–179.

COX, M., AND COX, M. A. A. 2001. *Multidimensional Scaling*. Chapman and Hall.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs. In *Proc. ACM Siggraph*.

FLEET, D., JEPSON, A., , AND BLACK, M. 2002. A layered motion representation with occlusion and compact spatial support. In *Proc. ECCV*, 692–706.

FREY, B. J., JOJIC, N., AND KANNAN, A. 2003. Learning appearance and transparency manifolds of occluded objects in layers. In *Proc. CVPR*.

GAY-BELLILE, V., BARTOLI, A., AND SAYD, P. 2007. Direct estimation of non-rigid registrations with image-based self-occlusion reasoning. In *Proc. ICCV*.

GU, X., GORTLER, S. J., AND HOPPE, H. 2002. Geometry images. *ACM Trans. Graph. (Proc. of* SIGGRAPH*)*, 355–361.

IRANI, M., ANANDAN, P., AND HSU, S. 1995. Mosaic based representations of video sequences and their applications. In *Proc. ICCV*.

LEMPITSKY, V., AND IVANOV, D. 2007. Seamless mosaicing of image-based texture maps. In *Proc. CVPR*, 1–6.

LI, Y., SUN, J., AND SHUM, H.-Y. 2005. Video object cut and paste. *ACM Trans. Graph. (Proc. of* SIGGRAPH*) 24*, 3, 595–600.

RAV-ACHA, A., KOHLI, P., ROTHER, C., AND FITZGIBBON, A. 2008. Unwrap mosaics. Tech. rep., Microsoft Research. http://research.microsoft.com/unwrap.

SAND, P., AND TELLER, S. J. 2006. Particle video: Long-range motion estimation using point trajectories. In *Proc. CVPR*, 2195–2202.

SEETZEN, H., HEIDRICH, W., STUERZLINGER, W., WARD, G., WHITEHEAD, L., TRENTACOSTE, M., GHOSH, A., AND VOROZCOVS, A. 2004. High dynamic range display systems. *ACM Trans. Graph. (Proc. of* SIGGRAPH*) 23*, 3, 760–768.

SEITZ, S. M., CURLESS, B., DIEBEL, J., SCHARSTEIN, D., AND SZELISKI, R. 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. CVPR*, vol. 1, 519–526.

SEYMOUR, M. 2006. Art of optical flow. *fxguide.com: Feature Stories* (Dec.).

SHADE, J. W., GORTLER, S. J., HE, L.-W., AND SZELISKI, R. 1998. Layered depth images. In *Proc. ACM Siggraph*, 231–242.

SHI, J., AND MALIK, J. 1997. Normalized cuts and image segmentation. In *Proc. CVPR*, 731–743.

THORMÄHLEN, T., AND BROSZIO, H., 2008. Voodoo Camera Tracker: A tool for the integration of virtual and real scenes. http://www.digilab.uni-hannover.de/docs/manual.html.

TOKLU, C., ERDEM, A. T., AND TEKALP, A. M. 2000. Two-dimensional mesh-based mosaic representation for manipulation of video objects with occlusion. *IEEE Trans. Image Proc. 9*, 9, 1617–1630.

TORRESANI, L., HERTZMANN, A., AND BREGLER, C. 2008. Non-rigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE Trans. PAMI*, (to appear).

TURK, G., AND LEVOY, M. 1994. Zippered polygon meshes from range images. In *Proc. ACM Siggraph*, 311–318.

VAN DEN HENGEL, A., DICK, A., THORMÄHLEN, T., WARD, B., AND TORR, P. H. S. 2007. VideoTrace: Rapid interactive scene modelling from video. *ACM Trans. Graph. (Proc. of* SIGGRAPH*)*.

WANG, J. Y. A., AND ADELSON, E. H. 1994. Representing moving images with layers. *IEEE Trans. Image Proc. 3*, 5, 625–638.

WOODFORD, O. J., REID, I. D., AND FITZGIBBON, A. W. 2007. Efficient new-view synthesis using pairwise dictionary priors. In *Proc. CVPR*.

ZHOU, K., WANG, X., TONG, Y., DESBRUN, M., GUO, B., AND SHUM, H.-Y. 2005. Texture-Montage: Seamless texturing of surfaces from multiple images. *ACM Trans. Graph. (Proc. of* SIGGRAPH*)*, 1148–1155.

ZIGELMAN, G., KIMMEL, R., AND KIRYATI, N. 2002. Texture mapping using surface flattening via multi-dimensional scaling. *IEEE Trans. on Visualization and Computer Graphics 8*, 2, 198–207.