Meta Agent Teaming Active Learning for Pose Estimation (Supplementary Materials)

Jia Gong[1] Zhipeng Fan[2] Qiuhong Ke[3] Hossein Rahmani[4] Jun Liu[1*]
[1]Singapore University of Technology and Design, Singapore; [2]New York University, United States
[3]The University of Melbourne, Australia; [4]Lancaster University, United Kingdom
jia_gong@mymail.sutd.edu.sg, zf606@nyu.edu, qiuhong.ke@unimelb.edu.au
h.rahmani@lancaster.ac.uk, jun_liu@sutd.edu.sg

1. Kinetic Chain Space

In the pose estimation task, the pose estimator aims to predict the positions of the target joints in the hand (or body) from an image. One crucial clue to characterize the pose is the spatial correlation between the target joints. To track this correlation, we transfer the joints' locations to the bone vectors and collect the bones' topological information such as the bones' lengths and rotations via Kinetic Chain Space (KCS) [5, 2].

Given a human hand (or body) image x with its pose y, we can derive M bone vectors from $M+1$ joints' positions in the pose y, as shown in Fig. 1. Here, we denote the bone, which starts from the i^{th} joint and ends at the j^{th} joint, as $\boldsymbol{b}_{i,j}$. Then the bone vector $\boldsymbol{b}_{i,j}$ is expressed as:

$$\boldsymbol{b}_{i,j} = y_j - y_i, \tag{1}$$

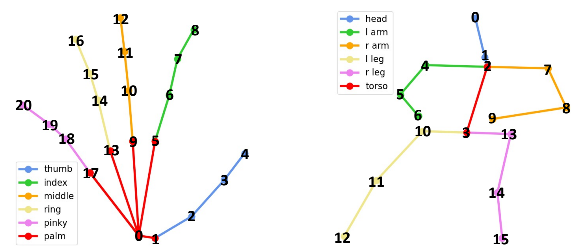
where y_i and y_j are the positions of the i^{th} and the j^{th} joints.

Next, we concatenate all M bone vectors to build a bone matrix $\boldsymbol{B}_{P_0} = [\boldsymbol{b}_{i_1,j_1}; ...; \boldsymbol{b}_{i_M,j_M}]$ for the whole hand (or body), denoted with P_0, where i_m and j_m correspond to the indices of the beginning and end joints of the m^{th} bone. Then the KCS of the whole hand (or body) is the matrix product of the bone matrix $\boldsymbol{B}_{P_0}$ and its transpose as:

$$KCS_{P_0} = \boldsymbol{B}_{P_0}(\boldsymbol{B}_{P_0})^T. \tag{2}$$

Here, the u^{th} diagonal element, i.e., the element at (u, u) of KCS_{P_0}, corresponding to the value of $\boldsymbol{b}_{i_u,j_u}(\boldsymbol{b}_{i_u,j_u})^T$, characterizes the length of the u^{th} bone. The (u, v) element of KCS_{P_0} is the inner product of the bones $\boldsymbol{b}_{i_u,j_u}$ and $\boldsymbol{b}_{i_v,j_v}$, which characterizes the angle between the u^{th} and the v^{th} bones. As KCS_{P_0} is a symmetric matrix, we only use the elements in the upper triangular of KCS_{P_0} to build the global topological feature of the whole hand (or body) as f_{P_0}.

* Corresponding Author



(a) Hand Pose (b) Human Pose

Figure 1. The joints and local parts of the human hand or body. The hand skeleton contains 21 joints and 20 bones while the human body skeleton contains 16 joints and 15 bones. In both cases, we partition the skeleton into 6 local parts and mark each local part with different colors.

Moreover, to observe the performance of the pose estimator on the local parts of the human hand (or body), we also consider the topological property of each finger in the hand (or each limb in the body). For the human hand, as shown in Fig. 1 (a), we derive 20 bone vectors from 21 joints' positions and categorize them into six parts, denoted with $\{P_i\}_{i=1}^{6}$:

- thumb finger P_1: $\{\boldsymbol{b}_{1,2}, \boldsymbol{b}_{2,3}, \boldsymbol{b}_{3,4}\}$;
- index finger P_2: $\{\boldsymbol{b}_{5,6}, \boldsymbol{b}_{6,7}, \boldsymbol{b}_{7,8}\}$;
- middle finger P_3: $\{\boldsymbol{b}_{9,10}, \boldsymbol{b}_{10,11}, \boldsymbol{b}_{11,12}\}$;
- ring finger P_4: $\{\boldsymbol{b}_{13,14}, \boldsymbol{b}_{14,15}, \boldsymbol{b}_{15,16}\}$;
- pinky finger P_5: $\{\boldsymbol{b}_{17,18}, \boldsymbol{b}_{18,19}, \boldsymbol{b}_{19,20}\}$;
- palm P_6: $\{\boldsymbol{b}_{0,1}, \boldsymbol{b}_{0,5}, \boldsymbol{b}_{0,9}, \boldsymbol{b}_{0,13}, \boldsymbol{b}_{0,17}\}$.

Similarly, for the human body, we can obtain 15 bone vectors from 16 body joints, and divide them into six local parts (as shown in Fig. 1 (b)):

- head P_1: $\{\boldsymbol{b}_{0,1}, \boldsymbol{b}_{1,2}\}$;
- torso P_2: $\{\boldsymbol{b}_{2,3}\}$;
- left arm P_3: $\{\boldsymbol{b}_{2,4}, \boldsymbol{b}_{4,5}, \boldsymbol{b}_{5,6}\}$;
- right arm P_4: $\{\boldsymbol{b}_{2,7}, \boldsymbol{b}_{7,8}, \boldsymbol{b}_{8,9}\}$;
- left leg P_5: $\{\boldsymbol{b}_{3,10}, \boldsymbol{b}_{10,11}, \boldsymbol{b}_{11,12}\}$;
- right leg P_6: $\{\boldsymbol{b}_{3,13}, \boldsymbol{b}_{13,14}, \boldsymbol{b}_{14,15}\}$.

For each local part P_i, similarly, we concatenate all the bone vectors in the local part to build the bone matrix B_{P_i} and calculate KCS_{P_i} via Eq. 2 to obtain its local topological feature f_{P_i}. Finally, we obtain the global topological feature f_{P_0} and six local topological features $\{f_{P_1}, f_{P_2}, f_{P_3}, f_{P_4}, f_{P_5}, f_{P_6}\}$ corresponding to the pose y. Note that each global or local topological feature will be used individually to build a corresponding topological space to measure the distribution drifts between the labeled and unlabeled datasets (Please refer to the Sec. 3.1 of the main manuscript). Also note that, in implementation, the local topological features $\{f_{P_1}, f_{P_2}, f_{P_3}, f_{P_4}, f_{P_5}, f_{P_6}\}$ can be derived from the global topological feature f_{P_0} directly to save computation cost.

2. Meta Optimization

Our active learning framework consists of two phases: the **Training Phase** to train the agent team module to learn a cooperative sampling policy, and the **Deployment Phase** to apply the trained agent team to sample unlabeled images. Given an unlabeled dataset, we start our MATAL from the Training Phase. We first randomly sample a small number of images to build the initial dataset D_{init} with annotations, and then simulate the active learning procedures on D_{init} to train the agent team. Next, we freeze the agent team and move to the Deployment Phase in which the agent team raises images from the remaining unlabeled set to be annotated. During this phase, the labeled dataset is initialized by D_{init} at the beginning, and will be updated by adding the newly annotated images. Furthermore, with the growing scale of the labeled dataset, we can also return to the Training Phase to further refine the agent team, i.e., enhancing the performance of our agent team by replacing the initial dataset D_{init} with the enlarged labeled dataset, for re-training the agent team. In implementation, we return to the Training Phase each time the size of the labeled dataset doubles compared to the previous time the agent team was trained. However, this re-training process can still be time-consuming, as it requires simulating the active learning procedures on the expanded labeled dataset. To address this, we adopt a Meta-Learning approach to accelerate the re-training procedure.

More specifically, to enable efficient optimization on the updated labeled dataset, inspired by MAML [1], we propose to learn the meta parameters $\boldsymbol{\theta}_{meta}$ for the agent team, which can quickly adapt to the new labeled sets. We formulate re-training the agent team on the enlarged labeled dataset as a novel task of training on a small subset of it and generalizing well on the rest of the labeled set. Intuitively, this formulation encourages the agent team to quickly adapt to the gradually expanded dataset. To simulate such a process, we partition D_{init} to build a smaller subset as the meta-train set D_{init}^{mtr}, and a larger subset as the meta-test set D_{init}^{mte}.

Our goal then becomes to learn the parameters $\boldsymbol{\theta}_{meta}$ that can adapt quickly to the meta-test set D_{init}^{mte} after being optimized on the meta-train set D_{init}^{mtr}. Here, with randomly initialized agent team parameters $\boldsymbol{\theta}_{meta}$, we first update the agent team on the D_{init}^{mtr} to obtain $\boldsymbol{\theta}_{mtr}^{*}$ following Alg. Teaming Sampling Policy Learning detailed in the main manuscript, and then we employ the updated agent team parameterized by $\boldsymbol{\theta}_{mtr}^{*}$ to perform the active learning steps on D_{init}^{mte} and leverage the loss to update our meta-parameters. We use the Temporal Difference error [3] $TD_{meta}(\boldsymbol{\theta}_{mtr}^{*})$ as the meta loss to update the agent team:

$$TD_{meta}(\boldsymbol{\theta}_{mtr}^{*}) = \sum_{t=0}^{H-1}\Big(\sum_{m=1}^{N} q^m(s_t, a_t^m, h_t^m; \theta_{m-mtr}^{*}) - r_{t+1} - \gamma \sum_{m=1}^{N} q^m(s_{t+1}, a_{t+1}^m, h_{t+1}^m; \theta_{m-mtr}^{*})\Big)^2, \tag{3}$$

where θ_{m-mtr}^{*} is the parameters of the m^{th} agent, and $\boldsymbol{\theta}_{mtr}^{*} = \{\theta_{1-mtr}^{*}, \theta_{2-mtr}^{*}, ..., \theta_{N-mtr}^{*}\}$. Finally, we update $\boldsymbol{\theta}_{meta}$ with the following equation:

$$\boldsymbol{\theta}_{meta} = \boldsymbol{\theta}_{meta} - \beta \nabla TD_{meta}(\boldsymbol{\theta}_{mtr}^{*}), \tag{4}$$

where β is the learning rate of meta-optimization. We detail this Meta-Optimization step in Alg. 3. By minimizing the meta loss, we can obtain the meta parameters $\boldsymbol{\theta}_{meta}$ that enables quick adaptation to the large meta-test set D_{init}^{mte} by updating only based on a relatively small meta-train set D_{init}^{mtr}.

3. Additional Experiments

3.1. Hyper-parameters Study

In this section, we analyze the influence of two hyper-parameters of our MATAL model: **the partition ratio of** $D^{re} : D_{init}^{U} : D_{init}^{L}$, and **the number of agents in the agent team**.

First, we investigate the performance of our MATAL framework under different partition ratios of $D^{re} : D_{init}^{U} : D_{init}^{L}$ on NYU dataset [4], and present the results in Fig. 2.

Algorithm: Meta Optimization

input : The initial set D_{init}, the initial parameter of agent team $\boldsymbol{\theta}_{init}$, pose estimator g, learning rate β

Initialize meta paramters: $\boldsymbol{\theta}_{meta} \leftarrow \boldsymbol{\theta}_{init}$

while *not done* **do**

  $D^{mtr}_{init}, D^{mte}_{init} \leftarrow D_{init}$

  Copy $\boldsymbol{\theta}_{meta}$ as $\boldsymbol{\theta}_{mtr}$

  $\boldsymbol{\theta}^{*}_{mtr} \leftarrow$Update $\boldsymbol{\theta}_{mtr}$ on D^{mtr}_{init} via *Alg. Teaming Sampling Policy Learning*.

  Generate H active learning iterations by the agent team with $\boldsymbol{\theta}^{*}_{mtr}$ on D^{mte}_{init}:$\{s_t, a_t, r_{t+1}, s_{t+1}\}^{H}_{t=1}$

  Calculate the loss of $\{s_t, a_t, r_{t+1}, s_{t+1}\}^{H}_{t=1}$ via Eq. 3 to obtain $TD_{meta}(\boldsymbol{\theta}^{*}_{mtr})$

  Update $\boldsymbol{\theta}_{meta} \leftarrow \boldsymbol{\theta}_{meta} - \beta \nabla TD_{meta}(\boldsymbol{\theta}^{*}_{mtr})$

end

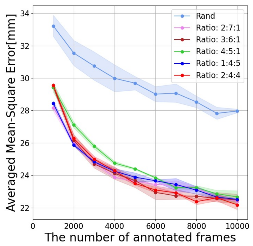


Figure 2. The results for various partition ratios of D^{re} : D^{U}_{init} : D^{L}_{init}.

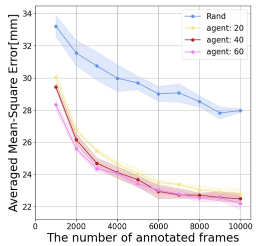


Figure 3. The results for different numbers of agents.

Our MATAL performs effective selection under various partition ratios compared to the random sampling, and it obtains the best performance when the partition ratio is 3:6:1. This also shows that our MATAL is stable for a wide range of partition ratios.

Moreover, we look into the ability of the agent team to complete effective batch image selection with different numbers of agents. Here, we increase the numbers of agents from 20 to 60 and evaluate the performance of MATAL on NYU dataset [4]. As shown in Fig. 3, our proposed MATAL framework with different numbers of agents all outperform the random sampling significantly.

3.2. Quantitative Results

We first compare the qualitative results of our MATAL with two baselines: 'single-agent selecting multiple images' and 'random sampling'. Here, we pick up two depth images from NYU test set, and visualize the recovered poses of both images in five different active learning iterations, as shown in Fig. 4. We observe that our MATAL can rapidly improve the recovered poses and build the most realistic poses at the end of the active learning procedure. The poses recovered by the single agent also tend to be more natural than random sampling, but their qualities are still significantly lower than our Multi-Agent method.

Moreover, we present the poses of selected images by our MATAL in five different active learning iterations on NYU training set. As shown in Fig. 5, our MATAL generally selects the images that the pose estimator cannot give accurate prediction in each active learning iteration. Comparing the poses of selected images across different stages of the active learning procedure (i.e., horizontally), we can observe that the selected poses at the early stages look more common and occur more frequently in our daily life. On the other hand, the images selected by our MATAL at the end of the active learning procedure tend to be more complex and

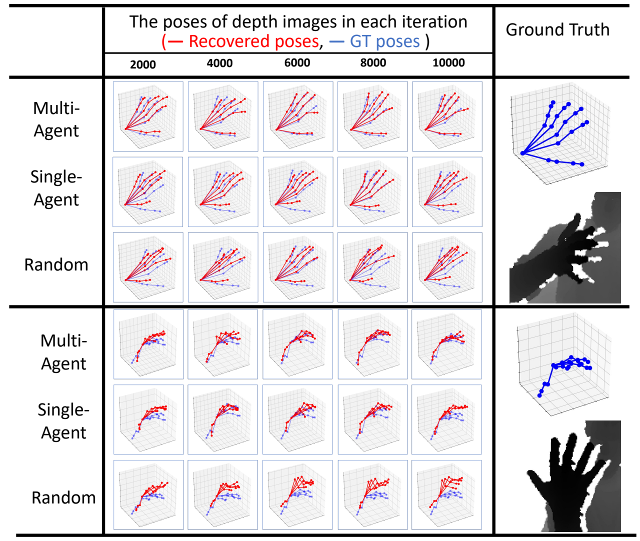


Figure 4. Qualitative comparison for the recovered poses in different active learning iterations. The blue skeletons are the ground truth and the red ones are the poses recovered by the pose estimator.

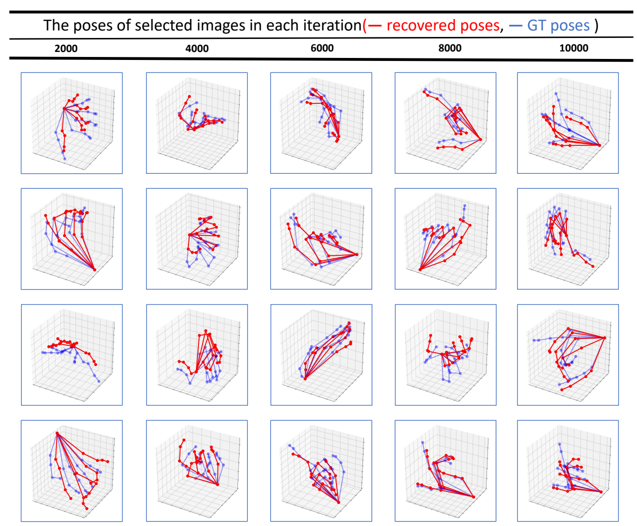


Figure 5. Qualitative results for the selected images' poses by our MATAL in different active learning iterations. The blue skeletons are the ground truth poses and the red ones are the poses recovered by the pose estimator.

less common. It shows that our MATAL tends to pick more representative and influencing images at the beginning to quickly boost the performance of the pose estimator, while at the later stages, the agent team focuses more on the challenging cases that are difficult for the current pose estimator.

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[2] Kehong Gong, Jianfeng Zhang, and Jiashi Feng. Poseaug: A differentiable pose augmentation framework for 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8575–8584, 2021.

[3] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[4] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):1–10, 2014.

[5] Bastian Wandt, Hanno Ackermann, and Bodo Rosenhahn. A kinematic chain space for monocular motion capture. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.