How to Refine 3D Hand Pose Estimation from Unlabelled Depth Data ? Supplementary Material

Endri Dibra[†], Thomas Wolf[†], Cengiz Öztireli, Markus Gross
Department of Computer Science, ETH Zürich
[†]Equal contribution from both authors
{edibra,cengizo,grossm}@inf.ethz.ch, wolftho@student.ethz.ch

1. Architecture

We illustrate in Fig. 1 our candidate base model in detail.

2. Videos

We attach a video, consisting of three main parts, which show some qualitative results on the optimization procedure explained in the paper, as well as additional qualitative results obtained from a real subject.
In the first two parts, **Pose Optimization** and **Shape Optimization**, we adapt the base model and hand shape to a single image until convergence. This serves to visualize the Sec 4.3 from the paper and gives insight on how the pose predictions get updated during learning.
In the next part, **Comparisons**, we show two videos, comparing the base model to the adapted model trained on our dataset (with 50K unlabelled images). This refers to Sec 4.4 in the paper. The first video shows a validation sequence of 300 frames with various hand poses. The second video shows a live prediction using the Intel RealSense SR300 camera. Note that not only the pose prediction quality is enhanced, but also jitter is removed significantly.

3. Implementation

We implemented linear blend skinning, the depth renderer and the collision component as custom operations in Tensorflow using CUDA kernels. To implement a custom operation $f : A \mapsto B$ in Tensorflow, the explicit gradient of $g(f)$ has to be given, where $g : B \mapsto \mathbb{R}$ is an arbitrary loss functon.

3.1. Linear blend skinning

Let m be the number of joints. Let $M = [M_1, \dots, M_m]$, where $M_i \in \mathbb{R}^{4\times4}$ is the matrix, that transforms from joint space of joint i to view space for the hand pose θ (calculated with the forwared kinematic chain). Let $P = [p_1, \dots, p_n]$ with $p_i \in \mathbb{R}^4$ all point positions in homogeneous coordinates and $w_{i,j} \in W \in \mathbb{R}^{n\times m}$ the weight, that defines how much point p_i is bound to the joint $j \in [m]$.

3.1.1 Derivative w.r.t. M

We write f^{skin} in terms of a scalar values with $c \in [4]$ as coordinate axis:

$$f^{\text{skin}}_{i,c}(P, M) = \sum_{d=1}^{4}\sum_{j=1}^{m} w_{i,j} M_{j,c,d} p_{i,d} \tag{1}$$

The derivative of $f := f^{\text{skin}}$ is:

$$\frac{\partial f_{i,c}(P,M)}{\partial M_{k,l,m}} = \mathbb{1}_{\{l=c\}} w_{i,k} p_{i,m} \tag{2}$$

The loss function $g(f)$ can be differentiated using the chain rule:

$$\frac{\partial}{\partial M_{k,l,m}} g(f(M,P)) = \sum_{i=1}^{n}\sum_{c=1}^{4} \frac{\partial g(f)}{\partial f_{i,c}} \frac{\partial f_{i,c}(P,M)}{\partial M_{k,l,m}} \tag{3}$$

$$= \sum_{i=1}^{n}\sum_{c=1}^{4} \frac{\partial g(f)}{\partial f_{i,c}} \mathbb{1}_{\{l=c\}} w_{i,k} p_{i,m} \tag{4}$$

$$= \sum_{i=1}^{n} \frac{\partial g(f)}{\partial f_{i,l}} w_{i,k} p_{i,m} \tag{5}$$

3.1.2 Derivative w.r.t P

To update P, we also need to differentiate f^{skin} w.r.t. P. In the equations above, we left out the batch dimension of f^{skin}, since all the operations can be done in parallel and independently for each batch element. However, our points P are the same for each batch element, s.t. we need to be more precise here. Let $f^{\text{skin}} \in \mathbb{R}^{n\times4} \times \mathbb{R}^{N\times n\times4} \mapsto \mathbb{R}^{N\times n\times4\times4}$ where n is the number of points and N the size of the batch. Note that also M exists for each batch element and therefore has now four dimensions. For a batch index b, point

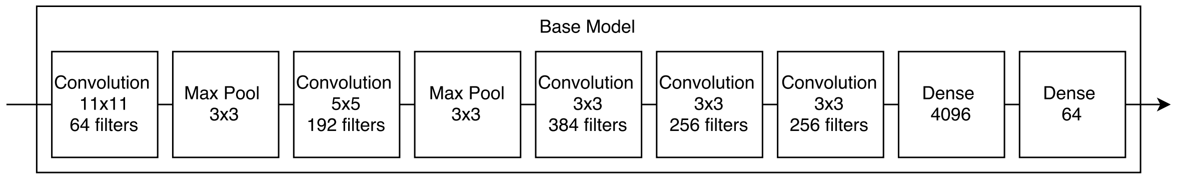
Base Model
Convolution 11x11 64 filters
Max Pool 3x3
Convolution 5x5 192 filters
Max Pool 3x3
Convolution 3x3 384 filters
Convolution 3x3 256 filters
Convolution 3x3 256 filters
Dense 4096
Dense 64

Figure 1. CNN architecture of the base model

index i and coordinate index c we rewrite Eq. 1 as:

$$f^{\text{skin}}_{b,i,c}(P, M) = \sum_{d=1}^{4}\sum_{j=1}^{m} w_{i,j} M_{b,j,c,d} p_{i,d} \tag{6}$$

Taking the derivative of $f := f^{\text{skin}}$ gives:

$$\frac{\partial f_{b,i,c}(P, M)}{\partial p_{k,l}} = \mathbb{1}_{\{k=i\}} \sum_{j=1}^{m} w_{i,j} M_{b,j,c,l} \tag{7}$$

And the derivative of $g(f)$ is given by:

$$\begin{aligned}\frac{\partial}{\partial p_{k,l}} g(f(M, P)) &= \sum_{b=1}^{N}\sum_{i=1}^{n}\sum_{c=1}^{4} \frac{\partial g(f)}{\partial f_{b,i,c}} \frac{\partial f_{b,i,c}(P, M)}{\partial p_{k,l}} \\ &= \sum_{b=1}^{N}\sum_{i=1}^{n}\sum_{c=1}^{4} \frac{\partial g(f)}{\partial f_{b,i,c}} \mathbb{1}_{\{k=i\}} \sum_{j=1}^{m} w_{i,j} M_{b,j,c,l} \\ &= \sum_{b=1}^{N}\sum_{c=1}^{4} \frac{\partial g(f)}{\partial f_{b,k,c}} \sum_{j=1}^{m} w_{i,j} M_{b,j,c,l} \end{aligned} \tag{8}$$

3.2. Depth Renderer

Let $f := f^{\text{depth}}$. The derivative of a loss function $g(f)$ for $c \in \{x, y, z\}$ is given by:

$$\frac{\partial}{\partial p_{l,c}} g(f(P)) = \sum_{i=1}^{120}\sum_{j=1}^{120} \frac{\partial g(f_{i,j})}{\partial f_{i,j}} \frac{\partial f_{i,j}(P)}{\partial p_{l,c}} \tag{9}$$

with

$$\frac{\partial f_{i,j}(P)}{\partial p_{l,x}} = h(l, P)(j - p_{l,x}) \tag{10}$$

$$\frac{\partial f_{i,j}(P)}{\partial p_{l,y}} = h(l, P)(i - p_{l,y}) \tag{11}$$

$$\frac{\partial f_{i,j}(P)}{\partial p_{l,z}} = -\mathbb{1}_{\{l=\operatorname{argmax}_k(\text{depth}_{i,j}(p_k))\}} \phi(p_l) \tag{12}$$

where

$$\begin{aligned} h(l, P) = &\mathbb{1}_{\{l=\operatorname{argmax}_k(\text{depth}_{i,j}(p_k)) \wedge \phi(p_l)>0\}} \\ &\frac{4(1 - p_{l,z})}{r^2} \left(1 - \frac{(j - p_{l,x})^2 + (i - p_{l,y})^2}{r^2}\right) \end{aligned} \tag{13}$$

To implement the loops over the image from Eq. 9 efficiently, we can make use of the finite spatial support of ϕ and loop only over a range of $\{\lfloor p_{l,x} - r\rfloor, \dots, \lceil p_{l,y} + r\rceil\}$ for each point p_l, which is for some indices i, j with $1 \leq i, j \leq 120$ fulfilling $l = \operatorname{argmax}_k(\text{depth}_{i,j}(p_k))$.

3.3. Collision Component

Let $B := [b_1, \dots, b_m] \in \mathbb{R}^{m\times 3}$ the joint positions, $C \subset \mathbb{N} \times \mathbb{N}$ be all joint indices paired with their parent. Let $\mathcal{L} := \mathcal{L}^{\text{coll}}$ and $f := f^{\text{coll}}$. We define f by:

$$f(a, b, p) = \max(0, \min(f_1(a, b, p), f_2(a, b, p))) \tag{14}$$

with

$$f_1(a, b, p) = (0.5h)^2 - \text{dist}_h(a, b, p) \tag{15}$$

and

$$f_2(a, b, p) = r^2 - \text{dist}_r(a, b, p) \tag{16}$$

where $\text{dist}_h(a, b, p)$ and $\text{dist}_r(a, b, p)$ are the squared distances of a point p from $\mu = \frac{a+b}{2}$ in cylindrical coordinates of a cylinder with endpoints a and b and height h:

$$\text{dist}_h(a, b, p) = \left(\frac{2}{h}(b - \mu)^T(p - \mu)\right)^2 \tag{17}$$

$$\text{dist}_r(a, b, p) = \|p - \mu\|^2 - \text{dist}_h(a, b, p) \tag{18}$$

The derivative of $g(\mathcal{L})$ is given by:

$$\frac{\partial}{\partial b_l} g(\mathcal{L}(B)) = \frac{g'(\mathcal{L})}{m} \sum_{(i,j)\in C}\sum_{k=1}^{m} \mathbb{1}_{i\neq k, j\neq k} \frac{\partial f(b_i, b_j, b_k)}{\partial b_l} \tag{19}$$

with

$$\begin{aligned}\frac{\partial f(b_i, b_j, b_j)}{\partial b_l} = &-\mathbb{1}_{\{\min(f_1,f_2)>0\}} \\ &(\mathbb{1}_{\{f_1<f_2\}} \frac{\partial\, \text{dist}_h}{\partial b_l} + \mathbb{1}_{\{f_1>f_2\}} \frac{\partial\, \text{dist}_r}{\partial b_l}) \end{aligned} \tag{20}$$

and

$$\begin{aligned}\frac{\partial\, \text{dist}_h(b_i, b_j, b_k)}{\partial b_l} = \\ \frac{4}{h}\sqrt{\text{dist}_h} \cdot \begin{cases} b_i - b_k, & \text{if } l = i \\ b_k - b_j, & \text{if } l = j \\ b_j - b_i, & \text{if } l = k \end{cases}\end{aligned} \tag{21}$$

Finger Part	1st Rotation Axis	2nd Rotation Axis	3rd Rotation Axis
1. Index Finger Lower	[-1.4, 0.1]	[-0.1, 0.4]	[-0.1, 0.6]
2. Index Finger Middle	[0, 0]	[-0.1, 0.1]	[-0.1, 1.2]
3. Index Finger Upper	[0, 0]	[0, 0]	[-1.0, 1.0]
4. Middle Finger Lower	[0, 0]	[-0.8, 0.2]	[-0.1, 1.9]
5. Middle Finger Middle	[0, 0]	[0, 0]	[-0.7, 1.2]
6. Middle Finger Upper	[0, 0]	[0, 0]	[-0.3, 1.6]
7. Ring Finger Lower	[0, 0]	[-0.7, 0.2]	[-0.1, 1.8]
8. Ring Finger Middle	[0, 0]	[0, 0]	[-0.7, 1.5]
9. Ring Finger Upper	[0, 0]	[0, 0]	[-0.3, 1.6]
10. Little Finger Lower	[0, 0]	[-0.5, 0.2]	[-0.1, 1.8]
11. Little Finger Middle	[0, 0]	[0, 0]	[-0.5,1.5]]
12. Little Finger Upper	[0, 0]	[0, 0]	[-0.3, 1.6]
13. Thumb Lower	[0, 0]	[-0.4, 0.4]	[-0.2, 1.8]
14. Thumb Middle	[0, 0]	[0, 0]	[-0.5, 1.5]
15. Thumb Upper	[0, 0]	[0, 0]	[-0.3, 1.6]

Table 1. Allowed euler angle bounds $[\underline{\varphi}, \bar{\varphi}]$ for each joint or finger part. Values are in radians.

$$\frac{\partial \operatorname{dist}_r(b_i, b_j, b_k)}{\partial b_l} = \frac{\partial \operatorname{dist}_h}{\partial b_l} + \begin{cases} 2b_k - b_i - b_j, & \text{if } l = i \\ 0.5(b_i + b_j) - p_k, & \text{if } l = j \\ 0.5(b_i + b_j) - p_k, & \text{if } l = k \end{cases} \tag{22}$$

Please note that we left out the arguments (b_i, b_j, b_j) for clarity in Eq. 20, 21 and 22.

3.4. Physical Component

In Tab.1 we give some extra detail on the bounds for the valid range $[\underline{\varphi}, \bar{\varphi}]$ of the euler angles for each finger part/joint of the rigged 3D hand model depicted in Fig.2 from the paper and downloaded online from Turbosquid [1]. An angle of zero represents the joint/finger in its neutral pose, which could be defined aribitrarily, but in our case is the open palm. There are three rotation axes. The first one has the same axis as the one from the finger. The second axis rotates the finger to the side (e.g. to account for the separation between the fingers). The third axis rotates the finger towards the palm. We first rotate around the first axis, then around the third axis and lastly around the second one.

[1]https://www.turbosquid.com/3d-models/rigged-male-hand-max/786338