Seeing Skin in Reduced Coordinates

Debanga R. Neog, Anurag Ranjan, and Dinesh K. Pai
Sensorimotor Systems Laboratory, University of British Columbia, Vancouver, Canada

***Abstract*—We present a skin tracking and reconstruction method that uses a monocular camera and a depth sensor to recover skin sliding motions on the surface of a deforming object. Such depth cameras are widely available. Our key idea is to use a reduced coordinate framework that implicitly constrains skin to conform to the shape of the underlying object when it slides. The skin configuration in 3D can then be efficiently reconstructed by tracking two dimensional skin features in video. This representation is well suited for tracking subtle skin movements in the upper face and on the hand. The reconstructed skin motions have many uses, including synthesizing and retargeting animations, recognizing facial expressions, and for learning data-driven models of skin movement. In our face tracking examples, we recover subtle but important details of skin movement around the eyes. We validated the algorithm using a hand gesture sequence with known skin motion, recovering skin sliding motion with a low reconstruction error.**

I. Introduction

Technology for reconstructing and tracking 3D shapes is now widely available, especially due to the availability of inexpensive sensors such as Microsoft's Kinect and Intel's RealSense cameras. In conjunction with template-based motion tracking [1], [2], [3], [4], one can generate a sequence of 3D meshes that represent the shape of the body. However, these mesh animations do not accurately capture the motion of skin, since skin can slide over the body without changing the body's shape. For example the skin on the face and hands can stretch, wrinkle, and slide during natural movements without being detected by a depth sensor. Here we propose a new method for capturing the sliding motion of skin over a body using the color video information that is usually available in addition to depth.

The key observation is that the skin and muscles of the face, especially around the eyes [5], and on the back of the hand, are very thin and sheet-like. In such regions, skin can be well approximated as a thin sheet sliding on the surface of an underlying rigid or articulated rigid body structure, which we call *body* [6]. This approximation allows to represent skin in a low dimensional space and implicitly constrain it to always slide on the surface of the body. This is the core idea of our proposed method, which allows it to efficiently reconstruct subtle skin sliding motions. Such motions are small but highly noticeable, especially in the face. Capturing such skin movements from human subjects can enable the construction of data driven models of the face [7].

Facial expressions are one of the key components of effective communication. Emotions such as anger, happiness, or sadness are accompanied by characteristic facial tissue deformations. Eye movements are particularly important for non-verbal communication; in addition to changes in gaze, the configuration of the skin around the eyes convey significant information about our mental and physical states that can be recognized even from single images [8]. For example, we widen our eyes when we are surprised or droop the eyelids when we are fatigued. Similarly, when we observe a scene or a painting our eyelids produce a wide range of deformations which are highly correlated to gaze.

Surprisingly, there is very little research that focuses on tracking and reconstructing skin in the eye region. Recent work by Bermano, et al. [9] tracked eyelid motions and skin deformation around the eyes using a complex multiple camera setup, but reconstructed only simple motions such as blinking. On the other hand, in some other work eyelid margins are tracked but detailed skin deformation around the eyes are ignored [10], [11], [12]. Garrido, et al. [13] estimated 3D structure of face and refine the shape using photometric and optical flow constraints using a monocular setup, but ignore the fine reconstruction of eyelid motions. Currently, to our knowledge, no method exists to track and reconstruct skin sliding deformation around the eyes using a simple monocular camera setup.

Recovering 3D shape from monocular capture is an ill-posed problem and several constraints are imposed to limit the possible range of solutions. Two widely known techniques to tackle this problem are non-rigid structure from motion [14], [15] and shape from template [13], [16]. The Non-rigid Structure from Motion techniques (*NRSfM*) [14], [17], [18] are used to recover non-rigid structures from a sequence of images that captures motion of the object. The *NRSfM* offers a model-free formulation but it usually requires correspondences in a long image sequence. On the other hand, shape from template techniques use the constraints imposed by isometric or conformal deformations to reconstruct 3D shape [13], [16], [19], [20].

Our work is closer to shape from template techniques such as the work of Garrido, et al. [13]. Our proposed reduced coordinate representation of skin allows recovery of skin sliding motion in the eye region using a monocular image sequence. This framework is general and not limited to reconstructing facial skin sliding around the eyes. It can be used to reconstruct skin sliding motions on any deforming body, as long as skin shares the shape of the body when it slides. For example, we can model skin deformation on hands when we perform hand gestures. By simply flexing our fingers we can observe how skin slides on the dorsal sides of hand. It requires a lot of manual effort to produce such sliding motions using traditional skinning techniques. On the other hand, vision based approaches of hand tracking that do not require any wearables [21], [2], [3] can reconstruct hand geometries during hand gestures in real-time but they lack detail skin sliding motions. Our proposed method complements, rather than to compete with these techniques, and generate detail skin sliding motions from already available data from these tracking systems.

Contributions. Our main contribution is the use of a reduced coordinate representation of skin to track and reconstruct skin sliding during facial expressions and hand gestures. This representation makes our system efficient and robust. Our method complements existing face and gesture tracking techniques by recovering characteristic skin sliding motions from a sequence of images. The reduced coordinate representation automatically constrains skin to slide on the tracked surface. Furthermore, it is easy to use, with minimal setup. It can utilize widely available RGB-D cameras and can use any optical flow technique. Our algorithm can correct two types of errors: first, *tracking drift* generated by the optical flow technique, and second, *3D reconstruction error* due to error in the mappings of reduced coordinate representation to 3D. Although our system uses some existing techniques as a part of its pipeline, they are customizable and users can choose their own favorites.

The remainder of the paper is organized as follows. We describe the reduced coordinate representation in Sec. II-A and tracking details in Sec. II-B. The results are presented in Sec. III, while the limitations of our method are mentioned in Sec. III-C. Finally we conclude our paper in Sec. IV.

II. Methods

A. Representing Skin Motion in Reduced Coordinates

To represent the sliding motion of skin over a deforming surface and its measurement by a video camera, we use the reduced coordinate representation of skin introduced by Li, et al. [6]; however, we discretize the skin instead of the body. See Fig.1. Our reduced coordinate system has several benefits that makes it efficient and robust: First, by representing three dimensional skin in a two-dimensional space we can efficiently compute skin configuration. Second, by constraining the synthesized skin movement to always slide tangentially on underlying body, our skin reconstruction is robust against bulging and shrinking and other interpolation artifacts.

Since skin is a thin structure that slides on an underlying body, we need to represent the skin and body separately. We will assume that the character is discretized into a triangular mesh in a *reference* pose in 3D. The skin and body meshes are aligned in the reference pose (top row, Fig.1). The skin is parameterized by a map π, using an atlas of rectangular coordinate charts.

Following the notation of Li, et al. [6], skin points are denoted $\mathbb{X}$ in 3D and $\mathbb{u}$ in 2D chart coordinates. By a small but common abuse of notation, we will use the same symbols to denote the set of points corresponding to vertices of the skin mesh, represented as stacked vectors.

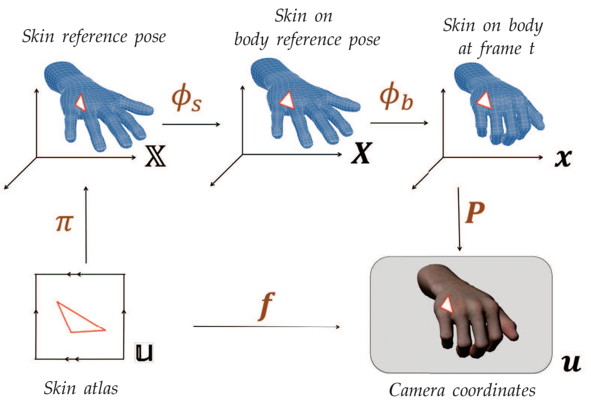

Fig. 1. Overview of spaces related to skin tracking.

They are related as:

$$\mathbb{X} = \pi(\mathbb{u}). \tag{1}$$

Each chart may be associated with a texture image $\mathbb{I}$. Such meshes, parameterization, and textures can all be obtained from standard RGDB scanning and mesh registration techniques (in our examples we used FaceShift [1]).

As the person moves in 3D, we assume they are imaged with some RGB-D system which produces a body mesh $\tilde{\boldsymbol{x}}_t$ at frame t. They are usually produced by some variant of the ICP algorithm with point cloud data. However, the key point is that these meshes capture the body shape but not the sliding of the skin over the body. The true skin mesh, $\boldsymbol{x}_t$, is no longer aligned with the body mesh $\tilde{\boldsymbol{x}}_t$. Our goal is to reconstruct $\boldsymbol{x}_t$.

To this end, the skin is imaged by a camera, with associated projection matrix P, to produce a color image I_t. The image coordinate of the skin vertex $\boldsymbol{x}_t$ is denoted $\boldsymbol{u}_t$. Therefore, we can write:

$$\boldsymbol{u}_t = P(\boldsymbol{x}_t). \tag{2}$$

Note that since we have depth information from the body mesh $\tilde{\boldsymbol{x}}_t$, P is essentially a 3D projective transformation (and not a projection) and therefore invertible. This inverse is called the *body map* M_t, and it maps a skin point in the camera coordinates on to the body surface.

Finally, note that once we know the locations $\boldsymbol{u}$ of mesh vertices in an image, we can define a function $\boldsymbol{f}$ in the visible regions of skin, from the atlas to image, by interpolating the values between vertices. This function can be used to warp or transfer pixels from the texture atlas images $\mathbb{I}$ to the camera image.

B. Reduced Coordinate Tracking

Our reduced coordinate skin tracking and reconstruction algorithm is summarized in Algorithm 1. Here we describe different components of this algorithm in greater detail.

Algorithm 1: Skin Tracking in Reduced Coordinates

```
Input  : Reference mesh $\mathbb{X}$, with a set of reference
         texture images $\mathbb{I}$; projection matrix $P$; for
         $t = 1 : T$, a sequence of body meshes $\tilde{\boldsymbol{x}}_t$ and
         camera images $I_t$
Output: Sequence of skin configuration, $\boldsymbol{x}_t$
1  Initialization: Set $\boldsymbol{u}_0$ and $I_0$, using $\mathbb{X}$ and $\mathbb{I}$
2  for $t = 1$ to $T$ do
3      Generate: body map $M_t$
4      Compute: dense optical flow $\boldsymbol{w_p}$ from $I_{t-1}$ to $I_t$
5      Using correspondences between $\boldsymbol{u}_0$ and $\boldsymbol{u}_t^*$, where
         $\boldsymbol{u}_t^* = \boldsymbol{u}_{t-1} + \boldsymbol{w_p}$, warp $\mathbb{I}$ with $f$ to generate $I_t^*$
6      Compute: corrective dense optical flow $\boldsymbol{w_c}$ from $I_t$ to
         $I_t^*$ to remove drift in flow
7      Update flow: $\boldsymbol{w} = \boldsymbol{w_p} + \boldsymbol{w_c}$                // Eq. 3
8      Advect: $\boldsymbol{u}_t = \boldsymbol{u}_{t-1} + \boldsymbol{w}$                 // Eq. 4
9      Optimize: to improve 3D reconstruction         // Eq. 5
         $\boldsymbol{x}_t = \arg\min_x(\|P\boldsymbol{x} - \boldsymbol{u}_t\|^2 + \lambda\,\|\boldsymbol{x} - M_t(\boldsymbol{u}_t)\|^2)$
10 end
11 return $\boldsymbol{x_t}$
```

1) Inputs: Our system requires a sequence of 3D meshes registered to the motion of a object. We call this sequence a 'body sequence' (or $\tilde{\boldsymbol{x}}_t, t = 1 : T$). The body sequence can be obtained by any mesh tracking technique (e.g., [2], [3]). For our face tracking example (Sec. III-A), we used FaceShift [1] to generate the body sequence. Any calibrated monocular camera can used to produce the image sequence.

2) Initialization: To bootstrap the flow computations, we need $\boldsymbol{u}_0$ and I_0. We first estimate the initial body mesh $\tilde{\boldsymbol{x}}_0$. In many cases, it is the same as the reference mesh $\mathbb{X}$; if not, a registration step is performed to align the two. Then we project $\tilde{\boldsymbol{x}}_0$ using P to obtain $\boldsymbol{u}_0$ in image coordinates. Since the vertex coordinates for each mesh triangle are known in both image coordinates and in the texture images $\mathbb{I}$, we can synthesize I_0 by warping each triangle's pixels from $\mathbb{I}$ to I_0.

3) Flow Computation and Correction: We use a dense tracker proposed by Brox and Malik [22] to estimate the flow w_p between I_{t-1} and I_t. We advect the tracked point locations $\boldsymbol{u}_{t-1}$ as: $\boldsymbol{u}_t^* = \boldsymbol{u}_{t-1} + \boldsymbol{w_p}$. The error in optical flow produces drift in the location of the tracked points with time. For longer sequences, the error quickly grows over frames. Therefore, to correct this drift we warp the texture images $\mathbb{I}$ to the frame I_t using Barycentric interpolation based on the locations of the tracked points (i.e., the function $\boldsymbol{f}$ in Fig.1) to produce I_t^*. We compute a new dense flow $\boldsymbol{w_c}$ from I_t to I_t^*. Then we correct the flow to obtain the final flow $\boldsymbol{w}$ as follows:

$$\boldsymbol{w} = \boldsymbol{w_p} + \boldsymbol{w_c}. \tag{3}$$

Using $\boldsymbol{w}$ we get the final locations of the tracked points by advecting $\boldsymbol{u}_{t-1}$:

$$\boldsymbol{u}_t = \boldsymbol{u}_{t-1} + \boldsymbol{w}. \tag{4}$$

Our algorithm can accommodate both feature-based or dense optical flow techniques. We used Large Displacement Optical Flow in our examples which is a dense flow technique and produces considerably accurate optical flow although computationally expensive. This in turn produces very low reconstruction errors. On the other hand, we also experimented with fast but less accurate Kanade-Lucas-Tomasi (KLT) feature tracker, a sparse optical flow technique. The results are documented in Sec. III.

4) Generating Body Map: As we discussed earlier, we use a body map M_t to reconstruct 3D skin in the physical space. To generate this map at t, we project the body mesh $\tilde{\boldsymbol{x}}_t$ on the image I_t to obtain $\tilde{\boldsymbol{u}}_t$. This provides the inverse at mesh points. We use Matlab's implementation of natural neighbor interpolation [23] to generate M_t using this points. Using this mapping, we can estimate 3D location corresponding to other query points $\tilde{\boldsymbol{u}}$. Instead of using natural neighbor interpolation we can also use linear interpolation, which is faster but results in high interpolation error.

5) 3D Reconstruction.: Now, using M_t we could, in theory, reconstruct the 3D skin position as $\boldsymbol{x}_t = M_t(\boldsymbol{u}_t)$. However this did not give the best results. Recall that the real skin mesh is not aligned with the body mesh. So the reconstructed skin point may not lie exactly on the body surface. To correct this, we also reproject reconstructed skin points onto the image and try to keep their locations in image coordinates close to that of the corresponding tracked skin points. This is implemented as an optimization that weights the two terms:

$$\boldsymbol{x}_t = \arg\min_x(\|P\boldsymbol{x} - \boldsymbol{u}_t\|^2 + \lambda\,\|\boldsymbol{x} - M_t(\boldsymbol{u}_t)\|^2). \tag{5}$$

The first term corresponds to minimizing the reprojection error, while the second term keeps the 3D reconstructed point close to the approximated body surface. We solve the optimization problem using a nonlinear Quasi-Newton solver in Matlab. For a given tracking system, we estimated λ by cross validating across a few set of examples. This value of λ is subsequently used for other data obtained from the same tracking system, and we obtained similar high quality results. For both face tracking and hand tracking examples we estimated $\lambda = 0.1$.

III. Results

We tested the results using two examples: first, we track and reconstruct characteristic skin sliding around the eyes, and second, we reconstruct sliding motions of skin on the hand in synthetic hand gesture sequences. The second example also contains ground truth data of skin sliding and we use that to validate our method. The code is written in Matlab 2015b (MathWorks, Inc.) and C++ on a desktop with an Intel Core i7 processor and 64GB of RAM. See our supplementary video at `https://youtu.be/TqO6eAtmqH8`.

A. Face Tracking

The face tracking example shows how using a simple monocular camera setup we can recover detail motions of skin around the eyes. Here we briefly describe the experimental setup. The setup is shown in Fig. 2(a). We used a single

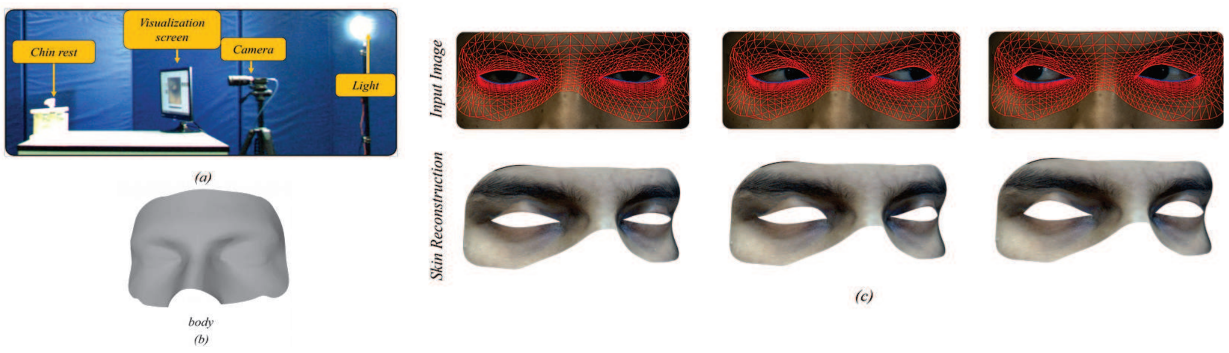

Fig. 2. We use a monocular capture setup (a) to capture subjects. Using a body (b) and skin tracked in an image sequence (c:top). These information along with the input image, we can reconstruct 3D skin (c:bottom). The whole sequence assumes a fixed body. See the video for the complete sequence.

Grasshopper3[1] camera, that can capture up to 120 fps with image resolution of 1960×1200 pixels. The actor sits on a chair and faces the camera with the head rested on a chin rest. The scene was lit by a DC powered LED light source[2] to overcome the flickering due to aliasing effects of an AC light source on a high frame rate capture. We used polarizing filters with the cameras to reduce specularity. We calibrated the camera using Matlab's Computer Vision System Toolbox. We used FaceShift [1] technology with a Kinect RGB-D camera to obtain the body mesh. This process took less than 15 minutes per actor.

For faces, a single chart and texture image is sufficient, and matches the common practice. In our algorithm we use dense flow to track skin features in image but tracking the eyelid margins is challenging because of occlusions that occur due to eyelashes and eyelid folds. Therefore, we tracked eyelid margins separately using an artificial neural network (ANN). We use a feed forward network (using Matlab `fitnet`) with 5 neurons in the hidden layer. To generate the features, we crop the eyelid region from the input RGB images and reduce it to 110 dimensions using PCA. The output of the model is the locations of 20 control points (manually annotated) that represent the shapes of the eyelid margins. For training we used 98 frames from a video of 2335 frames. Eyelid margins are manually annotated. We cross-validated the model output with the manually annotated data set, and obtained an error (RMSE) of 1.2 pixels per eyelid marker on average per image frame.

The results of skin reconstruction around the eyes for a sequence where the subject looks around is shown in Fig. 2(c). In this example, the eyelid margin produces complex shapes and eyelid skin slides over the skull and globe surface. This makes it a perfect example to demonstrate the applicability of our reduced coordinate representation of skin as discussed in Sec. II-A. Our result shows recovering the characteristic deformation of eyelids can greatly enhance the expressiveness; as humans are very sensitive to even subtle skin motions in the eye region. We approximate skull and globe as one rigid structure on which eyelid skin slides. Therefore, we combined the face reconstructed by FaceShift with a globe model to generate a body mesh that approximates the body mesh on which skin slides. See Fig. 2(b). We also show skin reconstruction of a blink sequence in Fig. 3. In this example our reconstruction can also recover medial motion of lower eyelid skin, which is normally observed in human blinks.

[1]Point Grey Research, Vancouver, Canada
[2]https://www.superbrightleds.com

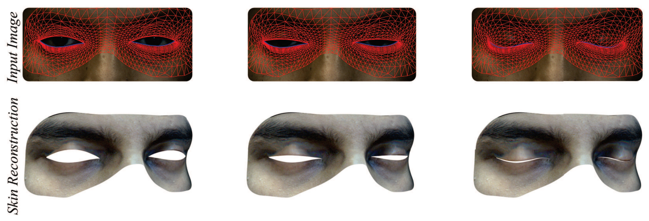

Fig. 3. Skin motion reconstruction in a blink sequence. Top row shows tracking points (red mesh) on input images. Bottom row shows 3D skin reconstructions.

B. Hand Tracking

To validate our algorithm with ground truth data for which the true skin movement was known, we used a hand tracking example with two artist generated animations of body motion. In the first animation the little finger is flexed, and in the second animation all the fingers are flexed to produce a hand grasping gesture. To the generate ground truth data we simulated skin sliding on the body using the skin simulation framework of Li, et al. [6]. The skin was then rendered using Autodesk's Maya software to generate an image sequence (1k $\times$ 1k resolution). The original animation and rendered image sequences are used as input to our algorithm.

We tracked and reconstructed 3D skin using our algorithm and compared against the ground truth skin movement to evaluate reconstruction error. The results of tracking are shown in Fig. 6. As expected, the projection of the body meshes shows large error from the ground truth as skin sliding motions are missing, whereas the error of our motion tracking remains low. In Fig. 6(b), the root mean squared error (RMSE) is plotted against the frame number, and the result of our tracking without motion refinement (shown in black) is also included. Without motion refinement the error gradually increases due to

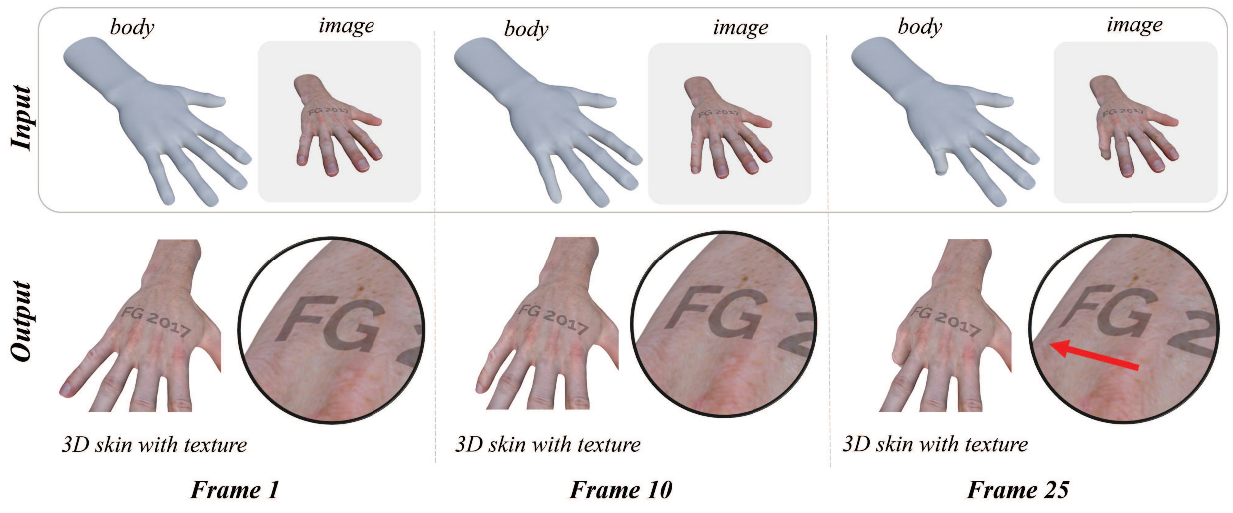

Fig. 4. Hand Tracking: Reconstruction of 3D skin for three frames in a hand movement sequence. The sliding motion of skin is produced by the flexion of the small finger. In the top row, the input body and image sequence is shown. In the bottom row, we show the 3D reconstruction of skin along with a zoomed in version to illustrate skin sliding. The red arrow in the last frame shows an approximate direction of skin sliding. See supplementary video to visualize the motion.

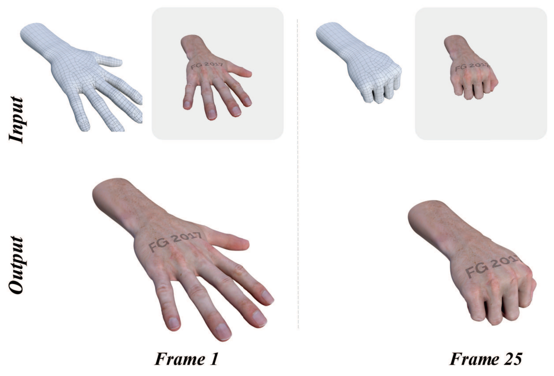

Fig. 5. Skin Reconstruction in a hand grasping gesture.

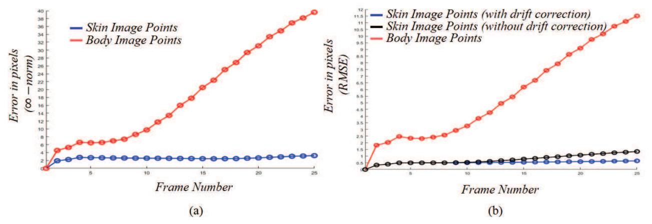

Fig. 6. Skin tracking error in image coordinates for hand tracking experiment. in (a) the errors are measured as ∞-norm between the tracked points and corresponding ground truth, in pixels. The body mesh is expected to produce high error (red) as it does not include skin sliding. In (b) we show that our algorithm produces low RMSE error with drift correction (blue: with drift correction, black: without drift correction).

drift in optical flow (shown in black), but the refinement step reduces the drift (in blue) which shows 52.57% reduction in error in the last frame of the hand tracking sequence. Note that the accuracy of tracking would vary depending on the tracking algorithm used. For example, in the last frame of the finger

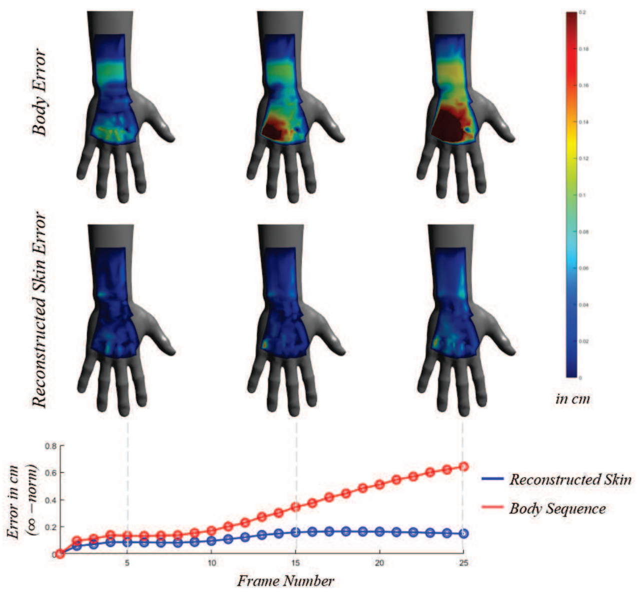

Fig. 7. Reconstruction error (in cm) of body sequence (top) and reconstructed skin (middle) from ground truth is shown in heatmaps on a rest pose. In bottom row, the errors (in ∞-norm) are plotted for all 25 frames in the hand tracking experiment.

flexing example, dense LDOF algorithm performs better with RMSE tracking error of 0.44 ± 0.12 pixels than feature-based KLT algorithm with RMSE error of 1.01 ± 0.29 pixels.

As discussed in Sec. II-B, we reconstruct 3D skin meshes sliding on the hand in physical space. The reconstructed skin meshes are very similar to the ground truth as we can see quantitatively in Fig. 7. The heatmaps depict the errors of each mesh vertex is measured as a distance (in cm) from the ground truth mesh for three frames (with interpolation), while the plot shows the ∞-norm of the overall mesh vertices for

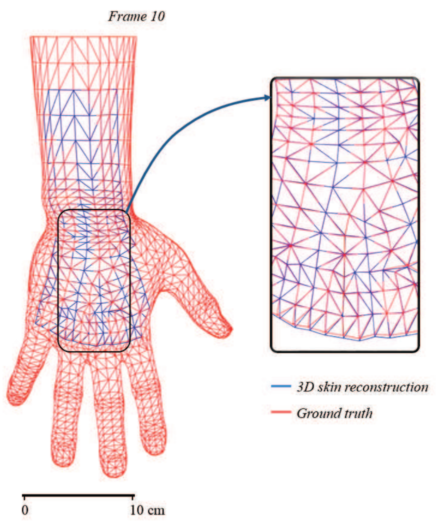

Fig. 8. Comparison of reconstructed 3D skin (blue) and ground truth (red) in hand tracking experiment.

TABLE I
SKIN RECONSTRUCTION ERROR IN HAND TRACKING EXPERIMENT

Type	Error(mm)	Frame 5	Frame 15	Frame 25
Finger Flexing	∞-norm	1.02	1.56	1.41
	RMSE	0.3 ± 0.16	0.33 ± 0.19	0.41 ± 0.2
Hand Grasping	∞-norm	1.29	1.54	2.63
	RMSE	0.51 ± 0.25	0.79 ± 0.33	0.81 ± 0.42

all the frames. A qualitative comparison is shown in Fig. 8. In our unoptimized Matlab implementation, on average, it takes 185s to compute the flow (with motion refinement) between two images (of size 1k square), 0.0078s to generate body map, and 7.54s to correct 3D skin reconstruction. We listed the reconstruction errors in mm in Table. I. The table includes result of both finger flexing and hand grasping sequences. See Fig. 5. In this case theThe hand grasping example has slightly high error as the motion is more extreme. Finally in Fig. 4, we show our reconstructed skin with texturing. The motion of the tattoo in the bottom row of Fig. 4 emphasizes the sliding of skin. Here skin can be thought of as the texture sliding on the body surface. To see the complete reconstruction sequence, please refer to our supplementary video.

C. Limitations

Our system has a few limitations. It requires the tracked skin features to be visible in all the image sequence to make sure that the skin points are not lost during tracking due to occlusions. Another situation where reconstruction error is high is when tracked points reaches the border of the visible skin region. There are two reasons for this: first, in the border body maps could be generated by extrapolation (which is only nearest neighbor interpolation in our case), and second, when the normal of the visible regions are in wide angle from the camera axis 3D reconstruction is very sensitive to small error in tracking. Another limitation is our reduced representation of skin cannot model out-of-plane deformations of skin, such as wrinkles. But fortunately, in our pipeline, these effects can be easily added by using normal or displacement maps.

IV. CONCLUSION

We proposed an efficient skin tracking and reconstruction algorithm that uses a reduced coordinate representation of skin. Using this representation, our method efficiently recovers 3D skin sliding motion by tracking 2D skin features in an image sequence. Most of the current face and gesture tracking and reconstruction methods ignore skin sliding, but our examples show that by recovering skin sliding we can add realism to an existing animation. Although we only showed applications of our method in facial expression and hand gesture examples, our method is very general and can be easily applied to other deforming objects with sliding surfaces. In future, we would like to eliminate some of the our limitations mentioned in Sec. III-C. In particular, we could reconstruct skin occluded from the camera by modeling elasticity of skin and solving a deformation energy minimization problem with the estimated tracked points of skin as constraints.

V. ACKNOWLEDGMENTS

This work was supported in part by grants from NSERC, Canada Foundation for Innovation, and the Canada Research Chairs Program.

REFERENCES

[1] Sofien Bouaziz, Yangang Wang, and Mark Pauly. Online modeling for realtime facial animation. *ACM Transactions on Graphics (TOG)*, 32(4):40, 2013.

[2] Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust articulated-icp for real-time hand tracking. volume 34, pages 101–114. Wiley Online Library, 2015.

[3] Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, et al. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics (TOG)*, 35(4):143, 2016.

[4] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. *Computer Vision – ECCV 2016*, October 2016.

[5] Jonathan J Dutton. Atlas of clinical and surgical orbital anatomy. *Philadelphia, PA*, 1994.

[6] Duo Li, Shinjiro Sueda, Debanga R Neog, and Dinesh K Pai. Thin skin elastodynamics. *ACM Transactions on Graphics (TOG)*, 32(4):49, 2013.

[7] Debanga R. Neog, João L. Cardoso, Anurag Ranjan, and Dinesh K. Pai. Interactive gaze driven animation of the eye region. *Proceedings of the 21st International Conference on Web3D Technology*, pages 51–59, 2016.

[8] Simon Baron-Cohen, Sally Wheelwright, Jacqueline Hill, Yogini Raste, and Ian Plumb. The "reading the mind in the eyes" test revised version: A study with normal adults, and adults with asperger syndrome or high-functioning autism. *Journal of child psychology and psychiatry*, 42(2):241–251, 2001.

[9] Amit Bermano, Thabo Beeler, Yeara Kozlov, Derek Bradley, Bernd Bickel, and Markus Gross. Detailed spatio-temporal reconstruction of eyelids. *ACM Transactions on Graphics (TOG)*, 34(4):44, 2015.

[10] Tsuyoshi Moriyama, Takeo Kanade, Jing Xiao, and Jeffrey F Cohn. Meticulously detailed eye region model and its application to analysis of facial images. *IEEE transactions on pattern analysis and machine intelligence*, 28(5):738–752, 2006.

[11] Javier Orozco, Ognjen Rudovic, Jordi Gonzàlez, and Maja Pantic. Hierarchical on-line appearance-based tracking for 3d head pose, eyebrows, lips, eyelids and irises. *Image and vision computing*, 31(4):322–340, 2013.

[12] Timothy F Cootes and Christopher J Taylor. On representing edge structure for model matching. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 1:I–1114, 2001.

[13] Pablo Garrido, Levi Valgaerts, Chenglei Wu, and Christian Theobalt. Reconstructing detailed dynamic face geometry from monocular video. *ACM Trans. Graph.*, 32(6):158–1, 2013.

[14] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, 2:690–696, 2000.

[15] Yuchao Dai, Hongdong Li, and Mingyi He. A simple prior-free method for non-rigid structure-from-motion factorization. *International Journal of Computer Vision*, 107(2):101–122, 2014.

[16] Adrien Bartoli, Yan Gérard, François Chadebecq, and Toby Collins. On template-based reconstruction from a single view: Analytical solutions and proofs of well-posedness for developable, isometric and conformal surfaces. *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2026–2033, 2012.

[17] Alessio Del Bue. A factorization approach to structure from motion with shape priors. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008.

[18] Paulo FU Gotardo and Aleix M Martinez. Computing smooth time trajectories for camera and deformable shape in structure from motion with occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):2051–2065, 2011.

[19] Abed Malti, Richard Hartley, Adrien Bartoli, and Jae-Hak Kim. Monocular template-based 3d reconstruction of extensible surfaces with local linear elasticity. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1522–1529, 2013.

[20] Mathieu Salzmann, Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. Closed-form solution to non-rigid 3d surface registration. *European conference on computer vision*, pages 581–594, 2008.

[21] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Training a feedback loop for hand pose estimation. *Proceedings of the IEEE International Conference on Computer Vision*, pages 3316–3324, 2015.

[22] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2011.

[23] Robin Sibson et al. A brief description of natural neighbour interpolation. *Interpreting multivariate data*, 21:21–36, 1981.