Dense 3D Motion Capture from Synchronized Video Streams

Yasutaka Furukawa[1]
Department of Computer Science
and Beckman Institute
University of Illinois at Urbana-Champaign, USA[1]

Jean Ponce[2,1]
Willow Team
LIENS (CNRS/ENS/INRIA UMR 8548)
Ecole Normale Supérieure, Paris, France[2]

Abstract: *This paper proposes a novel approach to nonrigid, markerless motion capture from synchronized video streams acquired by calibrated cameras. The instantaneous geometry of the observed scene is represented by a polyhedral mesh with fixed topology. The initial mesh is constructed in the first frame using the publicly available PMVS software for multi-view stereo [7]. Its deformation is captured by tracking its vertices over time, using two optimization processes at each frame: a local one using a rigid motion model in the neighborhood of each vertex, and a global one using a regularized nonrigid model for the whole mesh. Qualitative and quantitative experiments using seven real datasets show that our algorithm effectively handles complex nonrigid motions and severe occlusions.*

1. Introduction

The most popular approach to motion capture today is to attach distinctive markers to the body and/or face of an actor, and track these markers in images acquired by multiple calibrated video cameras. The marker tracks are then matched, and triangulation is used to reconstruct the corresponding position and velocity information. The accuracy of any motion capture system is limited by the temporal and spatial resolution of the cameras. In the case of marker-based technology, it is also limited by the number of markers available: Although relatively few (say, 50) markers may be sufficient to recover skeletal body configurations, thousands may be needed to accurately recover the complex changes in the fold structure of cloth during body motions [24], or model subtle facial motions and skin deformations [17, 18], a problem exacerbated by the fact that people are very good at picking unnatural motions and "wooden" expressions in animated characters. Markerless motion capture methods based on computer vision technology offer an attractive alternative, since they can (in principle) exploit the dynamic texture of the observed surfaces themselves to provide reconstructions with fine surface details[1] and dense estimates of nonrigid motion. Markerless technology using special make-up is indeed emerging in the entertainment industry [15], and several approaches to local *scene flow* estimation have also been proposed to handle less constrained settings [4, 13, 16, 19, 23]. Typically, these methods do not fully exploit global spatio-temporal consistency constraints. They have been mostly limited to relatively simple and slow motions without much occlusion, and may be susceptible to error accumulation. We propose a different approach to motion capture as a 3D tracking problem and show that it effectively overcomes these limitations.

[1]This has been demonstrated for static scenes, since, as reported in [21], modern multi-view stereo algorithms now rival laser range scanners with sub-millimeter accuracy and essentially full surface coverage from relatively few low-resolution cameras. Of course, instantaneous shape recovery is not sufficient for motion capture, since nonrigid motion cannot (easily) be recovered from a sequence of instantaneous reconstructions.

1.1. Related Work

Three-dimensional *active appearance models* (AAMs) are often used for facial motion capture [11, 14]. In this approach, parametric models encoding both facial shape and appearance are fitted to one or several image sequences. AAMs require an a priori parametric face model and are, by design, aimed at tracking relatively coarse facial motions rather than recovering fine surface detail and subtle expressions. *Active sensing* approaches to motion capture use a projected pattern to independently estimate the scene structure in each frame, then use optical flow and/or surface matches between adjacent frames to recover the three-dimensional motion field, or *scene flow* [10, 25]. Although qualitative results are impressive, these methods typically do not exploit the redundancy of the spatio-temporal information, and may be susceptible to error accumulation over time. Several *passive* approaches to scene flow computation have also been proposed [4, 13, 16, 19, 23]. Some start by estimating the optical flow in each image independently, then extract the 3D motion from the recovered flows [13, 23]. Others directly estimate both 3D shape and motion [4, 16, 19]: A variational formulation is proposed in [19], the motion being estimated in a level-set framework, and the shape being refined by the multi-view stereo component of the algorithm (see [6] for related work). A subdivision surface model is used in [16], the shape and motion of an object being initialized independently, then refined simultaneously. In contrast, visible surfaces are represented in [4] as collections of *surfels*—that is, small patches encoding shape, appearance, and motion. In this case, shape is first estimated in each frame independently by a multi-view stereo algorithm, then the 3D motion of each surfel from one frame to the next is estimated.

Existing scene flow algorithms suffer from two limitations: First, they have so far mostly been restricted to simple motions with little occlusion. Second, local motions are typically estimated independently between adjacent frames, then concatenated into long trajectories, causing accumulating drift [20] which may pose problems in applications such as body and face motion capture, or facial expression transfer from human actors to imaginary creatures [15, 2]. A strategy aptly called "track to first" in [3] solves the accumulation problem, and it is exploited in our approach (see [5, 22] for approaches free from accumulation drifts).

1.2. Problem Statement and Proposed Approach

This paper addresses motion capture from synchronized, calibrated video streams as a *3D tracking* problem, as opposed to scene flow estimation. The instantaneous geometry of the observed scene is represented by a polyhedral mesh with fixed topology. An initial mesh is constructed in the first frame using the publicly available PMVS software for multi-view stereo [7, 8], and its deformation is captured by tracking its vertices over time with two successive optimization processes at each frame: a local one using a rigid motion model in the neighborhood of each vertex, and a global one using a regularized nonrigid deformation model for the whole mesh. Erroneous motion estimates at vertices with high deformation energy are filtered out as outliers, and the optimization process is repeated without them. As demonstrated by our experiments (Sec. 4), the main contributions of this paper are in three areas:

• **Handling complex, long-range motions:** Our approach to motion capture as a 3D tracking problem allows us to handle fast, complex, and highly nonrigid motions with limited error accumulation over a large number of frames. This involves several key ingredients: (a) an effective mixture of locally rigid and globally nonrigid, regularized motion models; (b) the decomposition of the former into normal and tangential components, which allows us to use the mature machinery of multi-view stereopsis for shape estimation; and (c) a simple expansion procedure that allows us to propagate to a given vertex the shape and motion parameters inherited from its neighboring vertices.

• **Handling gross errors and heavy occlusion.** Our approach is capable of detecting and recovering from gross matching errors and tracks lost due to partial occlusion, thanks to a second set of key ingredients: (d) an effective representation of surface texture and image photoconsistency that allows us to easily spot outliers; (e) a global representation of shape by an evolving mesh that allows us to stop or restart tracking vertices as they become occluded or once again visible; and (f) an effective means for associating with a surface patch a reference frame and the corresponding texture adaptively during the sequence, which frees us from the need for a perfect initialization.

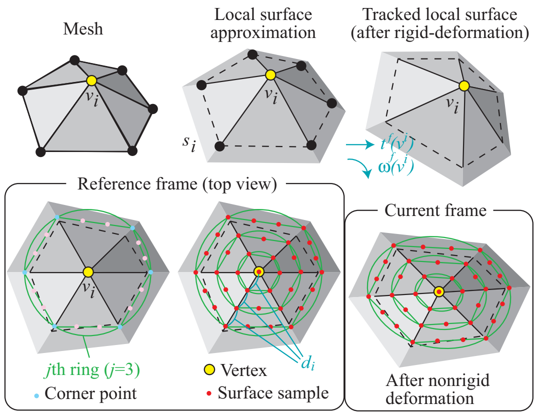
Mesh
Local surface approximation
Tracked local surface (after rigid-deformation)
v_i
s_i
$t^f(v^i)$
$\omega^f(v^i)$
Reference frame (top view)
Current frame
jth ring (j=3)
● Corner point
● Vertex
● Surface sample
d_i
After nonrigid deformation

Figure 1. Local geometric (top) and photometric (bottom) surface models. The surface region s_i associated with the vertex v_i is simply the union of the incident triangles. See text for details.

• **Quantitative validation.** This issue has been mostly ignored in scene flow research, in part because ground truth is usually not available. It is addressed in Sec. 4.

2. Spatio-Temporal Surface Model

We model the surface being tracked as a polyhedral mesh model with a fixed topology and moving vertices $v_1, \ldots, v_n$. As will become clear in the rest of this section, each vertex may or may not be tracked at a given frame, including the first one, allowing us to handle occlusion, fast motion, and parts of the surface that are not visible initially. The core computational task of our algorithm is to estimate in each frame f the position v_i^f of each vertex v_i. The rest of this section presents our local geometric and photometric models of the surface area s_i in the vicinity of v_i (Fig. 1), as well as the core tracking procedure used by our algorithm.

2.1. Local Surface Model

2.1.1 Local Geometric Model

We represent the surface in the vicinity of a vertex v_i by the union s_i of the incident triangles, and assume *local* rigid motion at each frame (the mesh *globally* moves in a nonrigid manner with the iteration of the local/global motion steps as explained in Sec. 3.2). Concretely, we attach a coordinate system to s_i with an origin at v_i and a z axis along the surface normal at v_i (the x axis is arbitrarily in the tangent plane), and represent its rigid motion by translational and rotational velocities $t^f(v_i)$ and $\omega^f(v_i)$ (Fig. 1, top).

2.1.2 Local Photometric Model

Some model of spatial texture distribution is needed to measure the *photoconsistency* of different projections of the surface region s_i. We assume that a surface is Lambertian and represent the appearance of s_i in an image by a finite number of pixel samples, which are computed as follows: We construct a discrete set of points sampled at regular intervals in concentric rings around v_i on s_i (Fig. 1, bottom). The spacing d_i between rings is chosen so images of two consecutive rings are (roughly) separated by one pixel in the image where s_i is visible with minimum foreshortening. There are τ rings around each vertex ($\tau = 4$ or 5 in all our experiments), and the ring points are sampled uniformly between *corner points* located at d_i intervals from each other along the edges incident to v_i, with $i-1$ samples per face for ring number i. Finally, each sample point is assigned the corresponding pixel value from an image by bilinear interpolation of neighboring pixel colors. Note that the sample point positions are computed as above only in the *reference frame* $\hat{f}_i$ attached to v_i (see Sec. 3 for how it is determined), and stored as barycentric coordinates in the affine coordinate systems formed by the vertices of the triangles they lie in. In all the other frames, the barycentric coordinates are used to recompute the sample positions. This provides a simple method for projecting them into new images despite nonrigid motions, and retrieving the corresponding texture patterns.

2.2. Shape and Motion Estimation

The rotational and translational velocities $\omega^f(v_i)$ and $t^f(v_i)$ representing the local rigid motion of the patch s_i can be decomposed into *normal* and *tangential* components (Fig. 2). The normal components essentially encode what amounts to shape information in the form of a "tangent plane" (the first two elements of $\omega^f(v_i)$) and a "depth" (the third element of $t^f(v_i)$) along its "normal", and their tangential components encode an in-plane rotation (the third element of $\omega^f(v_i)$) and a translational motion tangent to the surface (the first two elements of $t^f(v_i)$). Instead of estimating all six parameters at once, which is difficult for complex motions, we first estimate the normal (shape) component, then the full 3D motion.

2.2.1 Initial Motion Estimation by Expansion

Expansion strategies have recently proven extremely effective in turning a sparse set of matches into a dense one in multi-view stereo applications [8, 12]: Typically, a set of matching image patches is iteratively expanded using the spatial coherence of nearby features to *predict* the approximate position of a (yet) unmatched patch. Here, we propose to use the spatio-temporal coherence of nearby vertices to predict the motion structure of a vertex not tracked yet. Concretely, before applying the optimization procedures described below, the instantaneous motion parameters are simply initialized by taking an average of the values at the adjacent vertices that have already been tracked in the current frame. When no adjacent vertex has been tracked (yet), motion parameters are initialized by the values estimated at the vertex itself in the previous frame.

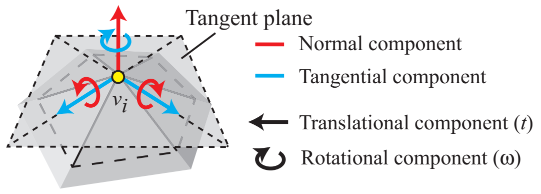
Tangent plane
v_i
Normal component
Tangential component
Translational component (t)
Rotational component (ω)

Figure 2. The *tangential* and *normal* components of a local rigid motion.

2.2.2 Shape Optimization

Optimizing the normal component of motion is very similar to optimizing depth and surface normal in multi-view stereo [8, 9]. Concretely, we maximize the sum of a *shape* photoconsistency function and a smoothness term

$$\left[\sum_{j\in V_i^f}\ \sum_{k\in V_i^f, j\neq k} \frac{N(Q_{ij}^f, Q_{ik}^f)}{|V_i^f|(|V_i^f|-1)/2}\right] - \mu_v^f|\bar{v}_i^f - v_i^f|^2/\epsilon^2 \tag{1}$$

using a conjugate gradient (CG) method. The first term simply compares sampled local textures in multiple images of the *current* frame to compute an average pairwise correlation score (Fig. 3). In this term, V_i^f denotes the set of indexes of the cameras in which v_i is visible in frame f; Q_{ij}^f is the set of sampled pixels colors for v_i in the image I_j^f acquired by camera number j; and $N(Q, Q')$ denotes the normalized cross correlation between Q and Q'. Note that Q_{ij}^f is determined by the normal components of the velocity field: This is how these parameters enter in our energy function. The second (smoothness) term prevents the vertex from moving too far from its initial position. In this term, μ_v^f is the number of nearby vertices used to initialize the motion parameters, which increases the effect of the smoothness term in the presence of many tracked neighbors, $\bar{v}_i^f$ denotes the position of the vertex at initialization, and ϵ is the average edge length in the mesh for normalization.

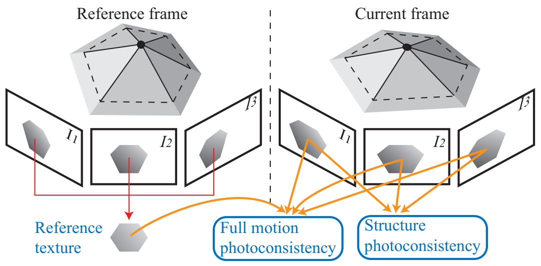

Figure 3. Shape and full motion photoconsistencies. See text for details.

2.2.3 Motion Optimization

After optimizing the normal component, the local velocity parameters are all refined by maximizing the sum of a *full motion* photoconsistency function and the same smoothness term as before:

$$\left[\sum_{j\in V_i^f}\sum_{k\in V_i^{\hat{f}_i}}\frac{N(Q_{ij}^f,Q_{ik}^{\hat{f}_i})}{|V_i^f||V_i^{\hat{f}_i}|}\right]-\mu_v^f|\overline{v}_i^f-v_i^f|^2/\epsilon^2, \qquad (2)$$

using again a CG method. Here, $\hat{f}_i$ is the reference frame of v_i (see Sec. 3 for the method used to determine it), and the first term simply compares reference textures with the image textures in the current frame (this is an example of the "track to first" [3] strategy mentioned earlier). In practice, both the shape and the full motion optimization steps are performed in a multi-scale, coarse-to-fine fashion using a three-level pyramid for each input image.

2.2.4 Visibility Estimation

The computation of the photoconsistency functions (Eqs. (1, 2)) requires the visibility information V_i^f, which is estimated as follows: We use the current mesh model to initialize V_i^f, then perform a simple photoconsistency check to filter out images containing unforeseen obstacles or occluders. Concretely, for each image in V_i^f, we compute an average normalized cross correlation score of sampled pixel colors with the remaining visible images. If the average score is below a certain threshold ψ_1, the image is filtered out as outlier. Specific values for this threshold as well as all other parameters are given in Sec. 4 (Table 1).

3. Algorithm

This section presents the three main steps –local optimization, mesh deformation, and filtering– of our tracking procedure. In practice, these steps are repeated four times at each frame to improve the accuracy of the results. See Fig. 4 at the end of this section for the overall algorithm.

3.1. Local Tracking

Let us now explain how the optimization procedures presented in Sec. 2.2 can be used to estimate the velocity of each vertex in the mesh and identify as needed the corresponding reference frame and reference texture. Vertices to be tracked are stored in a priority queue Z, where pairwise priority is determined by the following rules in order: (1) if a vertex has already been assigned a reference frame, and another one has not, the first one has higher priority; (2) the vertex with most neighbors already tracked in the current frame has higher priority; (3) the vertex with smaller translational motion in the previous frame has higher priority. At the beginning of each frame, we compute a set of visible images for each vertex as described in Sec. 2.2.4, then push onto Z all the vertices with (yet) unknown motion parameters that are visible in at least ρ images. While the queue is not empty, we pop a vertex v_i from the queue, and initialize its instantaneous motion parameters $\omega^f(v_i)$ and $t^f(v_i)$ by the expansion procedure of Sec. 2.2.1. If the vertex has already been assigned a reference frame, the shape optimization and full motion optimization (Sec. 2.2) are performed. At this point, tracking is deemed a failure if the shape photoconsistency term in Eq. (1) is below ψ_2 or the full motion photoconsistency term in Eq. (2) is below ψ_3, and a success otherwise. If the vertex has not been assigned a reference frame yet, we first compute barycentric coordinates of sample points as described in Sec. 2.1, then perform shape optimization only (the full motion optimization cannot be performed due to the lack of reference frame). At this point, if the shape photoconsistency in Eq. (1) is below ψ_2, we reject the estimated motion. Otherwise, the shape optimization is deemed a success, f becomes the reference frame $\hat{f}_i$, and the corresponding texture is computed by averaging the pixel values in Q_{ij}^f over the images j in V_i^f. In all cases, when tracking succeeds, we update the priority of the vertices adjacent to v_i and their positions in the queue.

3.2. Mesh Deformation

The local tracking step may contain erroneous motion estimates due to its rather greedy approach and the lack of regularization. Therefore, instead of just moving each vertex independently according to the estimated motion, we deform the mesh as a whole by minimizing an energy function that is a weighted sum of *data-attachment*, *smoothness*, and *local rigidity* terms over all the vertices:

$$\sum_i |v_i^f-\hat{v}_i^f|^2+\eta_1|[\zeta_2\Delta^2-\zeta_1\Delta]v_i^f|^2+\eta_2[\epsilon(v_i^f)-\epsilon(v_i^{\hat{f}_i})]^2.$$

The first (data-attachment) term simply measures the deviation between the actual position v_i^f of v_i in frame f and the position $\hat{v}_i^f$ predicted by the local optimization process. The second term uses the (discrete) Laplacian operator Δ

```
Input:  Vertices $v_i^{f-1}$ in the previous frame.
Output: Vertices $v_i^f$ in the current frame.

Repeat four times
  Update $V_i^f$ for each vertex $v_i$ (Sec. 2.2.4).
  Push vertices with unknown motion parameters that
  are visible in at least $\rho$ images onto a queue $Z$.
  While $Z$ is not empty
    Pop a vertex $v_i$ from $Z$.
    If $v_i$ does not have a reference texture
      Perform the shape optimization (Sec. 2.2.2).
      If the optimization succeeds
        Remember the reference texture and sampling points.
        Update priorities of its adjacent vertices in $Z$.
    else
      Perform the shape optimization (Sec. 2.2.2).
      Perform the full motion optimization (Sec. 2.2.3).
      If the optimization succeeds
        Update priorities of its adjacent vertices in $Z$.
  Deform the mesh by estimated motions (Sec. 3.2).
  Filter out erroneous motion estimates (Sec. 3.3).
  Deform the mesh without the erroneous motions (Sec. 3.2).
```

Figure 4. Overall tracking algorithm.

of a local parameterization of the surface in v_i to enforce smoothness ($\zeta_1 = 0.6$ and $\zeta_2 = 0.4$ in all our experiments) [8]. The third (local rigidity) term prevents too much stretching or shrinking of the surface in the neighborhood of v_i by measuring the discrepancy between the mean $\epsilon(v_i^f)$ of the edge lengths around v_i in frame f and its counterpart $\epsilon(v_i^{\hat{f}_i})$ in the reference frame $\hat{f}_i$. The total energy is minimized with respect to the 3D positions of all the vertices again by a CG method. Note that the data-attachment term is used only for vertices that have been successfully tracked.

3.3. Filtering

After surface deformation, we use the residuals $r_d(v_i)$ and $r_l(v_i)$ of the the data-attachment and local rigidity terms to filter out erroneous motion estimates.[2] Concretely, we smooth the values of $r_d(v_i)$ and $r_l(v_i)$ at each vertex by replacing each of them by its average over v_i and its neighbors, which process is repeated ten times. After smoothing, a motion estimate is detected as an outlier if $r_d(v_i)$ is more than $\epsilon^2(v_i^{\hat{f}_i})$ or $r_l(v_i)$ is more than $\epsilon^2(v_i^{\hat{f}_i})/4$. Having filtered out the erroneous motions, the mesh is deformed again. We decrease the two regularization parameters η_1 and η_2 by a factor of 4 after the filtering, since the main purpose of the first deformation is to act as a filter, while the second one is used to estimate an accurate surface model.

[2]The smoothness residual is not used for filtering, since we want to keep sharp features of the mesh. On the other hand, we want to avoid too much stretching or shrinking for materials such as cloth.

4. Experimental Results and Discussion

• **Implementation and datasets.** The proposed algorithm has been implemented in C++. A 3D mesh model for each dataset is obtained in the first frame by using the publicly available PMVS software [7] that implements [8], one of the best multi-view stereo algorithms to date according to the Middlebury benchmarks [21]. This program outputs a set of *oriented points* (points plus normals), then fits a closed polyhedral mesh over these points, deforming it under the influence of photoconsistency and regularization energy terms. The resulting mesh smoothly extrapolates the reconstructed data in places occluded from the cameras, an important point in practice, since it allows us to start tracking the (extrapolated) vertices when the corresponding surface area becomes visible. Seven real datasets are used for the experiments (Fig. 5): *flag*, *shirt*, *neck* (courtesy of R.L. Carceroni and K. Kutulakos [4]); *face1* (courtesy of P. Baker and J. Neumann [1]); *pants1*, *pants2* (courtesy of R. White, K. Crane and D.A. Forsyth [24]), and *face2* (courtesy of ImageMoversDigital). The characteristics of these datasets and the parameter values used in our experiments are given in Table. 1. The motions in *neck* and *face1* are very slow, but the textures are weak compared to the other datasets, and the mouth and eye motions in *face1* are challenging. Motions are fast in *flag* and *shirt*, but still relatively simple. On the other hand, *pants1* and *pants2*, although heavily textured, are quite challenging datasets involving fast and complex motions of cloth and its folds, with occlusions in various parts of the videos: In *pants1*, the actor picks up the cloth with his hands, causing severe occlusions and, in *pants2*, he dances very fast, yielding very complex motion and severe self-occlusions due to cloth folding, with image velocities greater than twenty pixels per frame in some image regions. Motions are relatively slow for *face2* throughout the sequence, but occasionally become very fast when the actress speaks. For *shirt* and *pants1*, we have reversed the original sequences: the motion was too fast otherwise at the beginning of the *shirt* sequence for tracking to succeed. The *pants1* sequence has been reversed in order not to track the hand and arm of the actor, that occlude large portions of the pants in the first frame.

• **Qualitative motion capture experiments.** Figure 5 shows, for each dataset, a sample input image from one frame in the sequence, the corresponding mesh with and without texture mapping, and the estimated motion field, rendered by line segments connecting the positions of sample vertices in the previous frame (red) to the current ones (green). Textures are mapped onto the mesh by averaging the back-projected textures from every visible image in every tracked frame. This is a good way to visually assess the quality of the results, since textures will only look sharp and clear when the estimated shape and motion parameters are accurate throughout the sequence. As shown by the fig-

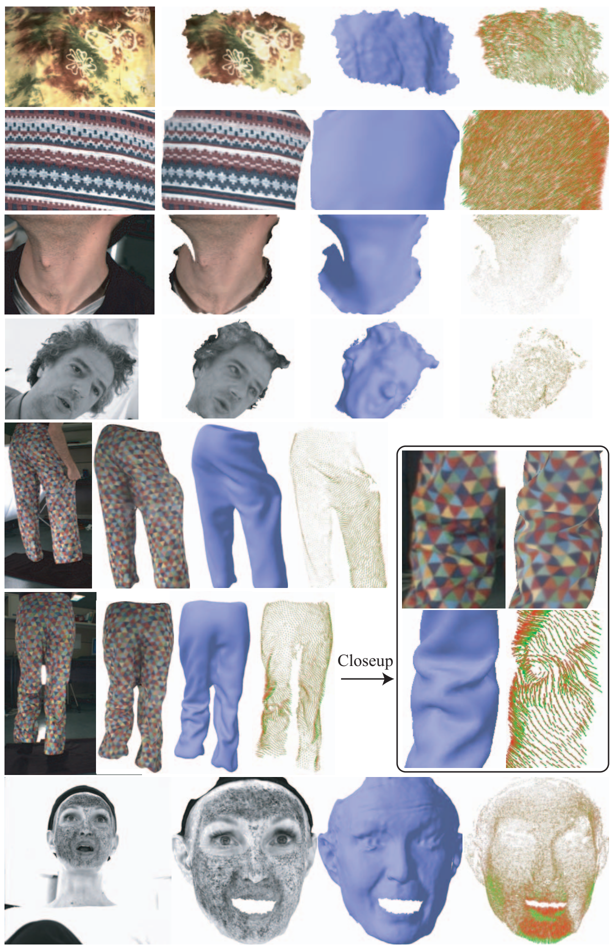

Figure 5. From left to right, and for each dataset: an input image, a tracked mesh with and without texture-mapping, and the corresponding motion field. In the close-ups of *pants2*, our texture-mapped model is indeed very close to the corresponding input image, but there are moderate discrepancies in some places, in particular in the middle of the complex fold structure where a surface region not visible by a sufficient number of cameras has not been tracked.

ure, this is indeed the case in our experiments, with sharp images looking very close to the originals. Of course, there are discrepancies in some places. The eyes of *face1* and *face2* provide an example, with a motion different from the other parts of the face and strongly conflicting with our local rigidity term $r_l(v_i)$. A second example is given for *pants2* (see closeups of Fig. 5), corresponding to a case where part of the fold structure of the cloth is not clearly visible by several of the eight cameras. Overall however, our algorithm has been able to accurately capture the cloth's very complicated shape and motion. Given the absence of ground truth data, it is difficult to compare our results to other experiments on the same datasets. The most obvious difference is that our method captures much denser information than [4, 24] for the *pants*, *flag*, *shirt*, and *neck* datasets. Indeed White *et al.* only track the vertices (about 2400 to-

Table 1. Top: Characteristics of the seven datasets: N, F and M are the numbers of cameras, frames and vertices on the mesh; w and h are the width and the height of input images in pixels; and s is the approximate size in pixels of the projection of mesh edges in frontal views. Bottom: Parameter values for our algorithm: (η_1, η_2) are the regularization parameters for the first deformation step (they are 4 times smaller after the filtering); (ψ_1, ψ_2, ψ_3) are thresholds on photoconsistency functions; and ρ is the minimum number of images in which a vertex has to be visible.

	flag	*shirt*	*neck*	*face1*	*pants1*	*pants2*	*face2*
N	7	7	7	22	8	8	10
F	37	12	69	90	100	155	325
M	4828	10347	5593	9035	8652	8652	39612
w	722	722	722	644	480	480	1000
h	482	482	482	484	640	640	1002
s	10	6	6	10	6	6	3
η_1	16	16	32	80	20	20	20
η_2	8	200	64	32	40	40	40
ψ_1	0.3	0.3	0.3	0.3	0.1	0.1	0.3
ψ_2	0.5	0.5	0.5	0.5	0.3	0.3	0.5
ψ_3	0.4	0.4	0.4	0.4	0.2	0.2	0.4
ρ	3	3	3	3	2	2	3

tal) of the triangular pattern printed on the pants in [24], as opposed to the 7000 mesh vertices or so that we track throughout the two *pants* sequences (see Fig. 6 for more details), without of course exploiting the known structure of the triangular pattern in our case. Likewise, Carceroni and Kutulakos track about 120 *surfels* for *neck*, and 200 for *shirt* and *flag* in [4], whereas the number of tracked vertices varies from about 4000 to 8000 for our method.

• **Qualitative evaluation of different key components of the proposed algorithm.** Two experiments have been used for this evaluation. First, we have run the proposed algorithm without the expansion procedure on *pants2* (Fig. 6, left) simply copying motion parameters estimated at the previous frame instead of interpolating the motion of nearby vertices already tracked. As shown in Fig. 6, the cloth motion cannot be captured in frame 124 without the expansion procedure. Our second experiment assesses the contribution of the proposed motion decomposition. This time, we have run our algorithm by directly applying the full motion optimization step without shape optimization. The right half of Fig. 6 shows that tracking without the decomposition fails in recovering details at the back side of both legs. One interesting observation regarding these two experiments is that tracking fails in frame 124 without the expansion scheme, and in frame 137 without motion decomposition, but the algorithm quickly recovers and recaptures the correct shape and motion in frames 132 and 147 of the two sequences (Fig. 6). Thus, even when our basic tracking procedure (local optimization, mesh deformation, and filtering) fails locally in certain frames due to overly complex or fast motions, it is capable of recovering from gross errors, *even* when deprived of two key ingredients that further enhance its robustness. This is a very appealing property for motion

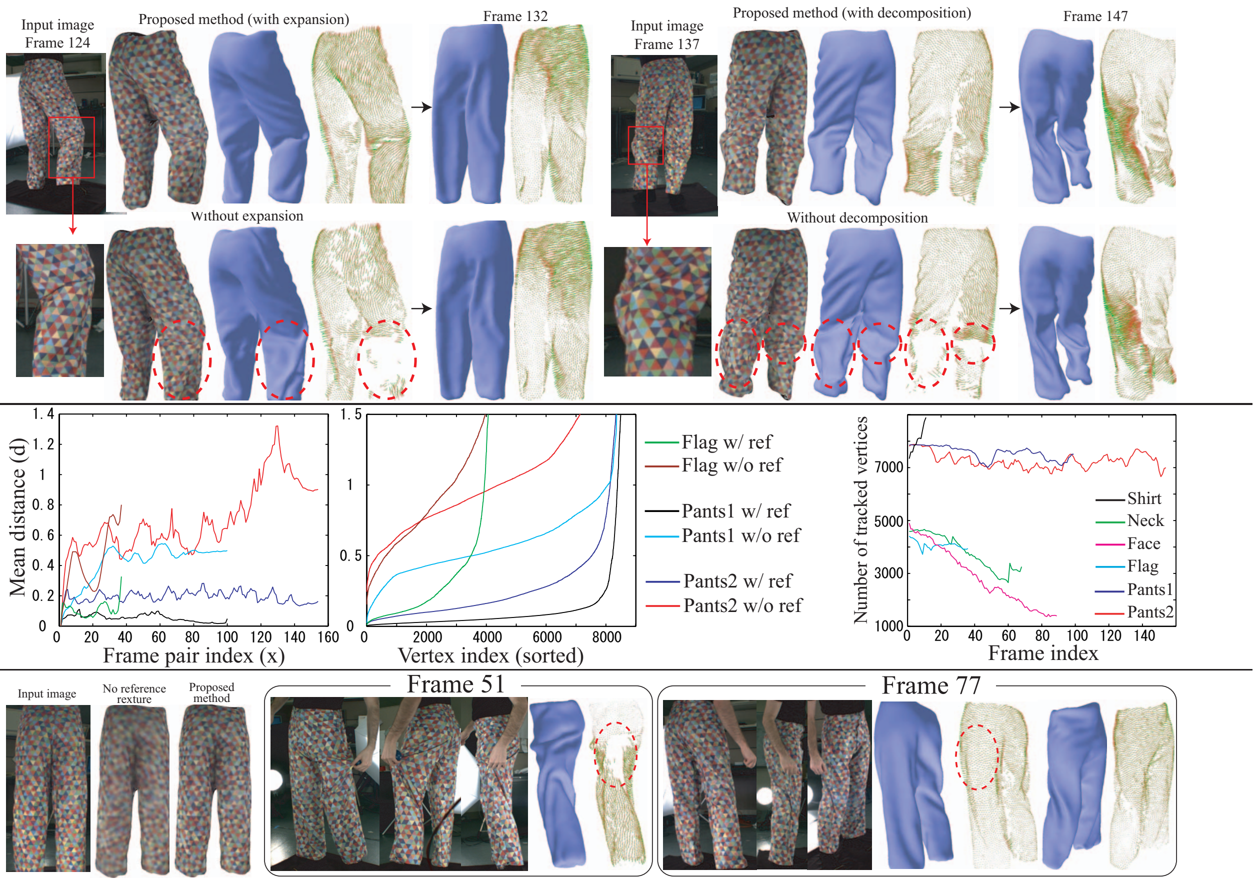

Figure 6. Top: results (left) with/without expansion and (right) with/without instantaneous motion decomposition. Center: (left) accumulated errors; (right) number of vertices that have been successfully tracked in each frame. Bottom: (left) qualitative assessment of drifting effects; (right) occlusion handling.

capture in practice, because one needs not reinitialize the model every time the tracker fails, and users can just work later on frames where automatic tracking is difficult.

• **Quantitative experiments.** Our last experiments demonstrate the robustness of the proposed method against drift (accumulating errors) and occlusion. Let us first show how our "track to first" strategy limits drift. We have chosen the *flag*, *pants1*, and *pants2* sequences for this experiment, since the corresponding motions are relatively complex. We run the proposed algorithm with and without using a reference frame, updating the reference texture in every frame when a vertex is tracked successfully in the latter case (this resembles the approach followed by most scene flow algorithms). In order to quantitatively measure accuracy, we have appended to each sequence of F frames in the three datasets its reversed copy (without its first frame) to form a new sequence consisting of $2F-1$ frames. Images at frames $F-x$ and $F+x$ are the same, hence the corresponding two meshes should be very close to each other (see [20] for similar experiments for assessing drift in 2D tracking). Let d denote the distance between the positions of the same vertex in frames $F-x$ and $F+x$, divided by the mean edge length of the mesh for normalization. The leftmost graph in the center panel of Fig. 6 plots, for each frame and each dataset, the value of d averaged over all vertices with and without the use of a reference frame.[3] As shown by this figure, the mean distance is consistently three to five times larger for each dataset when reference frames are not used. The value of d for frames 1 and $2F-1$ ($x = F-1$) is plotted for every vertex in the next graph (the vertices being sorted in an increasing order of the values of d), showing a similar contrast between the two variants for long-term drift. The added value of reference frames is also (qualitatively) clear from the texture-mapped models for *pants2* shown in the left of the bottom panel of Fig. 6, where texture is blurred for the model not using reference frames.

The rightmost graph of the center panel in Fig. 6 shows the number of vertices that have been successfully tracked in each frame. The number keeps decreasing for *face1*, be-

[3] We only retain the "best" vertices with the smallest distances to construct this graph. This is to exclude "outliers" such as vertices that do not correspond to actual parts of the surface (e.g., the top and bottom portions of the mesh in the pants sequences). In practice, 30% and 20% of the vertices in the *flag* and the *pants* sequences are excluded, respectively. See the next graph for the full distance distribution including "outliers".

cause a large surface region faces away from most cameras in the middle of the sequence. On the other hand, the number keeps increasing for *shirt* as the cloth moves away from the camera and more surface regions become visible, which illustrates the fact that our method is able to start tracking new vertices in surface areas that have been extrapolated by PMVS but are not visible from the cameras in the first frame with the topology of the mesh being fixed.[4] Finally, the right side of the bottom panel of Fig. 6 shows how occlusions are handled by our algorithm for the *pants1* dataset. In frame 51, vertices at the left side of the pants are not tracked due to the severe occlusions caused by a hand, but our algorithm restarts tracking these vertices once they become visible again as shown in our results for frame 77. Note that the right side of the pants is also occluded by the actor's right hand in frame 77, but it is visible from two other cameras, and thus has been successfully tracked by our algorithm.

Let us conclude with a few remarks. The running time of the proposed method depends on the dataset, the mesh resolution, and the number of input images, but it takes about one to two minutes per frame on a dual Xeon 3.2 GHz PC. It should be possible to speed up the whole computation quite a bit, for example by replacing the numerical derivatives currently used by our conjugate gradient implementation by analytical ones. Other improvements are part of our plans: It is important to analyze the relative contributions of the key components of our algorithm more thoroughly to reduce the amount of redundant computations. It also seems wasteful to compute angular velocities during local tracking, then discard them during global surface deformation, so we will seek more effective uses for this local information. More importantly perhaps, our current approach to the appearance of new surface regions over time is somewhat ad hoc, relying on PMVS to extrapolate the mesh in regions that are not matched in the first frame without allowing the topology of a mesh to change. A key part of our future work will be to address this problem in a more principled fashion.

Acknowledgments: This paper was supported in part by the National Science Foundation under grant IIS-0535152, the INRIA associated team Thetys, and the Agence Nationale de la Recherch under grants Hfimbr and Triangles. We thank R.L. Carceroni and K. Kutulakos for *flag*, *shirt* and *neck* data sets, and thank P. Baker and J. Neumann for *face1* data set. We also thank Hiromi Ono, Doug Epps and ImageMoversDigital for *face2* data set.

References

[1] P. Baker and J. Neumann. 3D-photography challenge (http://www.3d-photography.org).

[2] B. Bickel, M. Botsch, R. Angst, W. Matusik, M. Otaduy, H. Pfister, and M. Gross. Multi-scale capture of facial geometry and motion. In *SIGGRAPH*, 2007.

[3] A. Buchanan and A. Fitzgibbon. Interactive feature tracking using k-d trees and dynamic programming. In *CVPR*, 2006.

[4] R. L. Carceroni and K. N. Kutulakos. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3D motion, shape and reflectance. *IJCV*, 49, 2002.

[5] E. de Aguiar, C. Theobalt, C. Stoll, and H.-P. Seidel. Markerless deformable mesh tracking for human shape and motion capture. In *CVPR*, 2007.

[6] F. D. Frederic Huguet. A variational method for scene flow estimation from stereo sequences. In *ICCV*, 2007.

[7] Y. Furukawa and J. Ponce. PMVS (http://www-cvr.ai.uiuc.edu/∼yfurukaw/research/pmvs).

[8] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *CVPR*, 2007.

[9] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *ICCV*, 2007.

[10] C. Hernández Esteban, G. Vogiatzis, G. Brostow, B. Stenger, and R. Cipolla. Non-rigid photometric stereo with colored lights. In *ICCV*, 2007.

[11] S. C. Koterba, S. Baker, I. Matthews, C. Hu, J. Xiao, J. Cohn, and T. Kanade. Multi-view AAM fitting and camera calibration. In *ICCV*, 2005.

[12] M. Lhuillier and L. Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *PAMI*, 27, 2005.

[13] R. Li and S. Sclaroff. Multi-scale 3D scene flow from binocular stereo sequences. In *WACV/MOTION*, 2005.

[14] I. Matthews and S. Baker. Active appearance models revisited. *IJCV*, 60(2), 2004.

[15] Mova. LLC. Mova contour reality capture.

[16] J. Neumann and Y. Aloimonos. Spatio-temporal stereo using multi-resolution subdivision surfaces. *IJCV*, 47, 2002.

[17] M. Odisio and G. Bailly. Shape and appearance models of talking faces for model-based tracking. In *AMFG*, 2003.

[18] S. I. Park and J. K. Hodgins. Capturing and animating skin deformation in human motion. *ACM ToG*, 25(3), 2006.

[19] J.-P. Pons, R. Keriven, and O. Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *IJCV*, 72(2), 2007.

[20] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. In *CVPR*, 2006.

[21] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *CVPR*, 2006.

[22] J. Starck and A. Hilton. Correspondence labelling for wide-timeframe free-form surface matching. In *ICCV*, 2007.

[23] S. Vedula, S. Baker, and T. Kanade. Image-based spatio-temporal modeling and view interpolation of dynamic events. *ACM ToG*, 24(2), 2005.

[24] R. White, K. Crane, and D. Forsyth. Capturing and animating occluded cloth. In *SIGGRAPH*, 2007.

[25] L. Zhang, N. Snavely, B. Curless, and S. M. Seitz. Spacetime faces: high resolution capture for modeling and animation. *ACM ToG*, 23(3), 2004.

[4] Of course, a portion of the surface completely hidden from all cameras in the first frame cannot be reconstructed by the multi-view stereo at the beginning for *shirt*, but a surface model larger than the visible portion can be output by interpolation and regularization.