Model-Based 3D Hand Pose Estimation from Monocular Video

Martin de La Gorce, *Member, IEEE,* David J. Fleet, *Senior, IEEE* and Nikos Paragios, *Senior, IEEE*

Abstract—A novel model-based approach to 3D hand tracking from monocular video is presented. The 3D hand pose, the hand texture and the illuminant are dynamically estimated through minimization of an objective function. Derived from an inverse problem formulation, the objective function enables explicit use of temporal texture continuity and shading information, while handling important self-occlusions and time-varying illumination. The minimization is done efficiently using a quasi-Newton method, for which we provide a rigorous derivation of the objective function gradient. Particular attention is given to terms related to the change of visibility near self-occlusion boundaries that are neglected in existing formulations. To this end we introduce new occlusion forces and show that using all gradient terms greatly improves the performance of the method. Qualitative and quantitative experimental results demonstrate the potential of the approach.

Index Terms—Hand Tracking, Model Based Shape from Shading, Generative Modeling, Pose Estimation, Variational Formulation, Gradient Descent

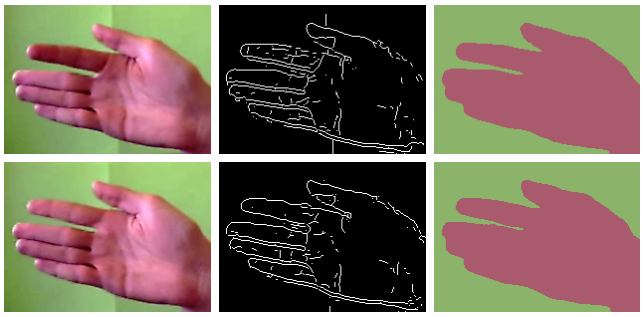


Fig. 1. Two hand pictures in the first column and their corresponding edge map and segmented silhouette in the two other columns. Edge and Silhouette are little informative to disambiguate the two different index poses.

1 Introduction

Hand gestures provide a rich form of nonverbal human communication for man-machine interaction. To this end, hand tracking and gesture recognition are central enabling technologies. Data gloves are commonly used as input devices but they are expensive and often inhibit free movement. As an alternative, vision-based tracking is an attractive, non-intrusive approach.

Fast, effective, vision-based hand pose tracking is, however, challenging. The hand has approximately 30 degrees of freedom, so the state space of possible hand poses is large. Seaching for the pose that is maximally consistent with an image is computationally demanding. One way to improve state space search in tracking is to exploit predictions from recently estimated poses, but this is often ineffective for hands as they can move quickly and in complex ways. Thus, predictions have large variances. The monocular tracking problem is exacerbated by inherent depth uncertainties and reflection ambiguities that produce multiple local optima.

Another challenge concerns the availability of useful visual cues. Hands are usually skin colored and it is difficult to discriminate one part of the hand from another based solely on color. The silhouette of the hand, if available, provides only weak information about the relative positions of different fingers. Optical flow estimates are not reliable as hands have little surface texture and are often self-occluding. Edge information is often ambiguous due to clutter. For example, the first column in Fig. 1 shows images of a hand with different poses of its index finger. The other columns show corresponding edge maps and silhouettes, which remain unchanged as the finger moves; as such, they do not reliably constrain the hand pose. For objects like the hand, which are relatively uniform in color, we posit that shading is a crucial visual cue. Nevertheless, shading has not been used widely for articulated tracking (but see [1], [2]). The main reason is that shading constraints require an accurate model of surface shape. Simple models of hand geometry where hands are approximated as a small number of ellipsoidal or cylindrical solids may not be detailed enough to obtain useful shading constraints. Surface occlusions also complicate shading cues.

Two complementary approaches have been suggested for monocular hand tracking. *Discriminative* methods aim to recover hand pose from a single frame through classification or regression techniques (e.g., [3], [4], [5], [6]). The classifier is learned from training data that is generated off-line with a synthetic model, or acquired by a camera from a small set of known poses. Due to the large number of hand DOFs it is impractical to densely sample the entire state space. As a consequence, these methods are perhaps best suited for rough initialization or recognition of a limited set of predefined poses.

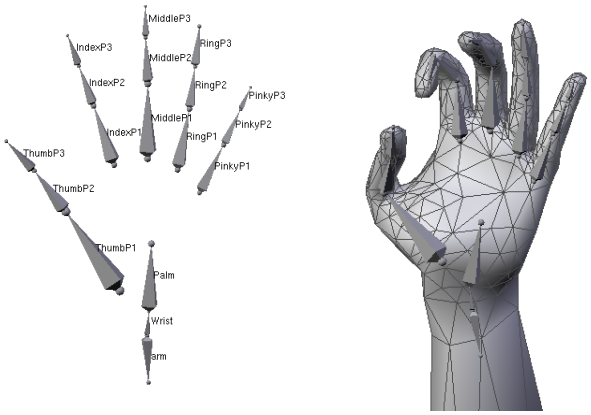
Generative methods use a 3D articulated hand model whose projection is aligned with the observed image for

pose recovery (e.g., [7], [8], [9], [10], [11]). The model projection is synthesized on-line and the registration of the model to the image can be done using local search, ideally with continuous optimization. A variety of cues such as the distance to edges, segmented silhouettes [12], [13] or optical flow [1] can be used to guide the registration. The method in [14] combines discriminative and generative approaches but does not use on-line synthesis. A set of possible poses in generated in advance, which restrict the set of pose that can be tracked.

Not surprisingly, similar challenges exist for the related problem of fully-body human pose tracking. For full-body tracking it is particularly difficult to formulate good likelihood models due to the significant variability in shape and appearance (e.g., see [15]). Interestingly, with the exception of recent work in [16] on human body shape and pose estimation for unclothed people, shading has not been used extensively for human pose tracking.

This paper advocates the use of richer generative models of the hand, both in terms of geometry and appearance, along with carefully formulated gradient-based optimization. We introduce a new analysis-by-synthesis formulation of the hand tracking problem that incorporates both shading and texture information while handling self-occlusion. Given a parametric hand model, and a well-defined image formation process, we seek the hand pose parameters which produce the synthetic image that is as similar as possible to the observed image. Our similarity measure (i.e., our *objective function*) simply comprises the sum of residual errors, taken over the image domain. The use of a triangulated mesh-based model allows for a good shading model. By also modeling the texture of the hand, we obtain a method that naturally captures the key visual cues without the need to add new ad-hoc terms to the objective function.

During the tracking process we determine, for each frame, the hand and illumination parameters by minimizing an objective function. The hand texture is then updated in each frame, and then remains static while fitting the model pose and illumination in the next frame. In contrast to the approach described in [1], which relies on optical flow, our objective function does not assume small inter-frame displacements in its formulation. It therefore allows large displacements and discontinuities. The optimal hand pose is determined through a quasi-Newton descent using the gradient of the objective function. In particular, we provide a novel, detailed derivation of the gradient in the vicinity of depth discontinuities, showing that there are important terms in the gradient due to occlusions. This analysis of continuity in the vicinity of occlusion boundaries is also relevant, but yet unexplored, for the estimation of full-body human pose in 3D. Finally, we test our method on challenging sequences involving large self-occlusion and out-of-plane rotations. We also introduce sequences with 3D ground truth to allow for quantitative performance analysis.



(a) (b)

Fig. 2. (a) The skeleton (b) The deformed hand triangulated surface

2 Generative Model

2.1 Synthesis

Our analysis-by-synthesis approach requires an image formation model, given a 3D hand pose, surface texture, and an illuminant. The model is derived from well-known *computer animation* and *graphics* concepts.

Following [17], we model the hand surface by a 3D, closed and orientable, triangulated surface. The surface mesh comprising 1000 facets (Fig. 2b). It is deformed according to pose changes of an underlying articulated skeleton using Skeleton Subspace Deformation [18], [19]. The skeleton comprises 18 bones with 22 degrees of freedom (DOF). Each DOF corresponds to an articulation angle whose range is bounded to avoid unrealistic poses. The pose is represented by a vector θ, comprising 22 articulation parameters plus 3 translational parameters and a quaternion to define the global position and orientation of the wrist with respect to the camera. To accommodate different hand shapes and sizes, we also add 54 scaling parameters (3 per bone), called *morphological parameters*. These parameters are estimated during the calibration process (see Sec. 4.1), subject to linear constraints that restrict the relative lengths of the parts within each finger.

Since hands have relatively little texture, shading is essential to modeling hand appearance. Here adopt the Gouraud shading model and assume Lambertian reflectance. We also include an adaptive albedo function to model texture and miscellaneous, otherwise unmodeled, appearance properties. The illuminant model includes ambient light and a distant point source, and is specified by a 4D vector denoted by L, comprising three elements for a directional component, and one for an ambient component. The irradiance at each vertex of the surface mesh is obtained by the sum of the ambient coefficient and the scalar product between the surface normal at the vertex and the light source direction. The irradiance across each face is then obtained through bilinear interpolation. Multiplying the reflectance and the irradiance

yields the appearance for points on the surface.

Texture (albedo variation) can be handled in two ways. The first associates an RGB triplet with each vertex of the surface mesh, from which one can linearly interpolate over mesh facets. This approach is conceptually simple but computationally inefficient as it requires many small facets to accurately model smooth surface radiance. The second approach, widely used in computer graphics, involves mapping an RGB reflectance (texture) image onto the surface. This technique preserves detail with a reasonably small number of faces.

In contrast with previous methods in computer vision that used textured models (e.g., [20]), our formulation (Sec. 3) requires that surface reflectance be continuous over the surface. Using bilinear interpolation of the discretized texture we ensure continuity of the reflectance within each face. However, since the hand is a closed surface it is impossible to define a continuous bijective mapping between the whole surface and a 2D planar surface. Hence, there is no simple way to ensure continuity of the texture over the entire surface of the hand.

Following [21] and [22], we use patch-based texture mapping. Each facet is associated with a triangular *region* in the texture map. The triangles are uniform in size and have integer vertex coordinates. As depicted in Fig. 3, each facet with an odd (respectively even) index is mapped onto a triangle that is the left-upper-half (respectively right-lower-half) of a square divided along its diagonal. Because we use bilinear interpolation we need to reserve some texture pixels (texels) outside the diagonal edges for points with non-integer coordinates (see Fig.3). This representation is therefore redundant for points along edges of the 3D mesh; i.e., 3D edges of the mesh belong to two adjacent facets and therefore occur twice in the texture map, while each vertex occurs an average of 6 times (according to the Eulerian properties of the mesh). We therefore introduce constraints to ensure consistency between RGB values at different points in the texture map that map to the same edge or vertex of the 3D mesh. By enforcing consistency we also ensure the continuity of the texture across edges of the mesh.

With bilinear texture mapping, consistency can be enforced with two sets of linear constraints on the texture map. The first set specifies that the intensities of points on mesh edges with integer-coordinates in the texture map must be consistent. The second set enforces the interpolation of the texture to be linear along edges of the triangular patches in the texture map. As long as the texture interpolation is linear along each edge, and the intensities of the texture map are consistent at points with integer texture coordinates, the mapping will be consistent for all points along each edge. The bilinear interpolation is already linear along vertical and horizontal edges, so we only need to consider the diagonal edges while defining the second set of constraints. Let T denote the discrete texture image. Let $(i+1, j)$ and $(i, j+1)$ denote two successive points with integer coordinates along a diagonal edge of a triangular patch

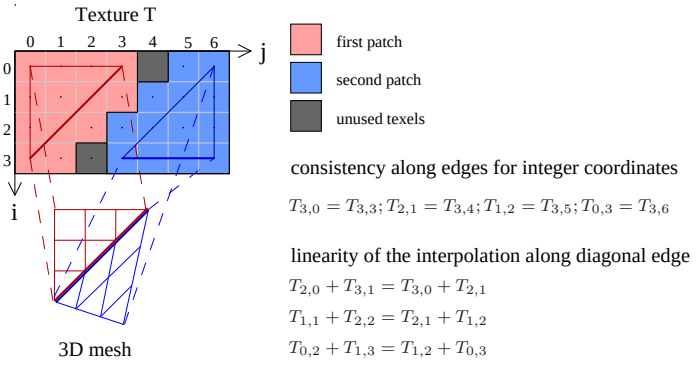


Fig. 3. Adjacent facets of the triangulated surface mesh project to two triangles in the 2D texture map T. Since the shared edge projects to distinct segments in the texture map, one must specify constraints that the texture elements along the shared edge be consistent. This is done here using 7 linear equality constraints.

in the texture map. Using bilinear texture mapping, the texture intensities of points with non-integer coordinates $(i+1-\lambda, j+\lambda), \lambda \in (0,1)$ along the edge are given by

$$\lambda(1-\lambda)(T_{i,j} + T_{i+1,j+1}) + \lambda^2 T_{i,j+1} + (1-\lambda)^2 T_{i+1,j} \ . \quad (1)$$

By twice differentiating this expression with respect to λ, we find that it is linear with respect to λ if and only if the following linear constraint is satisfied:

$$T_{i,j} + T_{i+1,j+1} = T_{i+1,j} + T_{i,j+1} \ . \quad (2)$$

The second set of constraints are for those points along the diagonal edges. Finally, let $\mathbb{T}$ denote the linear subspace of valid textures, i.e., whose RGB values satisfy the linear constraints to ensure continuity over the surface.

Given the mesh geometry, the illuminant, and the texture map, the formation of the model image at each 2D image point $\mathbf{x}$ can be determined. First, as in ray-tracing, we determine the first intersection between the triangulated surface mesh and the ray starting at the camera center and passing through $\mathbf{x}$. If no intersection exists the image at $\mathbf{x}$ is determined by the background. The appearance of each visible intersection point is computed using the illuminant and information at the vertices of the visible face. In practice, the image is computed on a discrete grid and the image synthesis can be done efficiently using the triangle rasterization technique in combination with a depth buffer.

When the background is static, we simply assume that an image of the background is readily available. Otherwise, we assume a probabilistic model for which background pixel colors are drawn from a background density $p_{bg}(\cdot)$ (e.g., it can be learned in the first frame with some user interaction).

This completes the definition of the generative process for hand images. Let $I_{syn}(\mathbf{x}; \theta, L, T)$ denote the synthetic image comprising the hand and background, at the point $\mathbf{x}$ for a given pose θ, texture T and illuminant L.

I_{obs} I_{syn} R

Fig. 4. The observed image I_{obs}, the synthetic image I_{syn} and the residual image R

2.2 Objective Function

Our task is to estimate the pose parameters θ, the texture T, and the illuminant L that produce the synthesized image $I_{syn}()$ that best matches the observed image, denoted $I_{obs}()$. Our objective function is based on a simple discrepancy measure between these two images. First, let the residual image R be defined as

$$R(\mathbf{x};\theta,T,L) \;=\; \rho\big(I_{syn}(\mathbf{x};\theta,L,T) - I_{obs}(\mathbf{x})\big)\;, \qquad (3)$$

where ρ is either the classical squared-error function or a robust error function such as the Huber function used in [23]. For our experiments we choose ρ to be the conventional squared-error function, and therefore the pixel errors are implicitly assumed to be IID Gaussian. Contrary to this IID assumption, nearby errors often tend to be highly correlated in practice. This is mainly due to simplifications of the model. Nevertheless, as the quality of the generative model improves, as we aim to do in this paper, these correlations become less significant.

When the background is defined probabilistically, we separate the image domain Ω (a continuous subset of $\mathbb{R}^2$) into the region $S(\theta)$ covered by hand, given the pose, and the background $\Omega\backslash S(\theta)$. Then the residual can be expressed using the background color density as

$$R(\mathbf{x};\theta,L,T) \;=\; \begin{cases} \rho\big(I_{syn}(\mathbf{x};\theta,L,T) - I_{obs}(\mathbf{x})\big),\;\; \forall \mathbf{x} \in S(\theta) \\ -\log p_{bg}(I_{obs}(\mathbf{x}))\;,\;\; \forall \mathbf{x} \in \Omega\backslash S(\theta) \end{cases} \qquad (4)$$

Fig. 11 (rows 3 and 6) depicts an example of the residual function for the likelihood computed in this way.

Our discrepancy measure, E, is defined to be the integral of the residual error over the image domain Ω:

$$E(\theta,T,L) \;=\; \int_{\Omega} R(\mathbf{x};\theta,L,T)\,d\mathbf{x} \qquad (5)$$

This simple discrepancy measure is preferable to more sophisticated measures that combine heterogeneous cues like optical flow, silhouettes, or chamfer distances between detected and synthetic edges. First, it avoids the tuning of weights associated with different cues that is often problematic; in particular, a simple weighted sum of errors from different cues implicitly assumes (usually incorrectly) that errors in different cues are statistically independent. Second, the measure in (5) avoids early decisions about the relevance of edges through thresholding, about the area of the silhouette by segmentation, and about the position of discontinuities in optical flow. Third, (5) is a continuous function of θ; this is not the case for measures based on distances between edges like the symmetric chamfer distance.

By changing the domain of integration, the integral of the residual within the hand region can be re-expressed as a continuous integral over the visible part of the surface. It can then be approximated by a finite weighted sum over centers of visible faces. Much of the literature on 3D deformable models adopts this approach, and assumes that the visibility of each face is binary (fully visible or fully hidden), and can be obtained from a depth buffer (e.g., see [20]). Unfortunately, such discretizations produce discontinuities in the approximate functional as θ varies. When the surface moves or deforms, the visibility state of a face near an occlusion boundary is likely to change, causing a discontinuity in the sum of residuals. This is problematic for gradient-based optimization. To preserve continuity of the discretized functional with respect to θ, the visibility state should not be binary. Rather, it should be a real-valued and smooth as the surface becomes occluded or unoccluded. In practice, this is cumbersome to implement and the derivation of the gradient is complicated. Therefore, in contrast with much of the literature on 3D deformable models, we keep the formulation in the continuous image domain when deriving the expression of functional gradient.

To estimate the pose θ and the lighting parameters L for each new image frame, or to update the texture T, we look for minima of the objective function. During tracking, the optimization involves two steps. First we minimize (5) with respect to θ and L, to register the model with respect to the new image. Then we minimize the error function with respect to the texture T to find the optimal texture update.

3 Parameter Estimation

The simultaneous estimation of the pose parameters θ and the illuminant L is a challenging non-linear, non-convex, high-dimensional optimization problem. Global optimization is impractical and therefore we resort to an efficient quasi-Newton, local optimization. This requires that we are able to efficiently compute the gradient of E in (5) with respect to θ and L.

The gradient of E with respect to lighting L is straightforward to derive. The synthetic image $I_{syn}()$ and the residual $R()$ are smooth functions of the lighting parameters L. As a consequence, we can commute differentiation and integration to obtain

$$\begin{aligned} \frac{\partial E}{\partial L}(\theta,L,T) &= \frac{\partial}{\partial L}\int_{\Omega} R(\mathbf{x};\theta,L,T)\,d\mathbf{x} \\ &= \int_{\Omega} \frac{\partial R}{\partial L}(\mathbf{x};\theta,L,T)\,d\mathbf{x}\;. \end{aligned} \qquad (6)$$

Computing $\frac{\partial R}{\partial L}(\mathbf{x};\theta,L)$ is straightforward with application of the chain rule to the generative process.

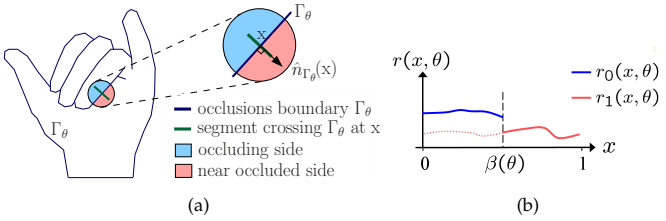


Fig. 5. (a) Example of a segment crossing the occlusions boundary Γ_θ. (b) A curve representing the residual along this segment

Formulating the gradient of E with respect to pose θ is not straightforward. This is the case along occlusion boundaries where the residual is a discontinuous function. As a consequence, for this subset of points, $R(\mathbf{x};\theta,L,T)$ is not differentiable and therefore we cannot commute differentiation and integration. Nevertheless, while the residual is discontinuous, the energy function remains continuous in θ. The gradient of E with respect to θ can therefore be specified analytically. Its derivation is the focus of the next section.

3.1 Gradient With Respect to Pose and Lighting

The generative process above was carefully formulated so that scene radiance is continuous over the hand. We further assume that the background and the observed image are known and continuous. Therefore the residual error is spatially continuous everywhere except at self-occlusion and occlusions boundaries denoted by Γ_θ.

3.1.1 1D Illustration

To sketch the main idea behind the gradient derivation we first consider a 1D residual function on a line segment that crosses a self-occlusion boundary, as shown in Fig. 5. Along this line segment the residual function is continuous everywhere except at the self-occlusion location, β. Accordingly we express the residual on the line in two parts, namely, the residual to the left of the boundary r_0 and the residual on the right r_1:

$$r(x,\theta) \;=\; \begin{cases} r_0(x,\theta), & x \in [0,\beta(\theta)] \\ r_1(x,\theta), & x \in (\beta(\theta),1] \end{cases} \quad . \tag{7}$$

Accordingly, the energy is then the sum of the integral of r_0 on the left and the integral of r_1 on the right:

$$E \;=\; \int_0^{\beta(\theta)} r_0(x,\theta)\,dx + \int_{\beta(\theta)}^1 r_1(x,\theta)\,dx \;. \tag{8}$$

Note that β is a function of the pose parameter θ. Intuitively, when θ varies the integral E is affected in two ways. First the residual functions r_0 and r_1 vary, e.g. due to shading changes. Second, the boundary location β also moves.

Mathematically, the total derivative of E with respect to θ is the sum of two terms,

$$\frac{dE}{d\theta} \;=\; \frac{\partial E}{\partial \theta} + \frac{\partial E}{\partial \beta}\frac{\partial \beta}{\partial \theta} \;. \tag{9}$$

The first term, the partial derivative of E with respect to θ, with β fixed, corresponds to the integration of the residual derivative everywhere except the discontinuity:

$$\frac{\partial E}{\partial \theta} \;=\; \int_{[0,1]\setminus\{\beta\}} \frac{\partial r}{\partial \theta}(x,\theta)\,dx \;. \tag{10}$$

The second term in (9) depends on the partial derivative of E with respect to the boundary location β. Using the fundamental theorem of calculus, this term reduces to the difference between the residuals at both sides of the occlusion boundary, multiplied by the derivative of the boundary location β with respect to the pose parameters θ. From (8) it follows that

$$\frac{\partial E}{\partial \beta}\frac{\partial \beta}{\partial \theta} \;=\; [r_0(\beta,\theta) - r_1(\beta,\theta)]\,\frac{\partial \beta}{\partial \theta} \;. \tag{11}$$

While the residual $r(x,\theta)$ is a discontinuous function of θ at β, the energy E is still a differentiable function of θ.

3.1.2 General 2D Case

The 2D case is a generalization of the 1D example. Like the 1D case, the residual image is spatially continuous almost everywhere, except for points at occlusion boundaries. At such points the hand occludes the background or other parts of the hand (see Fig. 5a). Let Γ_θ be the set of boundary points, i.e., the support of depth discontinuities in the image domain. Because we are working with a triangulated surface mesh, Γ_θ can be decomposed into a set line segments. More precisely, Γ_θ is the union of the projections of all visible portions of edges of the triangulated surface that separate front-facing facets from back-facing facets. For any point $\mathbf{x}$ in Γ_θ, the corresponding edge projection locally separates the image domain into two subregions of $\Omega\setminus\Gamma_\theta$, namely, the *occluding* side and the *near-occluded* side (see Fig. 5).

Along Γ_θ, the residual image $R()$ is discontinuous with respect to $\mathbf{x}$ and θ. Nevertheless, like in the 1D case, to derive ∇E, we need to define a new residual $R^+(\theta,\mathbf{x})$, which extends $R()$ by continuity onto the occluding points Γ_θ. This is done by replicating the content of $R()$ in $\Omega\setminus\Gamma_\theta$ from the *near-occluded* side. In forming R^+, we are recovering the residual of *occluded faces* in the vicinity of the boundary points. Given a point $\mathbf{x}$ on a boundary segment with outward normal $\hat{n}_{\Gamma_\theta}(\mathbf{x})$ (i.e., facing the *near-occluded* side as in Fig. 5a), we define

$$R^+(\theta,\mathbf{x}) \;=\; \lim_{k\to\infty}\left(R\left(\theta,\, \mathbf{x}+\frac{\hat{n}_{\Gamma_\theta}(\mathbf{x})}{k}\right)\right) \;. \tag{12}$$

One can interpret $R^+(\theta,\mathbf{x})$ on Γ_θ as the residual we would have obtained if the near-occluded surface had been visible instead of the occluding surface.

When a pose parameter θ_j is changed with an infinitesimal step $d\theta_j$, the occluding contour Γ_θ in the neighborhood of $\mathbf{x}$ moves with an infinitesimal step $v_j(\mathbf{x})\,d\theta_j$ (see (34) for a definition of the boundary speed v_j). Then the residual in the vicinity of $\mathbf{x}$ is discontinuous, *jumping* between $R^+(\theta,\mathbf{x})$ and $R(\theta,\mathbf{x})$. However, the surface area where this jump occurs is also infinitesimal

and proportional to $v_j(\mathbf{x})\,\hat{n}_{\Gamma_\theta}(\mathbf{x})\,d\theta_j$. The jump causes an infinitesimal change in E after integration over the image domain Ω.

Like the 1D case, one can express the functional gradient $\nabla_\theta E \equiv (\frac{\partial E}{\partial \theta_1}, \ldots, \frac{\partial E}{\partial \theta_n})$ using two terms:

$$\frac{\partial E}{\partial \theta_j} = \int_{\Omega \setminus \Gamma_\theta} \frac{\partial R}{\partial \theta_i}(\mathbf{x};\theta,L,T)d\mathbf{x} - \int_{\Gamma^\theta} \underbrace{\left[R^+(\mathbf{x};\theta,T,L) - R(\mathbf{x};\theta,L,T)\right]\hat{n}_{\Gamma_\theta}(\mathbf{x})}_{f_{oc}(x)}\, v_i(\mathbf{x})\, d\mathbf{x} \tag{13}$$

The first term captures the energy variation due to the smooth variation of the residual in $\Omega\setminus\Gamma_\theta$. It integrates the partial derivative of the residual R with respect to θ everywhere but at the occluding contours Γ_θ, where it is not defined. The analytical derivation of $\frac{\partial R}{\partial \theta_i}$ on $\Omega\setminus\Gamma_\theta$ requires application of the chain rule to the functions that define the generative process.

The second term in (13) captures the energy variation due to the movement of occluding contours. It integrates the difference between the residual on either side of the occluding contour, multiplied by the normal speed of the boundary when the pose varies. Here, f_{oc} is a vector field whose directions are normal to the curve and whose magnitudes are proportional to the difference of the residuals on either each side of the curve. These *occlusion forces* account for the displacement of occlusion contours as θ varies. They are similar to the forces obtained with 2D active regions [24], which derives from the fact that we kept the image domain Ω continuous while computing the functional gradient. Because the surface is triangulated, Γ_θ is a set of image line segments, and we could rewrite (13) in a form that is similar to the gradient flow reported in [25] for active polygons. Our *occlusion forces* also bear similarity to the gradient flows of [26], [27] for multi-view reconstruction, where some terms account for the change of visibility at occlusion boundaries. In their formulation no shading and texture are attached to the reconstructed surface, which results in substantial differences in the derivation.

3.1.3 Computing $\nabla_\theta E$

A natural numerical approximation to E is to first sample R at discrete pixel locations and then sum, i.e.,

$$E(\theta,T,L) \approx \tilde{E}(\theta,T,L) \equiv \sum_{\mathbf{x}\in\Omega\cap\mathbb{N}^2} R(\mathbf{x};\theta,L,T)\ . \tag{14}$$

Similarly, one might discretize the two terms of the gradient $\nabla_\theta E$ in (13). The integral over $\Omega\setminus\Gamma_\theta$ could be approximated by a finite sum over image pixel in $\Omega\cap\mathbb{N}^2$ while the integral along contours Γ_θ could be approximated by sampling a fixed number of points along each segment. Let $\tilde{\nabla}_\theta E$ be the numerical approximation to the gradient obtained in this way.

There are, however, several practical problems with this approach. First, the approximate gradient is not the exact gradient of the approximate objective function, i.e., $\tilde{\nabla}_\theta E \neq \nabla_\theta \tilde{E}$. Most iterative gradient-based optimization methods require direct numerical evaluation of the objective function to ensure that energy decreases at each iteration. Such methods expect the gradient to be consistent with the objective function.

Alternatively, one might try to compute the exact gradient of the approximate objective function, i.e., $\nabla\tilde{E}$. Unfortunately, due to the aliasing along occluding edges (i.e., discretization noise in the sampling in (14)), $\tilde{E}(\theta,T,L)$ is not a continuous function of θ. Discontinuities occur, for example, when an occlusion boundary crosses the center of a pixel. As a result of these discontinuities, one cannot compute the exact derivative, and numerical difference approximations are likely to be erroneous.

3.2 Differentiable Discretization of Energy E

To obtain a numerical approximation to E that can be differentiated exactly, we consider an approximation to E with antialiasing. This yields a discretization of the energy, denoted $\bar{E}$, that is continuous in θ and can be differentiated in closed form using the chain rule, to obtain $\nabla_\theta \bar{E}$. The anti-aliasing technique is inspired by the *discontinuity edge overdraw* method [28]. We use it here to form a new residual function, denoted $\bar{R}$.

The anti-aliasing technique progressively blends the residuals on both sides of occlusion boundaries. For a point $\mathbf{x}$ that is close to an occluding edge segment we compute a blending weight that is proportional to the signed distance to the segment:

$$w(\mathbf{x}) = \frac{(\mathbf{x} - q(\mathbf{x}))\cdot\hat{n}}{max(|\hat{n}_x|,|\hat{n}_y|)}\ , \tag{15}$$

where $q(\mathbf{x})$ is a point on the nearby segment, and $\hat{n} = (\hat{n}_x, \hat{n}_y)$ is the unit normal to the segment (pointing toward the *near-occluded* side of the boundary). Dividing the signed distance by $max(|\hat{n}_x|,|\hat{n}_y|)$ is a minor convenience that allows us to interpret terms of the gradient $\nabla_\theta\bar{E}$ in terms of the occlusion forces defined in (13). A detailed derivation is given in Appendix A.

Let the image segment $\overline{\bar{V}_m\bar{V}_n}$ be defined by end points $\bar{V}_m$ and $\bar{V}_n$. For $\overline{\bar{V}_m\bar{V}_n}$ we define an *anti-aliased region*, denoted $\mathcal{A}$, as a set of points on the occluded side near the segment (see Fig. 17); i.e., points that that project to the segment with weights $w(\mathbf{x})$ in $[0,1)$;

$$\mathcal{A} = \{\mathbf{x}\in\mathbb{R}^2 \,|\, w(\mathbf{x})\in[0,1),\ q(\mathbf{x})\in\overline{\bar{V}_m\bar{V}_n}\}\ . \tag{16}$$

For each point in this region we define the anti-aliased residual to be a linear combination of the residuals on the occluded and occluding sides:

$$\bar{R}(\mathbf{x};\theta,L,T) = w(\mathbf{x})R(\mathbf{x};\theta,L,T) + (1-w(\mathbf{x}))R(q(\mathbf{x});\theta,L,T)\ . \tag{17}$$

The blending is not symmetric with respect to the line segment, but rather it is shifted toward the occluded side. This allows use of a Z-buffer to handle occlusions

(see [28] for details). For all points outside the anti-aliased regions we let $\bar{R}(\mathbf{x};\theta,L,T) = R(\mathbf{x};\theta,L,T)$.

Using this anti-aliased residual image, $\bar{R}$, we define a new approximation to the objective function, denoted $\bar{E}$:

$$\bar{E}(\theta,T,L) \;=\; \sum_{\mathbf{x}\in\Omega\cap\mathbb{N}^2} \bar{R}(\mathbf{x};\theta,L,T)\;. \tag{18}$$

This anti-aliasing technique makes $\bar{R}$ and thus $\bar{E}$ continuous with respect to θ even along occlusion boundaries.

There are other anti-aliasing methods such as oversampling [29] or A-buffer [30]. They might reduce the magnitude of the jump in the residual when an occlusion boundary crosses the center of a pixel, as θ changes, perhaps making the jump visually imperceptible, but the residual remains discontinuous in θ. The edge overdraw method we use in (17) ensures the continuity of $\bar{R}$ and hence $\bar{E}$ with respect to θ.

To derive $\nabla_\theta \bar{E}$ we differentiate $\bar{R}$ using the chain rule.

$$\nabla_\theta \bar{E} \;=\; \sum_{\mathbf{x}\in\Omega\cap\mathbb{N}^2} \frac{\partial}{\partial\theta}\bar{R}(\mathbf{x};\theta,L,T)\;. \tag{19}$$

Using a backward ordering of derivative multiplications (called adjoint coding in the algorithm differentiation literature [31]), we obtain an evaluation of the gradient with computational cost that is comparable to the evaluation of the objective function.

The gradient of $\bar{E}$ with respect to θ, i.e., $\nabla_\theta \bar{E}$, is one of the possible discretizations of the analytical $\nabla_\theta E$. The differentiation of the anti-aliasing process with respect to θ yielded terms that sum along the edges and that can been interpreted as a possible discretization of the *occlusion forces* in (13). This is explained in Appendix A. The advantage of this approach over that discussed in section 3.1.3 is that $\nabla_\theta \bar{E}$ is the exact gradient of $\bar{E}$, and both can be numerically evaluated. This also allows one to check validity of the gradient $\nabla_\theta \bar{E}$ obtained by direct comparison with divided differences on $\bar{E}$. Divided differences are prone to numerical error and inefficient, but help to detect errors in the gradient computation.

3.3 Model Registration

3.3.1 Sequential Quadratic Programming

During the tracking process, the model of the hand is registered to each new frame by minimizing the approximate objective function $\bar{E}$. Let $P = [\theta, L]^T$ comprise the unknown pose and lighting parameters, and assume we obtain an initial guess by extrapolating estimates from the previous frames. We then refine the pose and lighting estimates by minimizing $\bar{E}$ with respect to P, subject to linear constraints which enforce joint limits.

$$\begin{aligned} \tilde{P} \;=\; & \underset{P}{argmin} \quad \bar{E}(P) \\ & \text{s.t. } AP \le b \end{aligned} \tag{20}$$

Using the object function gradient, we minimize $\bar{E}(P)$ using a sequential quadratic programming method [32] with an adapted Broyden-Fletcher-Goldfarb-Shanno (BFGS) Hessian approximation. This allows us to combine the well-known BFGS quasi-Newton method with the linear joint limit constraints. There are 4 main steps:

1) A quadratic program is used to determine a descent direction. This uses the energy gradient and an approximation to the Hessian, $\tilde{H}_t$, based on a modified BFGS procedure (see Appendix B):

$$\begin{aligned} \Delta_P = \quad & \underset{\Delta_P}{argmin} \left(\frac{d\bar{E}}{dP}(P_t)\Delta_P + \frac{1}{2}\Delta_P^t \tilde{H}_t \Delta_P \right) \\ & \text{s.t. } A(P_t+\Delta_P) \le b \end{aligned} \tag{21}$$

2) A line search is then performed in that direction:

$$\lambda^* \;=\; \underset{\lambda}{argmin}\, \bar{E}(P_t + \lambda\Delta_P) \;\text{ s.t. } \lambda \le 1\;. \tag{22}$$

The inequality $\lambda \le 1$ ensures that we stay in the linearly constrained subspace.

3) The pose and lighting parameters are updated

$$P_{t+1} \;=\; P_t + \lambda^*\Delta_P\;. \tag{23}$$

4) The approximate Hessian $\tilde{H}_{t+1}$ is updated using the adapted BFGS formula.

These steps are iterated until convergence.

To initialize the search with a good initial guess, we first perform a 1D search along the line that linearly extrapolates the two previous pose estimates. Each local minimum obtained during this 1D search is used as a starting point for our SQP method in the full pose space. The number of starting points found is usually just one, but when there are several, the results of the SQP method are compared and the best solution is chosen. Once the model is fit to a new frame, we then update the texture, which is then used during registration in the next frame.

3.4 Texture Update

Various methods for mapping images onto a static 3D surface exist. Perhaps the simplest method involves, for each 3D surface point, (1) computing its projected image coordinates, (2) checking its visibility by comparing its depth with the depth buffer, and (3) if the point is visible, interpolating image intensities at the image coordinates and then recovering the albedo by dividing the interpolated intensity by the model irradiance at that point. This approach is not suitable for our formulation for several reasons. First, we want to model texture for hidden areas of the hand, as they may become visible in the next frame. Second, our texture update scheme must avoids progressive contamination by the background color, or self-contamination between different parts of the hand.

Here, we formulate texture estimation as the minimization of the same basic objective functional as that used for tracking, in combination with a smoothness regularization term (independent of pose and lighting). That is, for texture T we minimize

$$E_{texture}(T) = \bar{E}(\theta,L,T) + \beta E_{sm}(T). \tag{24}$$

Fig. 6. Observed image and extracted texture mapped on the hand under the camera viewpoint and two other viewpoints. The texture is smoothly diffused into regions that are hidden from the camera viewpoint

For simplicity the smoothness measure is defined to be the sum of squared differences between adjacent pixels in the texture (texels). That is,

$$E_{sm}(T) = \sum_{i} \sum_{j \in \mathcal{N}_T(i)} \|T_i - T_j\|^2, \tag{25}$$

where $\mathcal{N}_T(i)$ represents the neighborhood of the texel i. While updating the texture, we choose ρ (appearing in the definition of R and hence $\bar{R}$ and $\bar{E}$) to be a truncated quadratic instead of the quadratic used for model fitting. This helps to avoid the contamination of the hand texture by background color whenever the hand model fit is not very accurate. The minimization of the function $E_{texture}(T)$ with $T \in \mathbb{T}$ can be done efficiently using iteratively reweighed least-squares. The smoothness term effectively diffuses the colour to texels that do not directly contribute to the image, either because they are associated to a occluded part (Fig.6) or because of texture aliasing artifacts. To improve robustness we also remove pixels near occlusion boundaries from the integration domain Ω when computing $E(\theta, L, T)$, and we bound the difference between the texture in the first frame and in subsequent texture estimates.

Finally, although cast shadows provide useful cues, they are not explicitly modeled in our approach. Instead, they are modeled as texture. Introducing cast shadows in our continuous optimization framework is not straightforward as it would require computation of related terms in the functional gradient. Nevertheless, despite the lack of cast shadows in our model, our experiments demonstrate good tracking results.

4 Experimental Results

We applied our hand pose tracking approach to sequences with large amounts of articulation and occlusion, and to two sequences used in [14], for comparison with a state-of-the-art hand tracker. Finally, we also report quantitative results on a synthetic sequence as well as a real sequence for which ground truth data were obtained. While the computational requirements depends on image resolution, for 640×480 images, our implementation in C++ and Matlab takes approximately 40 seconds per frame on a 2Ghz workstation.

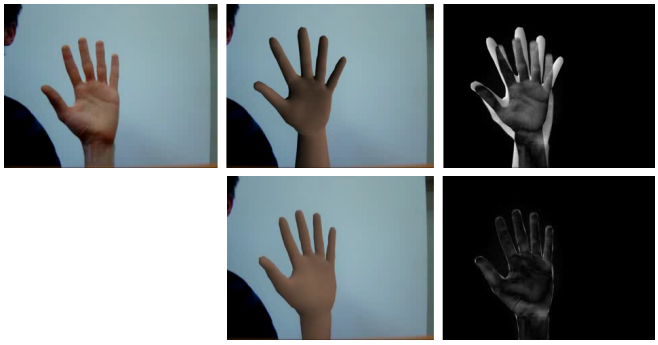


Fig. 7. Estimation of the pose, lighting and morphological parameters in the frame 1. The first row shows the input image, the synthetic image given rough manual initialization, and its corresponding residual. The second row shows the result obtained after convergence

4.1 Initialization

A good initial guess for the pose is required for the first frame. One could use a discriminative method to obtain a rough initialization (e.g., [3]). However, as this is outside the scope of our approach at present, so we simply assume prior knowledge. In particular, the hand is assumed to be roughly parallel to the image plane at initialization (Fig. 7). We further assume that the hand color (albedo) is initially constant over the hand surface, so the initial hand appearance is solely a function of pose and shading. The RGB hand color, along with the hand pose, the morphological parameters, and the illuminant are estimated simultaneously using the quasi-Newton method (see row 2 of Fig. 7 (see the video 1). The background image or its histogram are also provided by the user. Once the morphological parameters are estimated in frame 1, they remain fixed for the rest of the sequence.

4.2 Finger Bending and Occlusion Forces

In the first sequence (Fig. 8 and video 2) each finger bends in sequential order, eventually occluding other fingers and the palm. The background image is static, and was obtained from a frame where the hand was not visible. Each frame has 640×480 pixels. Note that the edges of the synthetic hand image match the edges in the input image, despite the fact that we did not use any explicit edge-related term in the objective function. Misalignment of the edges in the synthetic and observed images creates a large residual in the area between these edges and produces occlusion forces on the hand pointing in the direction that reduces the misalignment.

To illustrate the improvements provided by occlusion forces, we simply remove the occlusion-forces when computing the energy gradient. The resulting algorithm is unable track the hand for any significant period.

It is also interesting to compare the minimization of the energy in (18) with the approach outlined in Sec. 2.2, which sums errors over the surface of the 3D mesh.

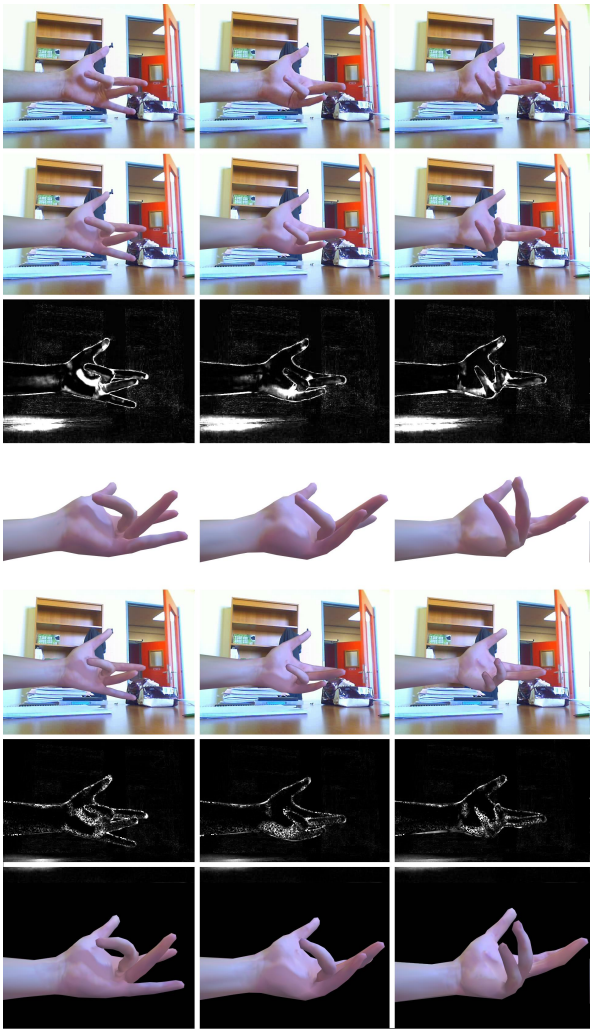


Fig. 8. Improvement due to self-occlusion forces: Rows 2-4 are obtained using our method. Rows 5-7 are obtained by summing the residual on the triangulated surface. Each row shows, from top to bottom: (1) the observed image; (2) the final synthetic image; (3) the final residual image; (4) a synthetic side view at $45°$; (5) the final synthetic image with residual summed on surface; (6) the residual for visible points on the surface, and (7) the synthetic side view.

These 3D points are uniformly distributed on the surface and their binary visibility is computed at each step of the quasi-Newton process. To account for the change of summation domain (from image to surface), the error associated to each point is weighted by the inverse of the ratio of surface areas between the 3D triangular face and its projection. The errors associated with the background are also taken into account, to remain consistent with the initial cost function. This also prohibits the hand from receding from the camera so that its projection shrinks. We included occlusion forces between the hand and background to account for variations of the background visibility while computing the energy gradient. The functional computed in both approaches would ideally be

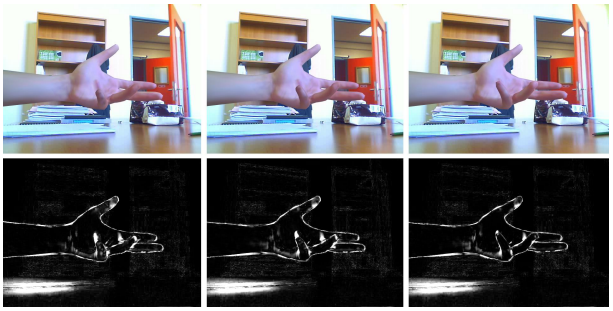


Fig. 9. Recovery from failure of the sum-on-surface method (Sec. 4.2) by our method with occlusion forces.

equal; their difference is bounded by the error induced by the discretization of integrals. For both methods we used a maximum of 100 iterations, which was usually sufficient for convergence.

The alternative approach produces reasonable results (Fig. 8 rows 5-7) but fails to recover the precise location of the finger extremities when they bend and occlude the palm. This is evident in column 3 of Fig. 8 where a large portion of the little finger is missing. Our approach (rows 2-4) yields accurate pose estimates through the entire sequence. The alternative approach fails because the hand/background silhouette is not particularly informative about the position of fingertips when fingers are bending. The residual error is localized near the outside extremity of the synthesized finger. Self-occlusion forces are necessary to pull the finger toward this region.

We further validate our approach by choosing the hand pose estimated by the alternative method (Fig. 8, column 3, rows 5-7) as an initial state for our tracker with occlusion forces (Fig. 9 and video 3). The initial pose and the corresponding residual are shown in column 1. The poses after 25 and 50 iterations are shown in columns 2 and 3, at which point the pose is properly recovered.

4.3 Stenger Sequences [14]

The second and third sequences (Fig. 10, Fig. 11, video 4 and 5) were provided by Stenger for direct comparison with their state-of-the-art hand tracker [14]. Both sequences are 320×240 pixels. In Sequence 2 the hand opens and closes while rotating. In Sequence 3 the index finger is pointing while the hand rotates in a rigid manner. Both sequences exhibit large inter-frame displacements which makes it difficult to predict good initial guesses for minimization. The fingers also touch one another often. This is challenging because our optimization framework does not prohibit the interpenetration of hand parts. Finally, the background in Sequence 3 is dynamic so the energy function has been expressed using (4).

For computational reasons, Stenger et al. [14] used a form of state-space dimension reduction, adapted to each sequence individually (8D for Sequence 2 - 2 for articulation and 6 for global motion - and 6D rigid

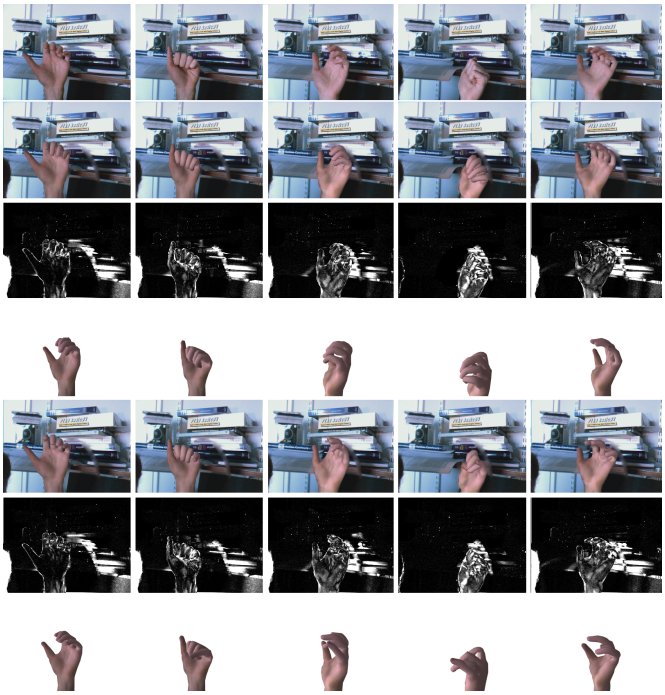


Fig. 10. Sequence 2: Each row shows, from top to bottom: (1) the observed image, (2) the final synthetic image with limited pose space, (3) the final residual image, (4) the synthetic side view with an angle of $45°$, (5) the final synthetic image with full pose space, (6) the residual image and (7) the synthetic side view.

movement for Sequence 3). We tested our algorithm both with and without pose-space reductions (respectively rows 2-4 and 5-7). For pose-space reduction, linear inequalities were defined between pairs or triplet of joint angles. This allowed us to limit the pose space while keeping enough freedom of pose variation to make fine registration possible. We did not update the texture for those sequences after the first frame. With pose-space reduction our pose estimates are qualitatively similar to Stenger's, although smoother through time. Without pose-space reduction, Stenger's technique becomes computationally impractical while our approach continues to produce good results. The lack of need for offline training and sequence-specific pose-space reduction are major advantages of our approach.

4.4 Occlusion

Sequence 4 (Fig. 12 and video 6) demonstrates the tracking of two hand, with significant occlusions as the left hand progressively occludes the right. Texture and pose parameters from both hands are simultaneously estimated, and self-occlusions of one hand by another are modeled just as was done with a single hand. Fig. 12 shows that the synthetic image looks similar to the observed image, due to the shading and texture models. In row 2, the ring finger and the little finger of the right hand are still properly registered despite the large occlusion. Despite the translational movement of the hands,

Fig. 11. Sequence 3: Each row shows, from top to bottom: (1) the observed image; (2) the final synthetic image with limited pose space; (3) the final residual image; (4) the synthetic side view with an angle of $45°$; (5) the final synthetic image with full pose space; and (6) the residual image, the synthetic side view.

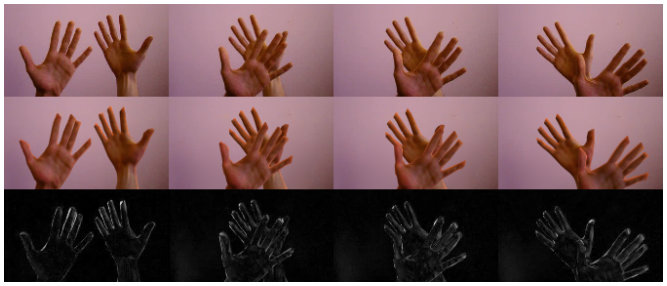


Fig. 12. Sequence 4: Two hands. The rows depicts 4 frames of the sequence, corresponding to (top) the observed images, (middle) images synthesized using the estimated parameters, and (bottom) the residual images.

we kept all 28 DOFs on each hand while performing pose estimation.

4.5 Quantitative Performance on Natural Images

While existing papers on hand pose tracking have relied mainly on qualitative demonstrations for evaluation, here we introduce quantitative measures on a binocular

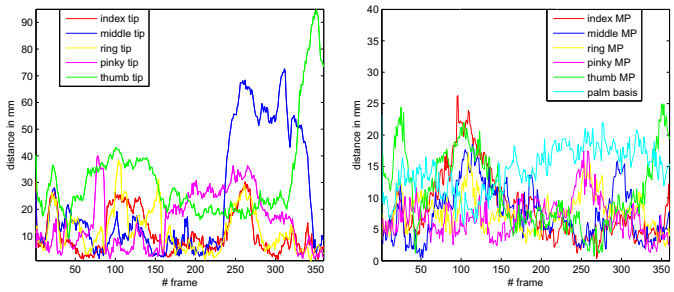


Fig. 13. Distance between reconstructed and ground truth 3D locations: (left) The distances for each of the finger tips; (right) The distances for each of the metacarpophalangeal point and the palm basis.

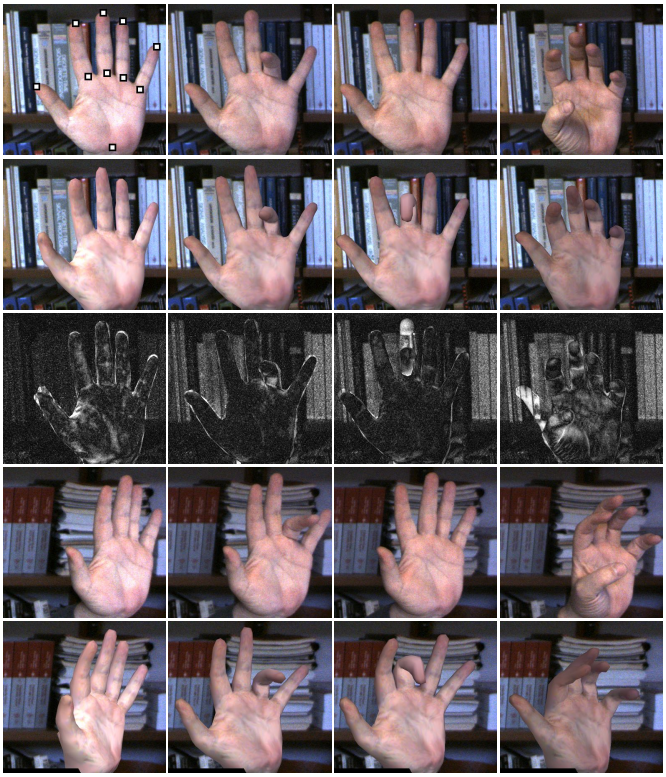


Fig. 14. Binocular Video: The columns show frames 2, 220, 254 and 350. Each row, from top to bottom, depicts: (1) the image from the right camera; (2) the corresponding synthetic image; (3) the residual for the right view; (4) the observed left image (not used for tracking); and (5) the synthetic image seen from the left camera.

sequence (see video 7) from which we can estimate 3D ground truth at a number of fiducial points.

The hand was captured with two synchronized, calibrated cameras (see rows 1 and 4 in Fig. 14). To obtain ground truth 3D measurements on the hand, we manually identified at each frame, the locations of 11 fiducial points on the hand, i.e., each finger tip, each metacarpophalangeal (MP)joint, and the base of the palm. These locations are shown in the top-left image of Fig. 14. From these 2D locations in two views we obtain 3D ground truth by triangulation [33].

The hand was tracked using just the right image sequence. For each of the 11 fiducial points we then computed the distance between the 3D ground truth and the estimated position on the reconstructed hand surface. Because the global size of our model does not necessarily match the true size of the hand, in each frame we allow a simple scaling transform of the estimated hand to minimize the sum of squared distances between the 3D ground truth points and the recovered locations. The residual Euclidean errors for each point, after this scaling, are shown in Fig. 13.

Errors for the thumb tip are large for frames 340-360, and errors for the middle finger are large for frames 240-350. During those periods the optimization lost track of the positions of the middle finger (see Fig.14 column 3) and the thumb (see Fig.14 column 4). Thumb errors are also large between frames 50 and 150, despite the fact that estimated pose appears consistent with the right view used for tracking (see residuals in Fig.14 row 3, columns 1-3). This reflects the intrinsic problem of monocularly estimating depth. The last row in Fig.14 shows the hand model synthesized from the left camera viewpoint, for which the depth errors are evident.

4.6 Quantitative Performance on Synthetic Images

We measured quantitative performance using synthetic images. The image sequence was created with the Poser© software (see video 8). The hand model used in Poser differs somewhat from our model both in the the triangulated surface and the joint parameterization. To obtain realistic sequences we enable ambient occlusion and cast shadows in the rendering process, neither of which are explicitly modeled in our tracker. Row 1 of Fig.15 shows several frames from the sequence.

A quantitative evaluation of estimated pose is obtain by measuring, for each frame, a distance between the reconstructed 3D surface and the ground truth 3D surface. For visualization of depth errors over the visible portion of the hand surface, we define a distance image D_s:

$$D_s(p) = \inf_{q \in S_g} \left(\Pi_{S_s}^{-1}(p), q\right) \text{ if } p \in \Omega_s, \ \infty \text{ otherwise} \quad (26)$$

where S_g is the ground truth hand surface, S_s is the reconstructed hand surface, and Ω_s is the silhouette of the surfaces S_s in the image plane. In addition, $\Pi_{S_s}^{-1}$ is the *back projection function* onto the surface S_s; i.e., the function that maps each point p in Ω_s to the corresponding point on the 3D surface S_s. To cope with scaling uncertainty, as discussed above, we apply a scaling transform on the reconstructed surface whose fixed point is the optical center of the camera, such that mean depth of the visible points in S_s is equal to the mean depth of points in S_g; i.e.,

$$\frac{\int_{p \in \Omega_s} \Pi_{S_s}^{-1}(p) \cdot \hat{z} dp}{\int_{p \in \Omega_s} 1 \, dp} = \frac{\int_{p \in \Omega_s} \Pi_{S_g}^{-1}(p) \cdot \hat{z} dp}{\int_{p \in \Omega_g} 1 \, dp} \quad (27)$$

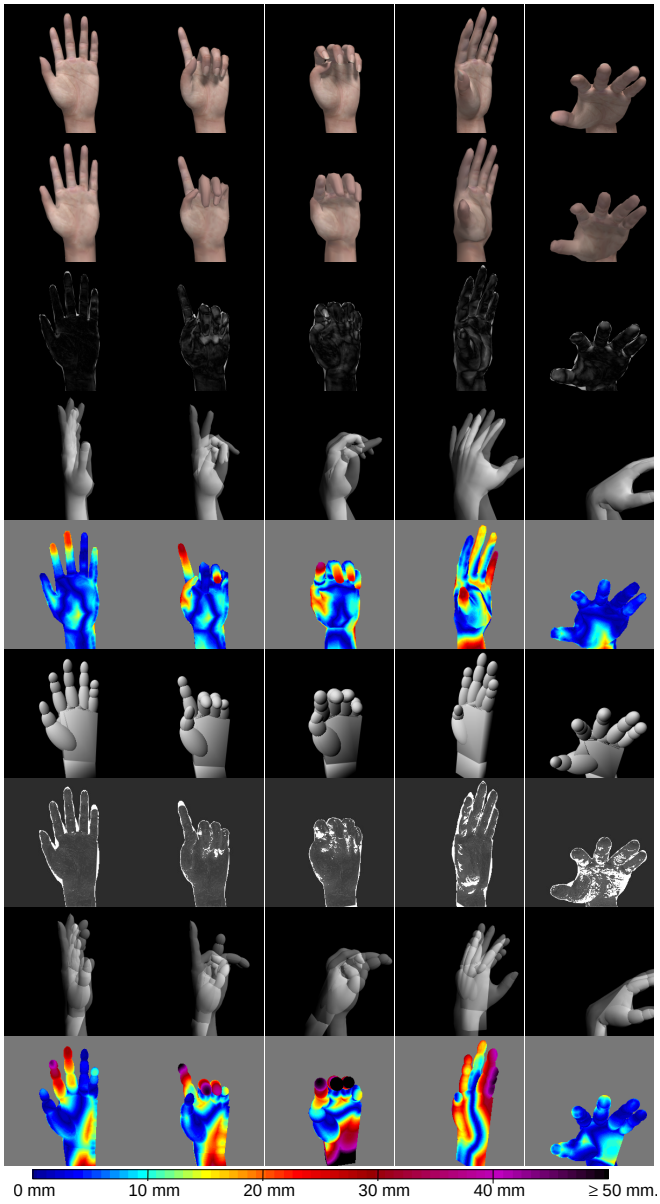


Fig. 15. Synthetic sequence: The first row depicts individual frames. Rows 2-5 depict tracking results: i.e., (2) estimated pose; (3) residual image; (4) side view superimposing ground truth and the estimated surfaces; and (5) distance image D_s. Rows 6-9 depict results obtained with the method in [10]: i.e., (6) generative image of [10] at convergence; (7) log-likelihood per pixel according to [10]; (8) a side view superimposing ground truth and estimated surfaces; and (9) distance image D_s.

where $\hat{z}$ is aligned with the optical axis, Ω_g is the silhouette in the ground truth image, and $\Pi_{S_g}^{-1}$ is the *back projection function* onto the surface S_g.

Figure 15 depicts the performance of our tracker and the hand tracker described in [10]. The current method is shown to be significantly more accurate. Despite the fact that the model silhouette matches the ground truth in the row 3 of Fig. 15, the errors over the hand surface

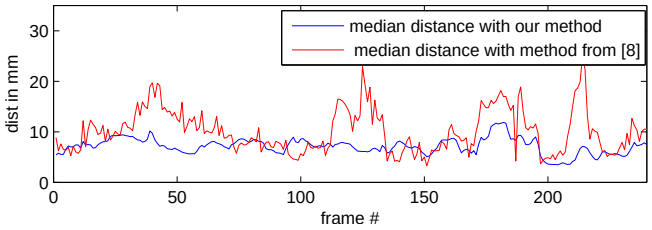


Fig. 16. Comparison, trough all frames of the video, of the median of the distances $D_s(p)$ over the point in Ω_s with our method and the one in [10].

are significant; e.g., see the middle and ring fingers. The method in [10] relies on a matching cost that is solely a function of the hand silhouette, and is therefore subject to ambiguities. Many of these ambiguities are resolved with the use of shading and texture. Finally, Fig. 16 plots the median distance $D_s(p)$ over points in Ω_s is shown as a function of time for both methods. Again, note that our method outperforms that in [10] for most frames.

5 Discussion

We describe a novel approach to the recovery of 3D hand pose from monocular images. In particular we build a detailed generative model that incorporates a polygonal mesh to accurately model shape, and a shading model to incorporate important appearance properties. We estimate the model parameters (i.e., shape, pose, texture and lighting) are determined through a variational formulation. A rigorous mathematical formulation of the objective function is provided in order to properly deal with occlusions. Estimation with a quasi-Newton technique is demonstrated with various image sequences, some with ground truth, for a hand model with 28 joint degrees of freedom.

The model provides state-of-the-art pose estimates, but future work remains in order to extend this work to an entirely automatic technique that can be applied to long image sequences. In particular there remain several situations in which the technique described here is prone to converge to a poor local minima and therefore lose track of the hand pose: 1) There are occasional reflection ambiguities which are not easily resolved by the image information at a single time instant; 2) Fast motion of the fingers sometimes reduces the effectiveness of our initial guesses based on a simple form of extrapolation from previous frames; 3) Entire fingers are sometimes occluded. In this case there will be no image support for a pose estimate of that region of the hand; 4) Cast shadows are only modeled as texture, so dynamic cast shadows can be difficult for the shading model to handle well (as can be seen in some of the experiments above). Modeling cast shadows is tempting, but it would mean a discontinuous residual function of θ along synthetic shadow borders, which are likely very complicated to handle analytically; and 5) Collisions or interpenetration

of the model fingers creates aliasing along the intersection boundaries in the discretized synthetic image. The change of visibility along self-collisions could be treated using forces along these boundaries, but this is not currently done and therefore the optimization may not converge in these cases. A solution might be to add non linear non-convex constraints that would prevent collisions.

When our optimization loses track of the hand pose, we have no mechanism to re-initialize the pose, or explore multiple pose hypotheses, e.g., with discriminative methods. Dynamical models of typical hand motions, and/or latent models of typical hand poses, may also help resolve some of these problems due to ambiguities and local minima. Last, but not least, the use of discrete optimization methods involving higher order interactions between hand-parts [34] with explicit modeling of visibilities [35] may be a promising approach for improving computational efficiency and for finding better energy minima.

APPENDIX A
IDENTIFYING OCCLUSION FORCES

As depicted in Fig. 17, we consider the residual along a segment $\overline{\bar{V}_m\bar{V}_n}$ joining two adjacent vertices along the occlusion boundary, $\bar{V}_m \equiv (x_m, y_m)$ and $\bar{V}_n \equiv (x_n, y_n)$, with normal $\hat{n} \equiv (\hat{n}_x, \hat{n}_y)$. Without loss of generality, we assume that $\hat{n}_y > 0$, and $\hat{n}_y > |\hat{n}_x|$; i.e., the edge is within $45°$ of the horizontal image axis, and the occluding side lies below the boundary segment. Accordingly, the antialiasing weights in (15) become

$$w(\mathbf{x}) \;=\; \frac{(\mathbf{x} - q(\mathbf{x})) \cdot \hat{n}}{\hat{n}_y} \;. \tag{28}$$

To compute $\nabla_\theta \bar{E}$ we need the partial derivatives of $\bar{R}$ with respect to θ (17). Some terms are due to changes in R while others are due to the weights w. (For notational brevity we omit the parameters θ, L and T when writing the residual function R). Our goal here is to associate the terms of the derivative due to the weights with the second line in (13). Denote this term C:

$$C \;=\; \sum_{\mathbf{x} \in \mathcal{A} \cap \mathbb{N}^2} \frac{\partial w(\mathbf{x})}{\partial \theta_j} [R(\mathbf{x}) - R(q(\mathbf{x}))] \;. \tag{29}$$

Let $q_y(\mathbf{x})$ be the vertical projection of the point $\mathbf{x}$ onto the line that contains the segment $\overline{\bar{V}_m\bar{V}_n}$. Also let $(\vec{x}, \vec{y})$ denote the coordinate axes of the image plane. For any point $\mathbf{x}$ in $\mathcal{A}$ one can show that

$$w(\mathbf{x}) \;=\; (\mathbf{x} - q_y(\mathbf{x})) \cdot \vec{y} \;. \tag{30}$$

Differentiating w with respect to θ_i gives

$$\frac{\partial w(\mathbf{x})}{\partial \theta_j} = -\frac{\partial q_y(\mathbf{x})}{\partial \theta_j} \cdot \vec{y} \;. \tag{31}$$

The vertically projected point $q_y(\mathbf{x})$ lies in the segment $\overline{\bar{V}_m\bar{V}_n}$, and therefore there exists $t \in [0, 1]$ such that

$$q_y(\mathbf{x}) \;=\; (1-t)\bar{V}_m + t\bar{V}_n \;. \tag{32}$$

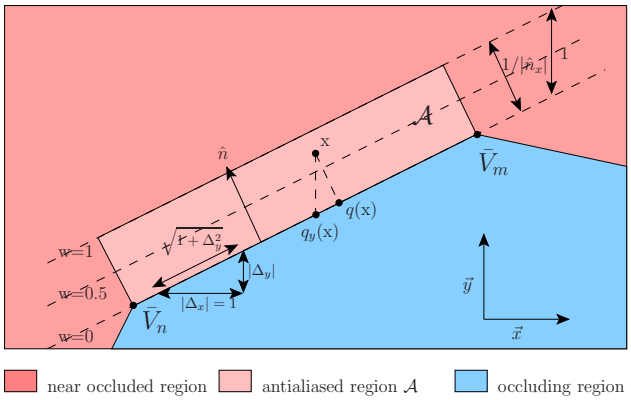


Fig. 17. Antialiasing weights for points in the vicinity of the segment.

We can then differentiate $q_y(\mathbf{x})$ with respect to θ_j, i.e.,

$$\frac{\partial q_y(\mathbf{x})}{\partial \theta_j} \;=\; \frac{\partial t}{\partial \theta_j}(\bar{V}_n - \bar{V}_m) + (1-t)\frac{\partial \bar{V}_m}{\partial \theta_j} + t\frac{\partial \bar{V}_n}{\partial \theta_j} \;. \tag{33}$$

Now, let v_j denote the curve speed at x when θ_j varies. It is the partial derivative of the curve Γ_θ with respect to a given pose parameter θ_j in θ, i.e.,

$$v_j(\mathbf{x}) \;=\; (1-t_{\mathbf{x}})\frac{\partial \bar{V}_{m_{\mathbf{x}}}}{\partial \theta_j} + t_{\mathbf{x}}\frac{\partial \bar{V}_{n_{\mathbf{x}}}}{\partial \theta_j} \;. \tag{34}$$

Because $(\bar{V}_n - \bar{V}_m) \cdot \hat{n} = 0$, with the curve speed v_j above, we obtain the following from (33)

$$\hat{n} \cdot \frac{\partial q_y(\mathbf{x})}{\partial \theta_j} \;=\; \hat{n} \cdot v_j(q_y(\mathbf{x})) \;. \tag{35}$$

Because $\vec{x} \cdot \frac{\partial q_y(\mathbf{x})}{\partial \theta_j} = 0$ we obtain

$$\hat{n} \cdot \frac{\partial q_y(\mathbf{x})}{\partial \theta_j} \;=\; (\hat{n}_x\vec{x} + \hat{n}_y\vec{y}) \cdot \frac{\partial q_y(\mathbf{x})}{\partial \theta_j} \;=\; \hat{n}_y\Big(\frac{\partial q_y(\mathbf{x})}{\partial \theta_j} \cdot \vec{y}\Big) \;. \tag{36}$$

Therefore

$$\frac{\partial w(\mathbf{x})}{\partial \theta_j} \;=\; -\frac{\partial q_y(\mathbf{x})}{\partial \theta_j} \cdot \vec{y} \;=\; -\frac{\hat{n}}{\hat{n}_y} \cdot \frac{\partial q_y(\mathbf{x})}{\partial \theta_j} \;=\; -\frac{\hat{n} \cdot v_j(q_y(\mathbf{x}))}{\hat{n}_y} \tag{37}$$

We can rewrite C as follow:

$$C \;=\; \frac{1}{\hat{n}_y} \sum_{\mathbf{x} \in \mathcal{A} \cap \mathbb{N}^2} -[R(\mathbf{x}) - R(p(\mathbf{x}))]\, \hat{n} \cdot v_j(q_y(\mathbf{x})) \;. \tag{38}$$

We now show that with some approximations we can associate this term with the second term in (13). We assume R and R^+ to be smooth. Therefore

$$R(q(\mathbf{x})) \;\approx\; R(q_y(\mathbf{x})) \tag{39}$$

$$R(\mathbf{x}) \;\approx\; R^+(q_y(\mathbf{x})) \;. \tag{40}$$

Therefore, given the definition of the *occlusion forces* (13), C can be approximated by $\tilde{C}$ defined as:

$$\tilde{C} = \frac{1}{\hat{n}_y} \sum_{\mathbf{x} \in \mathcal{A} \cap \mathbb{N}^2} -f_{oc}(q_y(\mathbf{x})).v_j(q_y(\mathbf{x})) \;. \tag{41}$$

The division by $max(|\hat{n}_x|, |\hat{n}_y|)$ in our definition of the weight (15) ensures that within each vertical line there is

a single point $\mathbf{x}$ with integer coordinates and its weight in $[0, 1)$. This appears more clearly in (30).

Given a vertical line with constant x, this point has the coordinates $\mathbf{x} = (x, \lceil \bar{y}(x) \rceil)$ where

$$\bar{y}(x) = y_m + (x - x_m)\Delta_y \ , \tag{42}$$

with

$$\Delta_y = (y_n - y_m)/(x_n - x_m) \tag{43}$$

being the slope of the segment. From (16) we obtain

$$\mathcal{A} \cap \mathbb{N}^2 = \{(x, \lceil \bar{y}(x) \rceil) | x \in \mathbb{N}, p((x, \lceil \bar{y}(x) \rceil) \in \overline{\bar{V}_m \bar{V}_n}\} \ . \tag{44}$$

We assume now that the condition that the point x should orthogonally project into the segment (i.e., $q((x, \lceil \bar{y}(x) \rceil) \in \overline{\bar{V}_m \bar{V}_n}$) can be approximated by the condition $x \in [x_n, x_m]$ as the line is within $45°$ of vertical. The resulting approximate anti-aliased region is no longer rectangular but a parallelogram with two vertical sides. After discretization on the grid this might result, in some cases, of a point being neglecting near the extremities of the segment.

$$\mathcal{A} \cap \mathbb{N}^2 \approx \{(x, \lceil \bar{y}(x) \rceil) | x \in \{\lceil x_n \rceil, \ldots, \lfloor x_m \rfloor\}\} \ . \tag{45}$$

This approximation of the anti-aliased region and the fact that $q_y(x, \lceil \bar{y}(x) \rceil) = (x, \bar{y}(x))$ allows us to approximate $\tilde{C}$ by $\tilde{C}_2$ as follow:

$$\tilde{C}_2 = \frac{1}{\hat{n}_y} \sum_{x=\lceil x_n \rceil}^{\lfloor x_m \rfloor} -f_{\text{oc}}\big((x, \bar{y}(x))\big) \cdot v_j\big((x, \bar{y}(x))\big) \ . \tag{46}$$

Now let $\Delta_V = (1, \Delta_y)$ denote the 2D displacement along the segment $\overline{\bar{V}_m \bar{V}_n}$ when x is incremented by one. Then

$$|\Delta_V| = \sqrt{1 + \Delta_y^2} = \frac{1}{\hat{n}_y} \ . \tag{47}$$

We also introduce $t = x - \lceil x_n \rceil$, $N = \lfloor x_m \rfloor - \lceil x_n \rceil$ and $\epsilon = \lceil x_n \rceil - x_n$. We obtain, after some derivation,

$$\tilde{C}_2 = |\Delta_V| \sum_{t=0}^{N-1} -f_{\text{oc}}(\bar{V}_n + (t+\epsilon)\Delta_V) \cdot v_j(\bar{V}_n + (t+\epsilon)\Delta_V) \ . \tag{48}$$

This last approximation of C can be identified as a discrete approximation of the integral, along $\overline{\bar{V}_m \bar{V}_n}$, of

$$f(\mathbf{x}) = -f_{\text{oc}}(\mathbf{x}) \cdot v_j(\mathbf{x}) \ . \tag{49}$$

This term corresponds to the contribution of the segment in the second term of (13). We demonstrated that the implementation based on a discrete image domain with anti-aliasing yields, after differentiation, terms that are consistent (up to an approximation) with the *occlusion forces* that were obtained by differentiating the objective function defined on the continuous image domain.

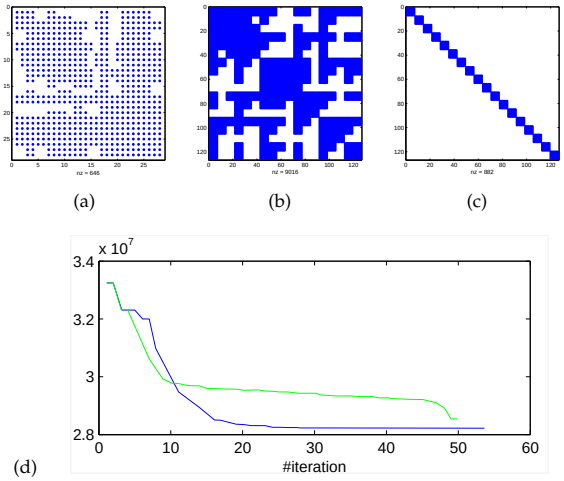


Fig. 18. Sparsity structure: (a) $\theta \in \mathbb{R}^{28}$ (b) $\theta \in \mathbb{R}^{126}$ (c) approximated structure. The bottom plot (d) shows the decrease in the energy functional with respect to number of functional evaluations. (blue:adapted BFGS; green:conventional BFGS)

APPENDIX B
BLOCKWISE BFGS UPDATES

Quasi-Newton methods rely on a good approximation to the Hessian of the objective function. Better approximations often lead to faster convergence. Here we approximate the objective function Hessian using a variant of the BFGS update method. It was adapted to exploit the partial independence of separated fingers. The resulting sparseness of the Hessian leads to the definition of a *Blockwise BFGS update*. This blockwise update increased the convergence rate by a factor ranging from of 3 to 6.

To obtain a good convergence rate even during the first iterations, it is important to initialize the approximate Hessian well. Rather than using the identity matrix, as is often done, we used a scaled version of the matrix $J_\theta^{\bar{V}\,t} J_\theta^{\bar{V}}$ where $J_\theta^{\bar{V}}$ is the Jacobian of projected vertices with respect to θ. This favors the displacements in the depth direction for which the gradient is small due to weak support from the image data.

Because the contributions to the overall cost of two well-separated fingers are independent, the true Hessian will not be fully populated but will exhibit blocks of zeros (Fig. 18.a). This sparsity is accentuated if, instead of using joints angles, we individually parameterize the pose of each bone using a 7D vector, comprising a quaternion and a translation vector such that $\theta \in \mathbb{R}^{126}$ (The bones of the wrist and the arm are rigidly fixed and therefore we need not represent one of the two). As shown in Fig. 18b), non-zero entries of the 126×126 Hessian appear on 7×7 blocks. Each off-diagonal block corresponds to a pair of hand parts that either occlude one another or share some facets in their influence area for the pose space deformation.

To exploit the Hessian sparsity, with quaternions used to parameterize bones, without the need for further non-

linear constraints, we first decompose the function $E(\theta)$ into $E(\theta) = E_q(Q(\theta))$, where Q maps the joint-angle representation to quaternions. The Hessian $\frac{\partial^2 E_q}{\partial^2 \theta}$ is then approximated by $(\frac{\partial Q}{\partial \theta})^t H_q (\frac{\partial Q}{\partial \theta})$ where $H_q = (\frac{\partial^2 E_q}{\partial^2 Q})$.

At each step, we refine the Hessian approximation H_q with an adapted BFGS update. We approximate the structure of H_q by assuming complete independence between parts of the hand. This produces block-diagonal structure (see Fig. 18.c) where non-zero entries occur in 7×7 blocks along the diagonal. The standard BFGS update does not exploit this structure, and would otherwise populate the entire matrix. Using the BFGS formula, rather than update H_q, we only update the non-zero 7×7 blocks on the diagonal independently. About 7 gradient evaluations are then necessary to obtain a reasonable local approximation of the Hessian, while the standard BFGS method would require about 28 evaluations. This has a direct impact on the convergence rate of the optimization. The method induces more zeros than in the true Hessian but still leads to significant improvement over the standard BFGS update. As we keep performing increments on θ during the optimization, we do not need to add nonlinear constraints that enforce validity of relative poses of connected bones that would be necessary if the increments were done in the quaternion representation space. The improvement in the minimization process, in terms of the number of objective function evaluations, is shown in Fig. 18, where we estimated the pose for a single frame.

Acknowledgment

This work was financially supported by grants to DJF from NSERC Canada and the Canadian Institute for Advanced Research (CIfAR).

References

[1] S. Lu, D. Metaxas, D. Samaras, and J. Oliensis, "Using multiple cues for hand tracking and model refinement," in *CVPR*, 2003, pp. II: 443–450. 1, 2

[2] A. O. Bălan, M. Black, H. Haussecker, and L. Sigal, "Shining a light on human pose: On shadows, shading and the estimation of pose and shape," in *ICCV*, 2007, pp. 1–8. 1

[3] R. Rosales, V. Athitsos, L. Sigal, and S. Scarloff, "3D hand pose reconstruction using specialized mappings," in *ICCV*, 2001, pp. I:378–385. 1, 8

[4] N. Shimada, "Real-time 3-d hand posture estimation based on 2-d appearance retrieval using monocular camera," in *RATFG*, 2001, pp. 23–30. 1

[5] V. Athitsos and S. Sclaroff, "Estimating 3d hand pose from a cluttered image." in *CVPR*, 2003, pp. 432–442. 1

[6] T. E. de Campos and D. W. Murray, "Regression-based hand pose estimation from multiple cameras," in *CVPR*, 2006, pp. I:782–789. 1

[7] T. Heap and D. Hogg, "Towards 3D hand tracking using a deformable model," in *FG*, 1996, pp. 140–145. 2

[8] J. Rehg and T. Kanade, "Model-based tracking of self-occluding articulated objects," in *ICCV*, 1995, pp. 612–617. 2

[9] B. Stenger, P. R. S. Mendonça, and R. Cipolla, "Model-based hand tracking using an unscented kalman filter," in *BMVC*, 2001, pp. I:63–72. 2

[10] M. de La Gorce and N. Paragios, "A variational approach to monocular hand-pose estimation," *CVIU*, vol. 114, no. 3, pp. 363–372, 2010. 2, 12

[11] E. Sudderth, M. Mandel, W. Freeman, and A. Willsky, "Visual hand tracking using nonparametric belief propagation," in *CVPR*, 2004, pp. 189–196. 2

[12] H. Ouhaddi and P. Horain, "3D hand gesture tracking by model registration," in *Proc. International Workshop on Synthetic-Natural Hybrid Coding and 3D Imaging*, 1999, pp. 70–73. 2

[13] Y. Wu, J. Y. Lin, and T. S. Huang, "Capturing natural hand articulation." in *ICCV*, 2001, pp. 426–432. 2

[14] B. Stenger, "Model-based hand tracking using a hierarchical bayesian filter," Ph.D. dissertation, University of Cambridge, UK, March 2004. 2, 8, 9

[15] M. Brubaker, L. Sigal, and D. Fleet, "Video-based people tracking," in *Handbook of Ambient Intelligence and Smart Environments*. Springer, 2009. 2

[16] A. O. Bălan, L. Sigal, M. J. Black, J. E. Davis, and H. W. Haussecker, "Detailed human shape and pose from images," in *CVPR*, 2007, pp. 1–8. 2

[17] M. Bray, E. Koller-Meier, L. Van Gool, and N. N. Schraudolph, "3d hand tracking by rapid stochastic gradient descent using a skinning model," in *European Conference on Visual Media Production (CVMP)*, 2004, pp. 59–68. 2

[18] *Joint-dependent local deformations for hand animation and object grasping*. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 1988. 2

[19] J. P. Lewis, M. Cordner, and N. Fong, "Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation," in *SIGGRAPH*, 2000, pp. 165–172. 2

[20] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *SIGGRAPH*, 1999, pp. 187–194. 3, 4

[21] M. Soucy, G. Godin, and M. Rioux, "A texture-mapping approach for the compression of colored 3d triangulations." *The Visual Computer*, vol. 12, no. 10, pp. 503–514, 1996. 3

[22] C. Hernández, "Stereo and silhouette fusion for 3d object modeling from uncalibrated images under circular motion," Ph.D. dissertation, ENST, May 2004. 3

[23] H. Sidenbladh, M. J. Black, and D. J. Fleet, "Stochastic tracking of 3d human figures using 2d image motion," in *ECCV*, 2000, pp. II:702–718. 4

[24] N. Paragios and R. Deriche, "Geodesic active regions for supervised texture segmentation," in *ICCV*, 1999, pp. II:926–932. 6

[25] G. Unal, A. Yezzi, and H. Krim, "Information-theoretic active polygons for unsupervised texture segmentation," *IJCV*, vol. 62, no. 3, pp. 199–220, 2005. 6

[26] P. Gargallo, E. Prados, and P. Sturm, "Minimizing the reprojection error in surface reconstruction from images," in *ICCV*, 2007, pp. 1–8. 6

[27] A. Delaunoy, E. Prados, P. Gargallo, J.-P. Pons, and P. Sturm, "Minimizing the multi-view stereo reprojection error for triangular surface meshes," in *BMVC*, 2008. 6

[28] P. V. Sander, H. Hoppe, J. Snyder, and S. J. Gortler, "Discontinuity edge overdraw," in *Proc. Symposium on Interactive 3D graphics (SI3D)*, 2001, pp. 167–174. 6, 7

[29] F. C. Crow, "A comparison of antialiasing techniques," *IEEE Comput. Graph. Appl.*, vol. 1, no. 1, pp. 40–48, 1981. 7

[30] L. Carpenter, "The a-buffer, an antialiased hidden surface method," *SIGGRAPH*, vol. 18, no. 3, pp. 103–108, 1984. 7

[31] A. Griewank, *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000. 7

[32] A. Conn, N. Gould, and P. Toint, *Trust-region methods*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000. 7

[33] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. New York: Cambridge University Press, 2000. 11

[34] N. Komodakis and N. Paragios, "Beyond pairwise energies: Efficient optimization for higher-order mrfs," in *CVPR*, 2009, pp. 2985 – 2992. 13

[35] C. Wang, M. de La Gorce, and N. Paragios, "Segmentation, ordering and multi-object tracking using graphical models," in *ICCV*, 2009, pp. 747–754. 13