The Vitruvian Manifold: Inferring Dense Correspondences for One-Shot Human Pose Estimation

Jonathan Taylor[†⋆] Jamie Shotton[†] Toby Sharp[†] Andrew Fitzgibbon[†]

[†]Microsoft Research Cambridge [⋆]University of Toronto

The supplementary material for our CVPR 2012 paper *The Vitruvian Manifold: Inferring Dense Correspondences for One-Shot Human Pose Estimation* comprises this document and the accompanying video.

1. Supplementary Results

The accompanying video illustrates the inferred correspondences and convergence of the optimization algorithm for several poses, and additionally gives demonstrations of how the algorithm performs on extended sequences containing complex motion. In addition to this we include in Fig. 1 a qualitative comparison of the skeletons inferred by the algorithms of [1, 2] and the Microsoft Kinect for Windows software development kit (SDK) to accompany the quantitative results in the main paper.

2. Implementation Appendix

While the paper contains a full description of the algorithm, this section includes further details of the specific implementation we used for the interested reader, including parameter settings (Fig. 2) and many of the derivatives of the energy function required by the L-BFGS optimizer.

2.1. Prior

Our prior consists of: (i) a Gaussian on pose, learned from motion capture data; and (ii) a heuristic term that discourages self-intersection. These each provide a modest quantifiable increase in accuracy, and qualitatively encourage more realistic poses.

The Gaussian over pose has full covariance over the rotation degrees of freedom and a single independent element corresponding to the scale degree of freedom. There is no prior over global translation as the actor can appear at any point in the scene. See Fig. 3 for 16 random samples from the rotational component of the Gaussian pose prior (i.e. global scale, rotation and translation is fixed).

2.2. Model Parameterization

We explicitly write out the parameters of our model as $\theta = (\mathtt{s}^T, \mathtt{t}^T, \mathtt{q}_0^T, ..., \mathtt{q}_L^T)^T \in \mathbb{R}^d$ with $\mathtt{s} \in \mathbb{R}$ defining global scale, $\mathtt{t} \in \mathbb{R}^3$ global translation and each $\mathtt{q}_l \in \mathbb{R}^4$ a quaternion defining the relative rotation of limb l if $l \in \{1, ..., L\}$ or global rotation if $l = 0$. To avoid overloading

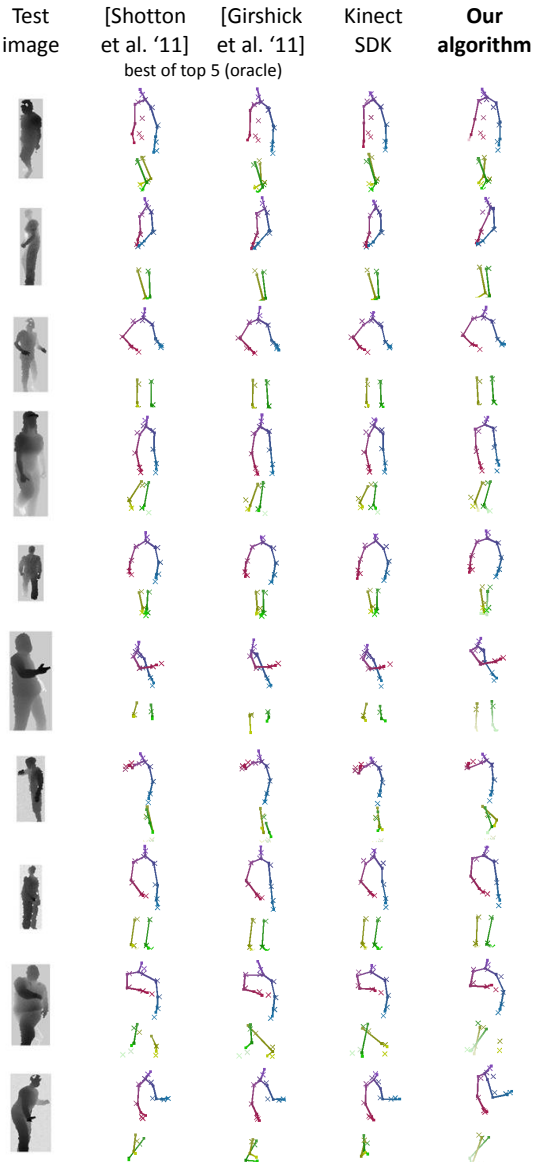

Figure 1. **Qualitative comparison with [2, 1] and the Kinect SDK.** All algorithms produce reasonable results on this random selection of test images. The quantitative results in the main paper, however, demonstrate the marked quantitative improvement in accuracy obtained by our algorithm. (Hips not drawn as [2, 1] do not predict hips, and the Kinect SDK hips are incompatible with our hips. Hips were not part of the metrics for this comparison). Note that the results from [2, 1] use the 'best of top 5' oracle and are not a realizable algorithm.

Parameter	Description	Value	Explanation
λ_{vis}	energy term weighting	$\frac{1}{n}$	n is the number of image pixels
λ_{prior}	energy term weighting	2×10^{-4}	
λ_{int}	energy term weighting	$\frac{1}{\lvert\mathcal{S}\rvert}$	
β	visibiliy sharpness	100	
γ	intersection sharpness	1000	
r_s	sphere radii	0.025m	
m	number of mesh vertices	~7000	
η	Geman-McClure parameter	0.1m / 10 pixels	meters → depth; pixels → silhouettes
τ	back-facing penalty	1.0	penalty paid
	number of trees	3	
	maximum depth of trees	20	
b	mean shift bandwidth	5 vits	used to cluster correspondences during tree training

Figure 2. **Settings for parameters introduced in the main paper and the derivations below.**

notation from the main paper, we use a fixed width font to denote these parameters and the transformations they specify. Explicitly, these specify a scaling transformation for $\mathtt{s}$

$$\mathtt{S}(\mathtt{s}) = \begin{pmatrix} \mathtt{s} & 0 & 0 & 0 \\ 0 & \mathtt{s} & 0 & 0 \\ 0 & 0 & \mathtt{s} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{1}$$

a translation transformation for $\mathtt{t} = (\mathtt{t}_1, \mathtt{t}_2, \mathtt{t}_3)$

$$\mathtt{T}(\mathtt{t}) = \begin{pmatrix} 1 & 0 & 0 & \mathtt{t}_1 \\ 0 & 1 & 0 & \mathtt{t}_2 \\ 0 & 0 & 1 & \mathtt{t}_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2}$$

and for each vector $\mathtt{q}_l$, a rotation $\mathtt{R}(\vartheta(\mathtt{q}_l))$ where $\vartheta(\mathtt{q}) = \frac{\mathtt{q}}{||\mathtt{q}||}$ normalizes $\mathtt{q}$ and for a unit quaternion $\mathtt{p} = (\mathtt{x}, \mathtt{y}, \mathtt{z}, \mathtt{w})$

$$\mathtt{R}(\mathtt{p}) = \begin{pmatrix} 1-2(\mathtt{y}^2+\mathtt{z}^2) & 2(\mathtt{xy}-\mathtt{zw}) & 2(\mathtt{xz}+\mathtt{yw}) & 0 \\ 2(\mathtt{xy}+\mathtt{zw}) & 1-2(\mathtt{x}^2+\mathtt{z}^2) & 2(\mathtt{yz}-\mathtt{xw}) & 0 \\ 2(\mathtt{xz}-\mathtt{yw}) & 2(\mathtt{yz}+\mathtt{xw}) & 1-2(\mathtt{x}^2+\mathtt{y}^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3}$$

We can explicitly define the remaining quantities in equation (1) and (2) of the main paper by

$$R_{glob}(\theta) = \mathtt{T}(\mathtt{t})\mathtt{R}(\vartheta(\mathtt{q}_0))\mathtt{S}(\mathtt{s}) \tag{4}$$

$$R_l(\theta) = \mathtt{T}(\mathtt{t}_l)\mathtt{R}(\vartheta(\mathtt{q}_l)) \tag{5}$$

where $\mathtt{t}_l$ for $l \in \{1, ..., L\}$ are fixed. It is readily seen that all transformations preserve unity in the 4th homogenous coordinate, a property assumed in each vertex position p_i, and thus we can assume that $\pi = P = I_{3\times 4}$, a matrix that drops the fourth coordinate. Further, dropping the distinction between model vertices v_j and predicted correspondences u_i, we have

$$M(u_i;\theta) = P\sum_{k=1}^{K} \alpha_{ik} T_{l_{ik}}(\theta) T^{-1}_{l_{ik}}(\theta_0) p_i \tag{6}$$

For each, vertex u_i let $n(u_i)$ denote it's surface normal in the base pose θ_0, perhaps computed using the local mesh geometry. Then an appropriate way to define the surface normal in a new pose θ is

$$N(u_i;\theta) = \frac{\hat{N}(u_i;\theta)}{||\hat{N}(u_i;\theta)||_2} \tag{7}$$

$$\hat{N}(u_i;\theta) = P\sum_{k=1}^{K} \alpha_{ik} \hat{T}_{l_{ik}}(\theta) \hat{T}^{-1}_{l_{ik}}(\theta_0) n(u_i)\,, \tag{8}$$

where $\hat{T}_l$ is composed only of the rotational components of T_l. That is, $n(u_i)$ is simply rotated by each limb's transformation, and the resulting combination normalized.

2.3. Gradient

The L-BFGS optimizer requires the gradient of the energy function to be provided. This could be computed using finite differences, but this would be inexact and needlessly slow, especially considering that the Jacobian of vertex coordinates with respect to the optimization parameters is particularly sparse. Therefore it is of benefit to compute the derivatives analytically so that they can be calculated directly. Below, we provide a recipe for doing this with a simplified energy function consisting of just E'_{vis} with pixel weights and β set to unity:

$$E'_{\text{vis}} = \sum_{i=1}^{n} V_i(\theta) \cdot \rho(e_i(\theta)) + (1 - V_i(\theta)) \cdot \tau \quad (9)$$

$$= n\tau + \sum_{i=1}^{n} V_i(\theta) \cdot [\rho(e_i(\theta)) - \tau] \ , \quad (10)$$

where $e_i(\theta) = ||x_i - M(u_i;\theta)||_2$, Geman-McClure function $\rho(e) = \frac{e^2}{e^2+\eta^2}$, and visibility weighting $V_i(\theta) = \sigma(-N(u_i;\theta)^\top A)$ where $\sigma(x)$ is the sigmoid function. Taking the derivative with respect to the j^{th} component[1] of the parameter vector θ we get:

$$\frac{\partial E(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \frac{\partial \rho(e_i(\theta))}{\partial \theta_j} V_i(\theta) + \frac{\partial V_i(\theta)}{\partial \theta_j} \cdot [\rho(e_i(\theta)) - \tau] \ . \quad (11)$$

Working on the first term, observe that

$$\frac{\partial \rho(e_i(\theta))}{\partial \theta_j} = \frac{2e_i(\theta)\eta^2}{(e_i(\theta)^2 + \eta^2)^2} \frac{\partial e_i(\theta)}{\partial \theta_j} \ . \quad (12)$$

and

$$\frac{\partial e_i(\theta)}{\partial \theta_j} = \frac{1}{e_i(\theta)} (M(u_i;\theta) - x_i)^\top \frac{\partial M(u_i;\theta)}{\partial \theta_j} \ . \quad (13)$$

Further:

$$\frac{\partial M(u_i;\theta)}{\partial \theta_j} = P \sum_{k=1}^{K} \alpha_{ik} \frac{\partial T_{l_{ik}}(\theta)}{\partial \theta_j} T_{l_{ik}}^{-1}(\theta_0) p_i \ , \quad (14)$$

Note that $T_{l_{ik}}(\theta)$ is composed only of matrices of the form $\mathtt{T(t)}$, $\mathtt{R(q)}$ and $\mathtt{S(s)}$, of which only one will depend on θ_j. Thus, it is enough to be able to compute derivatives for these individually. Specifically:

$$\frac{\partial \mathtt{T(t)}}{\partial \mathtt{t}_h} = \begin{pmatrix} 0 & 0 & 0 & \delta_{h1} \\ 0 & 0 & 0 & \delta_{h2} \\ 0 & 0 & 0 & \delta_{h3} \\ 0 & 0 & 0 & 0 \end{pmatrix} \ , \quad (15)$$

(using the Kronecker δ), and

$$\frac{\partial \mathtt{S(s)}}{\partial \mathtt{s}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \ . \quad (16)$$

For a rotation parameter $\mathtt{q}$, we write the corresponding unit quaternion as $\mathtt{p} = \vartheta(\mathtt{q}) = (\mathtt{x}, \mathtt{y}, \mathtt{z}, \mathtt{w})$. Using the chain rule, we obtain the total derivative with respect to each component $\mathtt{q}_j$ of $\mathtt{q}$:

$$\frac{\partial \mathtt{R}(\vartheta(\mathtt{q}))}{\partial \mathtt{q}_j} = \sum_{m=1}^{4} \frac{\partial \mathtt{R(p)}}{\partial \mathtt{p}_m} \frac{\partial \vartheta_m(\mathtt{q})}{\partial \mathtt{q}_j} \quad (17)$$

[1]Here we use j to indicate a component of parameter vector θ. In the main paper j refers to a model vertex index.

where the derivative of the normalization terms are given by

$$\frac{\partial \vartheta_m(\mathtt{q})}{\partial \mathtt{q}_j} = \frac{\delta_{mj}}{||\mathtt{q}||_2} - \frac{\mathtt{q}_m \mathtt{q}_j}{||\mathtt{q}||_2^3} \ . \quad (18)$$

and the other terms are easily calculated from equation Eq. 3. For example,

$$\frac{\partial \mathtt{R(p)}}{\partial \mathtt{x}} = \begin{pmatrix} 0 & 2\mathtt{y} & 2\mathtt{z} & 0 \\ 2\mathtt{y} & -4\mathtt{x} & -2\mathtt{w} & 0 \\ 2\mathtt{z} & 2\mathtt{w} & -4\mathtt{x} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \ . \quad (19)$$

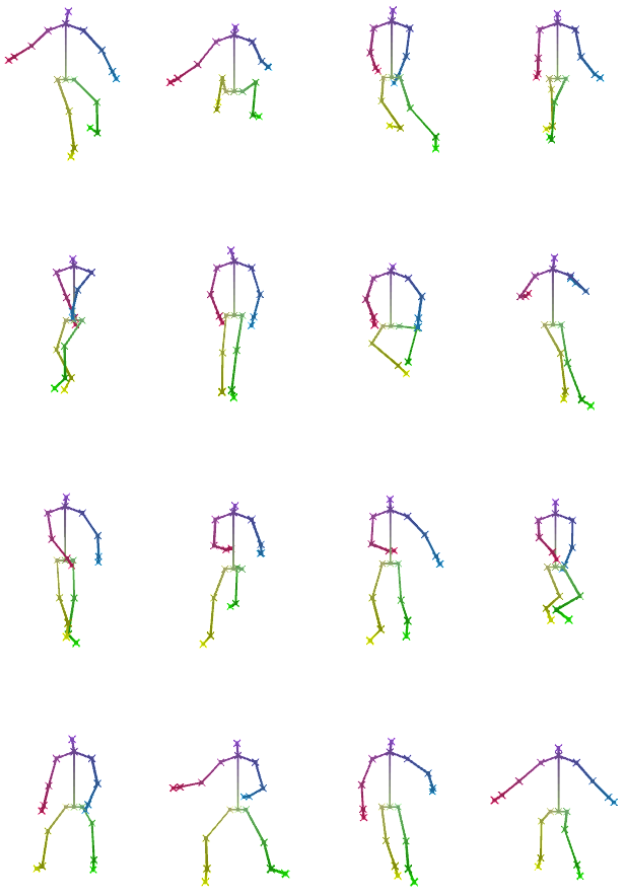

Figure 3. **Samples from our Gaussian pose prior.**

Turning to the second term on the right hand side of Eq. 11 we see

$$\frac{\partial V_i(\theta)}{\partial \theta_j} = -\sigma(-N_3(u_i;\theta))(1-\sigma(-N_3(u_i;\theta)) \frac{\partial N_3(u_i;\theta)}{\partial \theta_j} \quad (20)$$

where subscript 3s denote the Z component of the normal vectors, and $\frac{\partial N_3(u_i;\theta)}{\partial \theta_j}$ can be calculated easily from $\frac{\partial \hat{N}(u_i;\theta)}{\partial \theta_j}$ as it is simply a normalization (similar to $\frac{\partial \vartheta(\mathtt{q})}{\partial \mathtt{q}_j}$

above) and the calculation of $\frac{\partial \hat{N}(u_i;\theta)}{\partial \theta_j}$ is similar to that of $\frac{\partial M(u_i;\theta)}{\partial \theta_j}$ above.

The remaining gradients of the full energy in the paper can be derived analytically in a similar manner using standard calculus.

References

[1] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *Proc. ICCV*, 2011.

[2] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from a single depth image. In *Proc. CVPR*, 2011.