Procrustean Regression Networks: Learning 3D Structure of Non-Rigid Objects from 2D Annotations Supplementary Materials

Sungheon Park[1], Minsik Lee[2], and Nojun Kwak[3]

[1] Samsung Advanced Institute of Technology (SAIT), Korea
sungheonpark@snu.ac.kr
[2] Hanyang University, Korea mleepaper@hanyang.ac.kr
[3] Seoul National University, Korea nojunk@snu.ac.kr

1 Derivation of $\frac{\partial \widetilde{\mathbf{X}}}{\partial \mathbf{X}}$ in the Cost Function of PRN

To make this material self-contained, we include the derivation of the back-propagation process of the proposed PRN, which is similar to that appeared in [4]. We start by rewriting the cost function of PRN:

$$\mathcal{J} = \sum_{i=1}^{n_f} f(\mathbf{X}_i) + \lambda g(\widetilde{\mathbf{X}}), \tag{1}$$

while satisfying the alignment constraint

$$\mathbf{R} = \underset{\mathbf{R}}{\operatorname{argmin}} \sum_{i=1}^{n_f} \|\mathbf{R}_i\mathbf{X}_i\mathbf{T} - \frac{1}{n_f}\sum_{j=1}^{n_f}\mathbf{R}_j\mathbf{X}_j\mathbf{T}\| \qquad \text{s.t.} \quad \mathbf{R}_i^T\mathbf{R}_i = \mathbf{I}_3. \tag{2}$$

To integrate the alignment constraint (2) with the cost function (1), we introduce an orthogonal matrix $\mathbf{Q}_i$ that satisfies $\mathbf{R}_i = \mathbf{Q}_i\hat{\mathbf{R}}_i$ and assume $\mathbf{Q}_i = \mathbf{I}_3$ at the time of gradient evaluation without loss of generality. Then, $\widetilde{\mathbf{X}}_i = \mathbf{Q}_i\hat{\mathbf{R}}_i\mathbf{X}_i\mathbf{T} = \mathbf{Q}_i\mathbf{X}'_i$. Integrating the orthogonality constraint $\mathbf{Q}_i^T\mathbf{Q}_i = \mathbf{I}_3$ to (2) by introducing Lagrange multipliers $\mathbf{\Lambda}_i$ yields the following equation:

$$\sum_{i=1}^{n_f}\|\mathbf{Q}_i\mathbf{X}'_i - \frac{1}{n_f}\sum_{j=1}^{n_f}\mathbf{Q}_j\mathbf{X}'_j\|^2 + \frac{1}{2}\sum_{i=1}^{n_f}\left\langle \mathbf{\Lambda}_i, \mathbf{Q}_i^T\mathbf{Q}_i - \mathbf{I}_3\right\rangle. \tag{3}$$

Differentiating (3) with respect to $\mathbf{Q}_k (1 \le k \le n_f)$ yields

$$(\mathbf{Q}_k\mathbf{X}'_k - \frac{1}{n_f}\sum_{j=1}^{n_f}\mathbf{Q}_j\mathbf{X}'_j)(\mathbf{X}'^T_k - \frac{1}{n_f}\mathbf{X}'^T_k) + \sum_{i\neq k}(\mathbf{Q}_i\mathbf{X}'_i - \frac{1}{n_f}\sum_{j=1}^{n_f}\mathbf{Q}_j\mathbf{X}'_j)(-\frac{1}{n_f}\mathbf{X}'^T_k) + \mathbf{Q}_k\mathbf{\Lambda}_k = \mathbf{0}. \tag{4}$$

By rearranging (4) and multiplying $\mathbf{Q}_k^T$ on both sides, we get the following equation,

$$\mathbf{Q}_k(\frac{n_f-1}{n_f}\mathbf{X}'_k\mathbf{X}_k^{'T}+\mathbf{\Lambda}_k)\mathbf{Q}_k^T=\frac{1}{n_f}\sum_{i\neq k}\mathbf{Q}_i\mathbf{X}'_i\mathbf{X}_k^{'T}\mathbf{Q}_k^T. \tag{5}$$

On the left hand side, $\mathbf{\Lambda}_k$ is a symmetric matrix since the orthogonality constraint is symmetric (*i.e.*, $\mathbf{Q}_k^T\mathbf{Q}_k=\mathbf{Q}_k\mathbf{Q}_k^T=\mathbf{I}_3$). Hence, the left hand side is symmetric, and so is the right hand side, *i.e.*,

$$\sum_{i\neq k}\mathbf{Q}_i\mathbf{X}'_i\mathbf{X}_k^{'T}\mathbf{Q}_k^T=\sum_{i\neq k}\mathbf{Q}_k\mathbf{X}'_k\mathbf{X}_i^{'T}\mathbf{Q}_i^T. \tag{6}$$

By vectorizing (6), we get

$$\begin{aligned}&\mathrm{vec}(\sum_{i\neq k}\mathbf{Q}_k\mathbf{X}'_k\mathbf{X}_i^{'T}\mathbf{Q}_i^T)-\mathrm{vec}(\sum_{i\neq k}\mathbf{Q}_i\mathbf{X}'_i\mathbf{X}_k^{'T}\mathbf{Q}_k^T)\\ =&[(\sum_{i\neq k}\mathbf{Q}_i\mathbf{X}'_i\otimes\mathbf{I}_3)-(\mathbf{I}_3\otimes\sum_{i\neq k}\mathbf{Q}_i\mathbf{X}'_i)\mathbf{E}]\mathrm{vec}(\mathbf{Q}_k\mathbf{X}'_k)=\mathbf{0},\end{aligned} \tag{7}$$

where $\mathbf{E}$ is a permutation matrix that satisfies $\mathbf{E}\mathrm{vec}(\mathbf{H})=\mathrm{vec}(\mathbf{H}^T)$. On the other hand, differentiating $\mathbf{Q}_i^T\mathbf{Q}_i=\mathbf{I}_3$ and evaluating at $\mathbf{Q}_i=\mathbf{I}_3$ gives

$$\partial\mathbf{Q}_i^T\mathbf{Q}_i+\mathbf{Q}_i^T\partial\mathbf{Q}_i=\partial\mathbf{Q}_i^T+\partial\mathbf{Q}_i=0. \tag{8}$$

The above equation is a well-known relation about the derivative of an orthogonal matrix [2]. Here, $\partial\mathbf{Q}_i$ is interpreted as an infinitesimal generator of rotation, which is a skew-symmetric matrix. Let us denote $\partial\mathbf{Q}_i$ as

$$\partial\mathbf{Q}_i=\begin{bmatrix}0 & \partial q_{iz} & -\partial q_{iy}\\ -\partial q_{iz} & 0 & \partial q_{ix}\\ \partial q_{iy} & -\partial q_{ix} & 0\end{bmatrix}, \tag{9}$$

and if we define $\partial\mathbf{q}_i=[\partial q_{ix}\quad\partial q_{iy}\quad\partial q_{iz}]^T$, then $\mathrm{vec}(\partial\mathbf{Q}_i)=\mathbf{L}\partial\mathbf{q}_i$.

Now, given an arbitrary $3\times n_p$ matrix $\mathbf{S}$ and its column vectors $\mathbf{s}_1,\mathbf{s}_2,\cdots,\mathbf{s}_{n_p}$, one can easily verify that $(\mathbf{S}^T\otimes\mathbf{I}_3)\mathbf{L}=\begin{bmatrix}[\mathbf{s}_1]_\times^T & [\mathbf{s}_2]_\times^T\cdots[\mathbf{s}_{n_p}]_\times^T\end{bmatrix}^T$ where $[\mathbf{s}]_\times$ is a skew-symmetric matrix that is related to the cross product of the vector. We can also obtain $[(\mathbf{S}^T\otimes\mathbf{I}_3)\mathbf{L}]^T$ from $(\mathbf{S}\otimes\mathbf{I}_3)-(\mathbf{I}_3\otimes\mathbf{S})\mathbf{E}$ by selecting 8th, 3rd, and 4th rows. The rest of the rows are either essentially identical to these rows or trivial. As a consequence, from (7), we get

$$\mathbf{L}^T(\sum_{i\neq k}\mathbf{Q}_i\mathbf{X}'_i\otimes\mathbf{I}_3)\mathrm{vec}(\mathbf{Q}_k\mathbf{X}'_k)=\mathbf{L}^T\mathrm{vec}(\mathbf{Q}_k\mathbf{X}'_k\sum_{i\neq k}\mathbf{X}_i^{'T}\mathbf{Q}_i^{'T})=\mathbf{0}. \tag{10}$$

Differentiating (10) yields

$$\begin{aligned}\mathbf{L}^T\mathrm{vec}(\partial\mathbf{Q}_k\mathbf{X}'_k\sum_{i\neq k}\mathbf{X}_i^{'T}\mathbf{Q}_i^{'T}+\partial\mathbf{X}'_k\sum_{i\neq k}\mathbf{X}_i^{'T}+\mathbf{X}'_k\sum_{i\neq k}\partial\mathbf{X}_i^{'T}\\ +\mathbf{X}'_k\sum_{i\neq k}\mathbf{X}_i^{'T}\partial\mathbf{Q}_i^{'T})=\mathbf{0}.\end{aligned} \tag{11}$$

By rearranging (11) and substituting $\text{vec}(\partial \mathbf{Q}_i) = \mathbf{L}\partial \mathbf{q}_i$ yields

$$\begin{aligned}&\mathbf{L}^T(\sum_{i\neq k}\mathbf{X}'_i\mathbf{X}_k'^T \otimes \mathbf{I}_3)\mathbf{L}\partial\mathbf{q}_k + \mathbf{L}^T\sum_{i\neq k}(\mathbf{I}_3 \otimes \mathbf{X}'_k\mathbf{X}_i'^T)\mathbf{E}\mathbf{L}\partial\mathbf{q}_i \\ &= -\mathbf{L}^T(\sum_{i\neq k}\mathbf{X}'_i \otimes \mathbf{I}_3)\text{vec}(\partial\mathbf{X}'_k) - \mathbf{L}^T\sum_{i\neq k}(\mathbf{I}_3 \otimes \mathbf{X}'_k)\mathbf{E}\text{vec}(\partial\mathbf{X}'_i).\end{aligned} \tag{12}$$

Since the index k varies from 1 to n_f, n_f equations are made from (12). Let $\partial\mathbf{q}$ be a vector $\partial\mathbf{q} = [\partial\mathbf{q}_1^T, \partial\mathbf{q}_2^T, \cdots, \partial\mathbf{q}_{n_f}^T]^T$, and similarly we define $\text{vec}(\partial\mathbf{X}') = [\text{vec}(\partial\mathbf{X}'_1)^T, \text{vec}(\partial\mathbf{X}'_2)^T, \cdots, \text{vec}(\partial\mathbf{X}'_{n_f})^T]^T$. To formulate $\partial\mathbf{q}$ as a function of $\text{vec}(\partial\mathbf{X}')$, we enumerate n_f equations and build a linear system that has the form of

$$\mathbf{B}\partial\mathbf{q} = \mathbf{C}\text{vec}(\partial\mathbf{X}'). \tag{13}$$

where $\mathbf{B}$ and $\mathbf{C}$ are the matrices explained in the main text. Then, $\partial\mathbf{q}$ is expressed as

$$\partial\mathbf{q} = \mathbf{B}^{-1}\mathbf{C}\text{vec}(\partial\mathbf{X}'). \tag{14}$$

Next, differentiating $\mathbf{Q}_i\mathbf{X}'_i = \widetilde{\mathbf{X}}_i$ yields

$$\partial\mathbf{Q}_i\mathbf{X}'_i + \mathbf{Q}_i\partial\mathbf{X}'_i = \partial\widetilde{\mathbf{X}}_i. \tag{15}$$

By vectorizing (15), we get

$$(\mathbf{X}_i'^T \otimes \mathbf{I})\mathbf{L}\partial\mathbf{q}_i = \text{vec}(\partial\widetilde{\mathbf{X}}_i - \partial\mathbf{X}'_i). \tag{16}$$

Let $\text{vec}(\partial\widetilde{\mathbf{X}}) = [\text{vec}(\partial\widetilde{\mathbf{X}}_1)^T, \text{vec}(\partial\widetilde{\mathbf{X}}_2)^T, \cdots, \text{vec}(\partial\widetilde{\mathbf{X}})_{n_f}^T]^T$, and similar to (13), we build a linear system by varying the index i from 1 to n_f,

$$\mathbf{A}\partial\mathbf{q} = \text{vec}(\partial\widetilde{\mathbf{X}}) - \text{vec}(\partial\mathbf{X}'), \tag{17}$$

where $\mathbf{A}$ is also explained in the main text.

Substituting (14) to (17) yields

$$(\mathbf{A}\mathbf{B}^{-1}\mathbf{C} + \mathbf{I}_{3n_pn_f})\text{vec}(\partial\mathbf{X}') = \text{vec}(\partial\widetilde{\mathbf{X}}). \tag{18}$$

Finally, dividing both sides of (18) by $\partial\text{vec}(\mathbf{X})$ gives the derivative we need,

$$\frac{\text{vec}(\partial\widetilde{\mathbf{X}})}{\partial\text{vec}(\mathbf{X})} = (\mathbf{A}\mathbf{B}^{-1}\mathbf{C} + \mathbf{I}_{3n_pn_f})\mathbf{D}, \tag{19}$$

where $\mathbf{D}$ is a block-diagonal matrix explained in the main text. Note that $\mathbf{D}$ is based on the following derivative.

$$\frac{\text{vec}(\partial\mathbf{X}'_i)}{\partial\text{vec}(\mathbf{X}_i)} = \frac{(\mathbf{T} \otimes \hat{\mathbf{R}}_i)\text{vec}(\partial\mathbf{X}_i)}{\partial\text{vec}(\mathbf{X}_i)} = \mathbf{T} \otimes \hat{\mathbf{R}}_i. \tag{20}$$

The derivative of the cost function is calculated as

$$\frac{\partial\mathcal{J}}{\partial\text{vec}(\mathbf{X})} = \frac{\partial f}{\partial\text{vec}(\mathbf{X})} + \lambda\left\langle\frac{\partial g}{\partial\text{vec}(\widetilde{\mathbf{X}})}, \frac{\text{vec}(\partial\widetilde{\mathbf{X}})}{\partial\text{vec}(\mathbf{X})}\right\rangle, \tag{21}$$

where the dimensions of $\frac{\partial\mathcal{J}}{\partial\text{vec}(\mathbf{X})}$, $\frac{\partial f}{\partial\text{vec}(\mathbf{X})}$, and $\frac{\partial g}{\partial\text{vec}(\widetilde{\mathbf{X}})}$ are $1 \times 3n_pn_f$, and the dimension of $\frac{\text{vec}(\partial\widetilde{\mathbf{X}})}{\partial\text{vec}(\mathbf{X})}$ is $3n_pn_f \times 3n_pn_f$.

Table 1. MPJPE with missing 2D inputs on Human 3.6M dataset.

Missing Points Ratio (%)	0	5	10	15	20
PRN-FCN	66.7	69.8	70.8	72.1	73.8

2 Training Details

For the Human 3.6M dataset, PRN-FCN with 2D ground truth inputs receives 17 keypoints for each frame while that with stacked hourglass network detection inputs receive 16 keypoints. Both networks produce 17 keypoints as an output which is trained with ground truth 2D joint positions. Iterations for PRN-FCN and PRN-CNN are 70,000 and 120,000, respectively. For CNN architectures, 3D shapes of human bodies are directly estimated from RGB images. Image frames in the dataset are cropped using ground truth bounding box information so that a person is centered in the cropped image, which are then resized to 256×256.

For the 300-VW dataset, both 2D inputs and 3D outputs have 68 keypoints. Since the dataset is smaller than Human 3.6M, iterations for FCN and CNN are set to 14,000 and 100,000, respectively. The RGB images are also cropped so that the mean position of the 2D landmarks becomes the center of the image.

We used Adam optimizer [3] with the start learning rate of 10^{-4}. The learning rate is decayed by 0.8 for every 5,000 iterations. Number of hidden nodes in fully-connected ResBlock is set to 1,024 for Human 3.6M and 300-VW datasets and to 4,096 for SURREAL dataset respectively.

3 Additional Experimental Results

3.1 Robustness to missing points on Human 3.6M dataset

We measured the robustness of PRN-FCN when there exist missing points in both training and test datasets. We increased the ratio of missing points from 0% to 20% with 5% intervals and trained PRN for each case on Human 3.6M dataset. The missing points are randomly selected and 2D inputs of missing points are set to 0. The MPJPE of PRN-FCN for all cases are shown in Table 1. It is verified that PRN is robust to missing points since the error only slightly increases as the ratio of missing point gets larger.

3.2 Qualitative results of PRN-CNN on Human 3.6M dataset

Qualitative results of PRN-CNN are provided in Figure 1 with input images and ground truths. PRN-CNN is able to estimate accurate 3D poses for various images including the case when a part of body joints are self-occluded. The last column of Figure 1 shows a common failure case. PRN outputs poor estimation results when the input images are side-views of human bodies and the ground truth 3D poses have large depth variations. For those cases, a large depth change may only lead to small changes in RGB images, so PRN suffers from distinguishing those subtle changes.

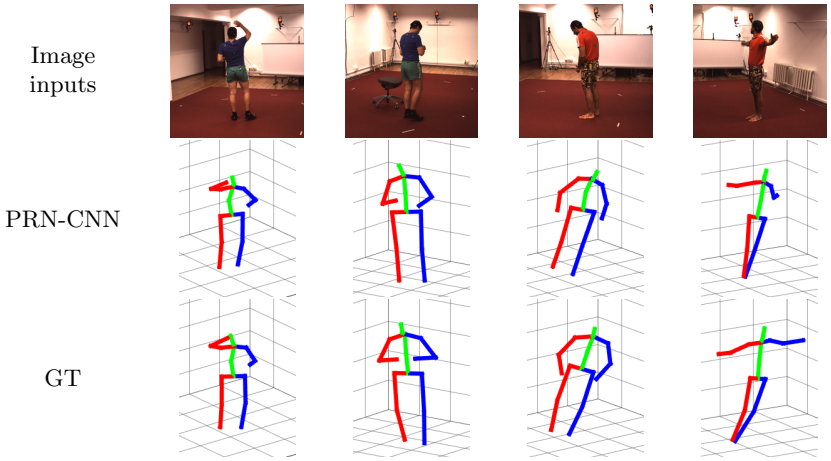

Fig. 1. Qualitative results of PRN-CNN with RGB image inputs on Human 3.6M dataset. The last row shows a failure case.

Table 2. Normalized error with RGB inputs on the 300-VW dataset.

Method	Normalized Error
CSF2 [1] + CNN	0.4331
PR [4] + CNN	0.4224
PRN-CNN	**0.3092**

3.3 Performance on 300-VW dataset with RGB inputs

We also provided the quantitative results on 300-VW dataset trained with RGB image inputs. As illustrated in Table 2, PRN-CNN performs better than the compared CNN models which are trained using NRSfM results.

We have also shown a few reconstruction results of PRN-CNN on various test images in Figure 2. The results from PRN are illustrated in views from the XY-plane and the YZ-plane. From the XY-plane viewpoints, it is verified that the 2D poses of faces are successfully learned in PRN, although it is hard to capture subtle variations of eyes or mouth. We can also verify from the YZ-plane viewpoints that the depth information is correctly estimated.

References

1. Gotardo, P.F., Martinez, A.M.: Non-rigid structure from motion with complementary rank-3 spaces. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 3065–3072. IEEE (2011)
2. Hall, B.: Lie groups, Lie algebras, and representations: an elementary introduction, vol. 222. Springer (2015)
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

Fig. 2. Qualitative results of PRN-CNN for 3D face shape estimation.

4. Park, S., Lee, M., Kwak, N.: Procrustean regression: A flexible alignment-based framework for nonrigid structure estimation. IEEE Transactions on Image Processing **27**(1), 249–264 (2018)