DeepHand: Robust Hand Pose Estimation by Completing a Matrix Imputed with Deep Features - Supplementary Material

Ayan Sinha* Chiho Choi* Karthik Ramani
Purdue University
West Lafayette, IN 47907, USA
{sinha12, chihochoi, ramani}@purdue.edu

This document serves as supplementary material to the paper, *DeepHand: Robust Hand Pose Estimation by Completing a Matrix Imputed with Deep Features*. We discuss additional details about our matrix model and validate our parameter choices.

1. Matrix Completion

In this section, we mathematically derive the final equation of our matrix completion model which estimates the unknown pose vector $\mathbf{p_2} \in \mathbb{R}^{1\times m}$ from the activation features $\mathbf{D} = [\mathbf{D_1}; \mathbf{d_2}]$ and the known parameter values $\mathbf{P_1}$.

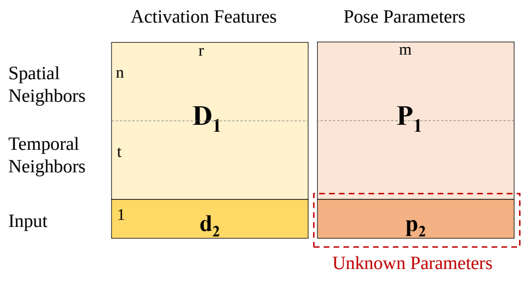


Figure 1: The individual block matrices in matrix completion imputed with deep features.

Figure 1 shows the imputation of the block matrices corresponding to equation (1) in the main manuscript, inclusive of spatial and temporal neighbors. First, n nearest neighbors to the input activation feature are retrieved from the database. These activation features and corresponding annotated parameter values are filled into the matrix block corresponding to $\mathbf{D}$ and $\mathbf{P}$, respectively. Additionally, the activation features corresponding to the t previous frames along with the estimated parameter values serve as temporal neighbors in the matrix blocks, $\mathbf{D}$ and $\mathbf{P}$. Suppose $\mathbf{p_2}$ is a submatrix of the matrix $\mathbf{X}$.

$$\mathbf{X} = \begin{bmatrix} \mathbf{D_1} & \mathbf{P_1} \\ \mathbf{d_2} & \mathbf{p_2} \end{bmatrix} \tag{1}$$

where $\mathbf{D_1} \in \mathbb{R}^{(n+t)\times r}$, $\mathbf{d_2} \in \mathbb{R}^{1\times r}$, and $\mathbf{P_1} \in \mathbb{R}^{(n+t)\times m}$, and r is the dimensionality of the feature vector.

Lemma 1.1. *Suppose that the matrix $\mathbf{X}$ is of rank k and partitioned as shown in equation 1. We assume that the matrix $\mathbf{D_1}$ also has rank k. Then*

$$\mathbf{p_2} = \mathbf{d_2}(\mathbf{D_1})^{+}\mathbf{P_1}, \tag{2}$$

where + denotes the Moore-Penrose pseudo-inverse.

Proof. The matrix $\mathbf{X}$ is decomposed using SVD to rank k as $\mathbf{X} = U\Sigma V'$ where $\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_k)$, $U \in \mathbb{R}^{(n+t+1)\times k}$, and $V \in \mathbb{R}^{(r+m)\times k}$. Assume $U_1 \in \mathbb{R}^{(n+t)\times k}$ and $U_2 \in \mathbb{R}^{1\times k}$. Consequently, $V_1 \in \mathbb{R}^{r\times k}$ and $V_2 \in \mathbb{R}^{m\times k}$. Then, we can rewrite $\mathbf{D_1} = U_1\Sigma V_1', \mathbf{P_1} = U_1\Sigma V_2', \mathbf{d_2} = U_2\Sigma V_1'$, and $\mathbf{p_2} = U_2\Sigma V_2'$. Let $\mathbf{D_1} = LR$, where $L = U_1 S$ and $R = SV_1'$ for $S = \sqrt{\Sigma}$. Using MacDuffee's theorem as done in [1],

$$\begin{aligned} \mathbf{D_1}^{+} &= R^{+}L^{+} \\ &= R'(RR')^{-1}(L'L)^{-1}L' \\ &= V_1 S(SV_1'V_1S)^{-1}(SU_1'U_1S)^{-1}SU_1' \\ &= V_1(V_1'V_1)^{-1}\Sigma^{-1}(U_1'U_1)^{-1}U_1' \end{aligned} \tag{3}$$

As a result, $\mathbf{d_2}(\mathbf{D_1})^{+}\mathbf{P_1}$ equates to

$$\begin{aligned} \mathbf{d_2}(\mathbf{D_1})^{+}\mathbf{P_1} &= (U_2\Sigma V_1')V1(V_1'V_1)^{-1}\Sigma^{-1} \\ &\quad (U_1'U_1)^{-1}U_1'U_1\Sigma V_2' \\ &= U_2\Sigma V_2' \\ &= \mathbf{p_2}. \end{aligned} \tag{4}$$

This completes the proof. □

*These authors made an equal contribution.

In practice, we kernelize the feature matrix $\mathbf{D}$ as radial basis functions (RBF):

$$\mathbf{K}\left(\mathbf{D},\mathbf{D}\right) = \exp\left(-\frac{\|\mathbf{D}^T\mathbf{D}\|^2}{2\sigma^2}\right), \tag{5}$$

where σ denotes the variance of the database (σ=200). The auxiliary knowledge about nearest neighbors is implicitly accounted for in the kernelized similarity matrix $\mathbf{K}$, making the estimation more robust to outliers and noise. Note that the kernelized matrices $\mathbf{K_1}$ and $\mathbf{k_2}$ replace matrices $\mathbf{D_1}$ and $\mathbf{d_2}$ in equation 1 with appropriate dimensions.

$$\mathbf{X} = \begin{bmatrix} \mathbf{K_1} & \mathbf{P_1} \\ \mathbf{k_2} & \mathbf{p_2} \end{bmatrix} \tag{6}$$

where $\mathbf{K_1} \in \mathbb{R}^{(n+t)\times(n+t)}$, $\mathbf{k_2} \in \mathbb{R}^{1\times(n+t)}$, and $\mathbf{P_1} \in \mathbb{R}^{(n+t)\times m}$. We ensure invertibility of matrix $\mathbf{K_1}$ by adding a diagonal matrix, $c\mathbf{I}$ to $\mathbf{K_1}$ where $c = 0.001$. Consequently, the kernelized version of equation 2 can be solved directly without resorting to an intermediary SVD of $\mathbf{K_1}$ which is computationally expensive. This diagonal matrix also acts as a regularizer and prevents overfitting similar in spirit to kernel ridge regression. The final solution is given by:

$$\mathbf{p_2} = \mathbf{k_2}\left(\mathbf{K_1} + c\mathbf{I}\right)^{-1}\mathbf{P_1}, \tag{7}$$

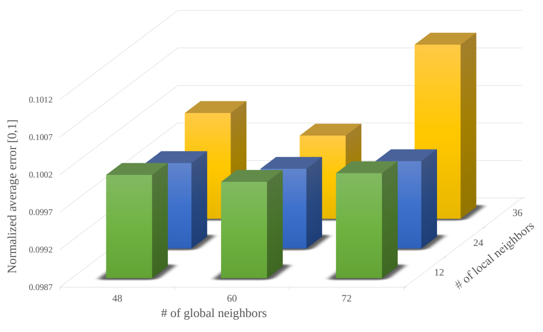
2. System Specifications

2.1. Running and training times

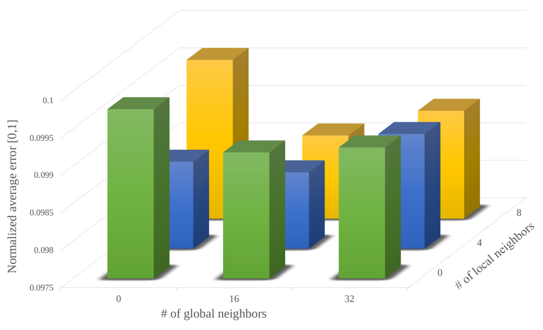
Our hierarchical framework for hand pose estimation takes advantage of multi-threading (OpenMP). The pose parameters in *Stage 2* corresponding to the five finger articulations are evaluated in parallel using five threads. Our system runs at 32 FPS ($\approx$ $31ms$ per frame) on an Intel Xeon E3-1240 CPU with 16GBs RAM. The computation time for each frame is split as $2ms$ for preprocessing (*i.e.*, region of interest extraction, and resizing the depth image to dimension 64×64), $9ms$ for *Stage1* which estimates the global orientation parameters, and $20ms$ for *Stage2* which estimates the local finger articulations. In order to speed up training, the ConvNets were trained with the aid of GPU (NVIDIA Quadro K4000 Graphics card). The training for global orientation parameters took about four hours and local parameters for the 144 bins took about 30 hours.

2.2. Justification of Design Choices

We use 60 spatial neighbors with 16 temporal neighbors for global parameter estimation and 24 spatial neighbors with 4 temporal neighbors, respectively, for all quantitative evaluations in Section 6.3 and 6.4 in the main manuscript. In this subsection, we empirically validate the choice of these parameters. Figure 2a compares the



(a)



(b)

Figure 2: Design choices. Joint angle error is normalized between 0 and 1. (a) Choice of spatial neighbors n. Minimum joint angle error is achieved when the number of neighbors for global pose estimation are 60 and for local estimation are 24. (b) Choice of temporal neighbors t. The system shows highest accuracy with 16 neighbors for global estimation and 4 neighbors for local estimation.

accuracy achieved by using different number of spatial neighbors from the database for global and local parameter estimation on the synthetic database described in the main manuscript. The minimum mean joint angle error is achieved when we use 60 and 24 spatial neighbors for global and local parameter estimation, respectively. A higher number of neighbors is inefficient both in terms of accuracy and time in our DMC model, whereas lesser number of neighbors may result the estimation of joint parameters to be stuck in a local minima. We also conducted experiments to find the balance between spatial and temporal neighbors. In order to reduce jitter in the pose estimates, we add t number of temporal neighbors in the matrix block (*i.e.*, matrix $\mathbf{D}$ and $\mathbf{P}$) as shown in Figure 1. In Figure 2b, we see that 16 temporal neighbors for global and 4 temporal neighbors for local parameter estimation is optimal in terms of achieved accuracy. The lower number of temporal nearest neighbors compared to spatial nearest neighbors indicates that the activation features contain implicit infor-

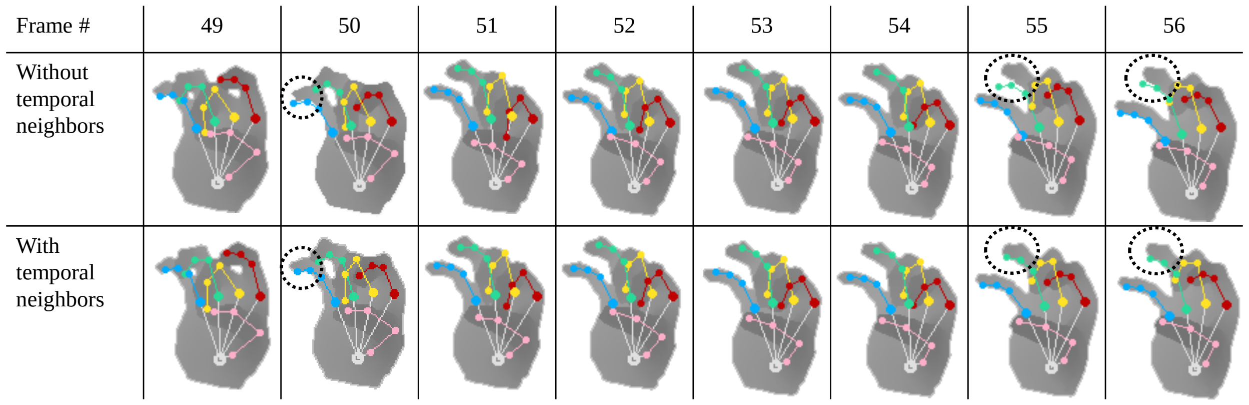


Figure 3: The effect of temporal neighbors on final hand pose estimation. Top row shows the result of DMC without temporal neighbors and bottom row shows the result with temporal neighbors on continuous frames from our synthetic dataset. The dashed circles highlight the increased robustness and reduced jitter of final hand pose by incorporating temporal frames into DMC.

mation about adjacent hand poses. The lower number of temporal nearest neighbors also makes our method robust to rapid hand movements, severe occlusion and other scenarios for which temporal information may not be reliable. However, including temporal nearest neighbors reduces jitter. This effect is displayed in Figure 3. The top row displays the result on continuous frames without incorporating temporal neighbors and the bottom row corresponds to the result by including temporal neighbors. We observe that the resulting hand pose by incorporating temporal neighbors is more robust (see dashed circles), and reduces jitter in a real-time setting.

References

[1] A. B. Owen and P. O. Perry. Bi-cross-validation of the svd and the nonnegative matrix factorization. *The Annals of Applied Statistics*, pages 564–594, 2009. 1