Real-time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor – Supplemental Document –

Franziska Mueller[1,2] Dushyant Mehta[1,2] Oleksandr Sotnychenko[1]
Srinath Sridhar[1] Dan Casas[3] Christian Theobalt[1]
[1]Max Planck Institute for Informatics [2]Saarland University [3]Universidad Rey Juan Carlos
{frmueller, dmehta, osotnych, ssridhar, dcasas, theobalt}@mpi-inf.mpg.de

This document includes additional experiments (Section 1), details about our new synthetic dataset *SynthHands* (Section 2), and information about our custom CNN architecture and training procedure (Section 3). We also refer to our supplemental video[1] for more visual results.

1. Evaluation

In this section, we show additional experiments and comparisons of our method.

Improvement by Combining 2D and 3D Predictions: Figure 3 qualitative depicts the predicted joint locations by each of the key components of our pipeline — 2D predictions, 3D predictions, and final tracked skeleton — on the test sequence `Fruits`. Note that the modes of failure for the 2D and 3D predictions are not the same which leads to accurate skeleton tracking even if one kind of prediction is incorrect. Thus, the combination of 2D and 3D predictions with the tracking framework consistently produces the best results.

Comparisons to the State of the Art: We were unable to quantitatively evaluate on the only other existing egocentric hand dataset [7] due to a different sensor currently unsupported by our approach. Using our method with the *Senz3D* camera requires adaptation of intrinsic camera parameters and noise characteristics for training. However, to provide a visual comparison to evaluate our method, we recorded sequences that mimic sequences used by Rogez *et al*. [7], and show qualitative evaluation in Figure 4 and in the supplementary video. Our method achieves significantly more accurate hand tracking, while running in real time.

To show the completely different nature of our problem which cannot be solved by employing state-of-the-art methods for hand tracking in free air, we applied the method of Sridhar *et al*. [8] to our real test sequences from *EgoDexter*. Figure 6 demonstrates catastrophic failures of the aforementioned approach. On the other hand, Figure 5 and the supplemental video show how our method successfully tracks sequences mimicked from the work of Sridhar *et al*. We achieve comparable results, with improved stability of the hand root position.

Performance on 3rd-Person Viewpoint: Even though the machine learning components of our method were only trained on our egocentric dataset *SynthHands*, Figure 7 demonstrates the generalizability to 3rd-person views. Note that the hand localization step is robust to other skin-colored parts, like faces, in the input.

Analysis of Failure Cases: Despite the demonstrated success of our approach, we still suffer from limitations that produce erroneous tracking results. Figure 8 depicts the results from several intermediate steps of our method in case of failure. In particular, we show errors due to extreme self occlusions, severe hand and object occlusions, and when the hand is located out of the camera field of view.

Table 1: SynthHands Details

Mode of Variation	Amount of Variation
Pose	63,530 frames of real hand motion, sampled every 5th frame
Wrist+Arm Rotation	wrist: sampled from 70 deg. range arm: sampled from a 180 deg. range
Shape	x, y, z scale sampled uniformly in [0.8, 1.2]; female + male mesh
Skin Color	2 x 6 hand textures (female/male)
Camera Viewpoints	5 egocentric viewpoints
Object Shapes	7 objects
Object Textures	145 textures
Background Clutter	10,000 real images, uniform random u,v offset in [-100, 100]

[1]http://handtracker.mpi-inf.mpg.de/projects/OccludedHands/

2. *SynthHands* Dataset

Table 1 shows the modes of variation in the *SynthHands* dataset. Representative frames are shown in Figure 2.

3. CNN Architecture and Training

In this section, we explain the network architecture used for *HALNet* and *JORNet* and provide training details. Furthermore, we present experiments which lead to our specific design decisions.

3.1. Network Design

The ResNet[2] architecture has been successfully used for full body pose estimation in previous work [6]. While ResNet50 offers a good tradeoff between speed and accuracy, hand motion is fast and exhibits rapid directional changes. Further, the egocentric camera placement leads to even faster relative motion in the scene. Therefore, we experimented based on recent investigations into the nature of representations learned by ResNets [1] to get a faster architecture without significantly affecting the accuracy.

Core Architecture: Starting from ResNet50, we remove a residual block from level 3, and keep only 4 residual blocks at level 4. Level 5 is replaced with two 3×3 convolution layers with 512 (conv4e) and 256 (conv4f) features and no striding. Both of these layers also use batch normalization [3]. From *conv4f*, A 3×3 convolutional stub followed by bilinear upsampling produces the joint location heatmaps, and a fully-connected layer with 200 nodes followed by another fully-connected layer predict the joint location vector. See Figure 9 for details.

The resulting architecture needs 10 ms for a forward pass at resolution 320 x 240 (*HALNet*) and 6 ms at resolution 128 x 128 (*JORNET*) on a Nvidia Pascal Titan X GPU. This is a significant speed-up compared to the ResNet50 version which needs 18 ms and 11 ms, respectively. Evaluation on the *SynthHands* test set shows that the drop in accuracy is only marginal. ResNet50 trained on the hand localization task achieves 2.1 px average error whereas *HALNet* achieves 2.2 px.

Intermediate Supervision: For *HALNet* and *JORNet*, we treat a subset of the feature maps at each of *res3a*, *res4a* and *conv4e* in the networks as the predicted heatmaps, for intermediate supervision [5]. For *JORNet*, we additionally use the feature maps at the aforementioned stages to predict joint positions for intermediate supervision (see Figure 9).

Auxiliary Task in *HALNet*: Predicting heatmaps for all joints as auxiliary task helps *HALNet* to learn the structure of the hand. This leads to a better performance compared to a version only trained to regress the root heatmap. On the *SynthHands* test set, regressing heatmaps for all joints instead of only for the root improves the 2D pixel error by **6.4%** (from 2.35 px to 2.2 px).

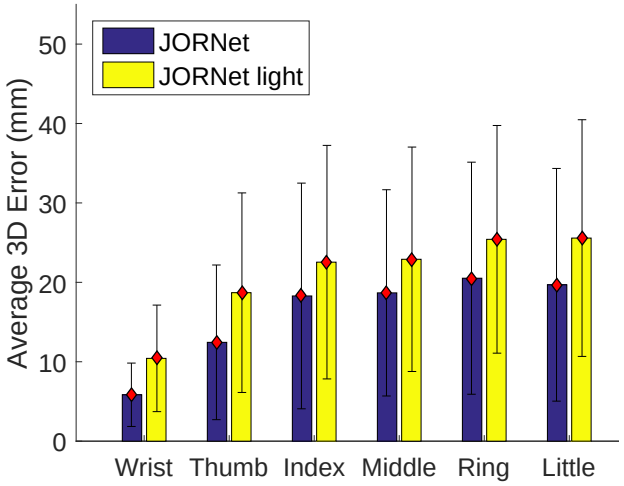


Figure 1: Training *JORNet* to regress heatmaps and local joint positions for all joints instead of only for fingertips and the wrist (*JORNet light*) reduces the error on the *SynthHands* test set.

Regressing All Joints in *JORNet*: Although fingertips and wrist location alone provide a strong constraint for the pose of the hand, training *JORNet* to regress the heatmaps and local 3D positions for all joints improves the accuracy. Figure 1 shows the average error for the wrist and all fingertips in 3D on the *SynthHands* test set. The full *JORNet* version yields a significant increase in performance compared to *JORNet light* which was only trained for wrist and fingertips.

3.2. Training Details

We use the Caffe [4] framework for training our networks, using the AdaDelta scheme with momentum set to 0.9 and weight decay to 0.005. Both networks are trained with an input batch size of 16. For *HALNet*, we use a base learning rate of 0.05 and train for 45k iterations. The input has a spatial resolution of 320x240 px, and the output heatmaps have a resolution of 40x30 px. The main heatmap loss has a loss weight of 1.0, and all intermediate heatmap losses have loss weights of 0.5. For *JORNet*, the input has a spatial resolution of 128x128 px. We train with a base learning rate of 0.05, with main heatmap loss weight set at 1.0 and joint position loss weight at 2500. The intermediate heatmap losses have their loss weights set to 0.5 and intermediate joint position loss weights set to 1250. After 45k iterations, the base learning rate is lowered to 0.01, the intermediate heatmap loss weights lowered to 0.1 and the intermediate joint position loss weights lowered to 250, and trained for a further 15k iterations.

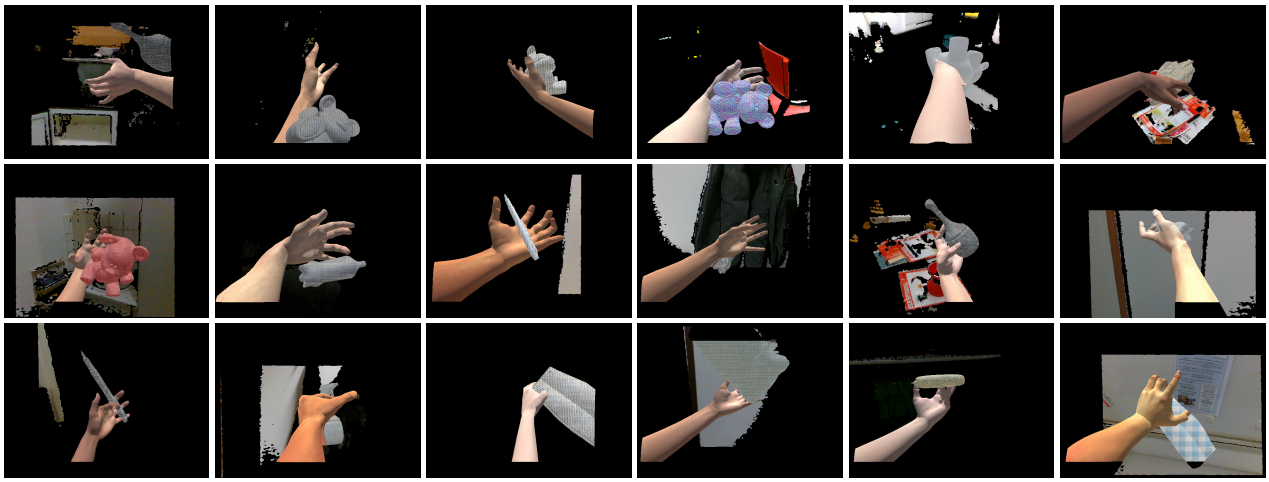


Figure 2: An example set of random samples taken from our *SynthHands* dataset. See Table 1 for description of dataset variability.

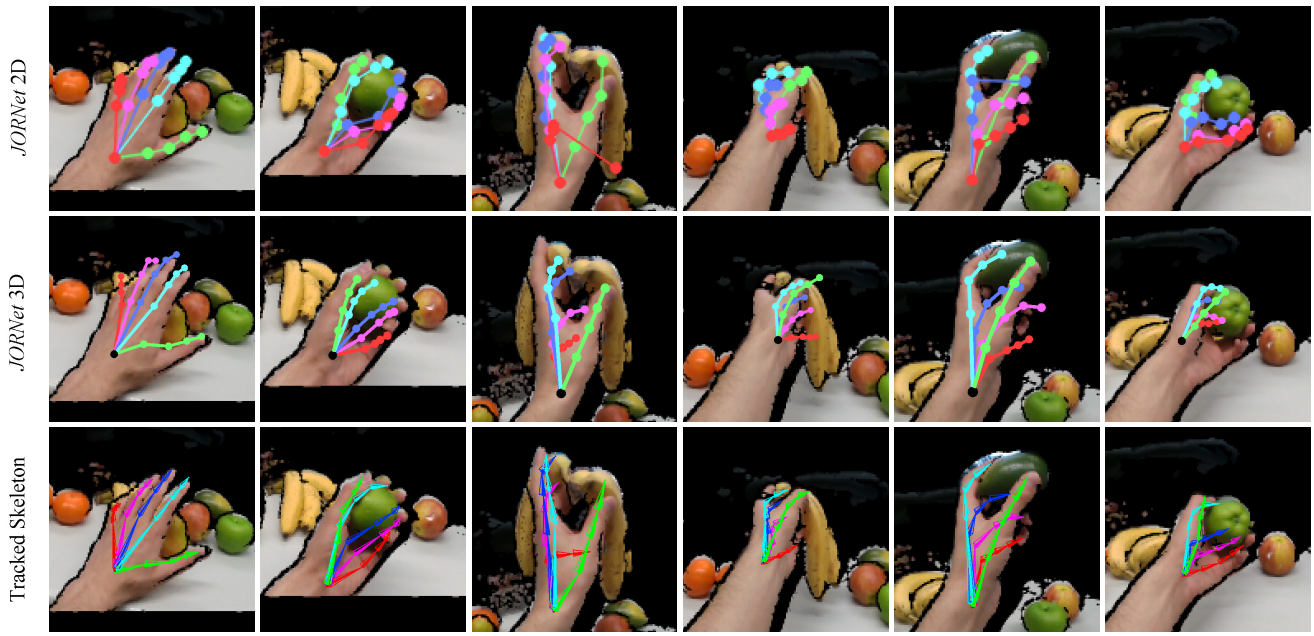
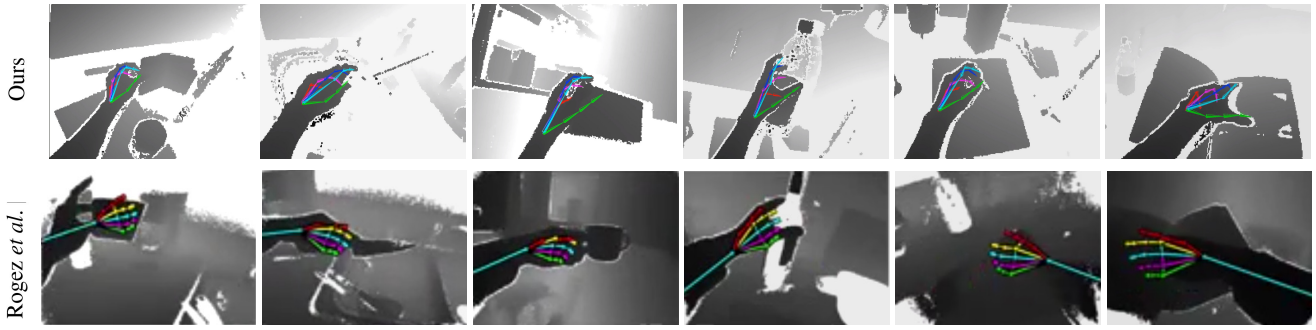


Figure 3: 2D predictions (top row), 3D predictions (middle row) and tracked skeleton (bottom row) on our real annotated sequence `Fruits`. The combination of 2D and 3D predictions in the tracking framework leads to better results than either of the predictions in isolation.



(a) Pinch sponge (b) Grab Knife (c) Reach cupboard (d) Grab bottle (e) Write in notebook (f) Open book

Figure 4: Qualitative evaluation of our results (top) and Rogez *et al*. [7] (bottom). We mimic the motions originally used by [7] because, due to sensor differences (*i.e*. lens intrinsics, etc.), we cannot directly run our trained CNNs on their data.

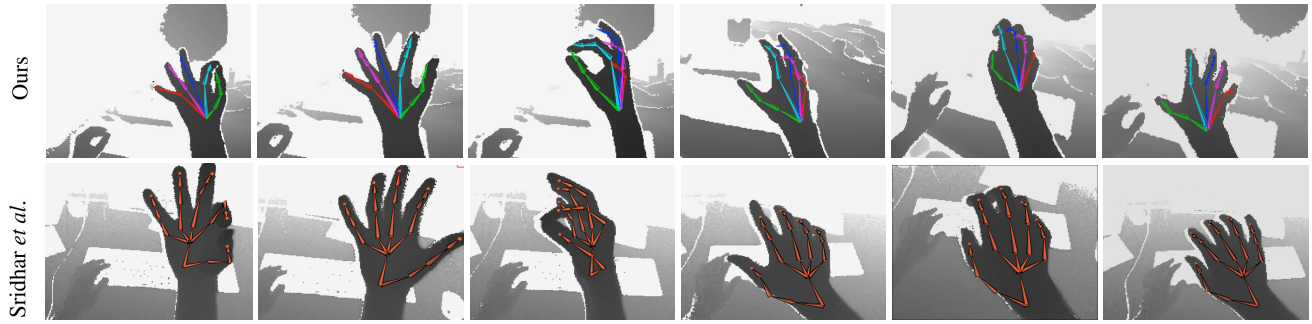


Figure 5: Qualitative evaluation of our results (top) and Sridhar *et al*. [8] (bottom).

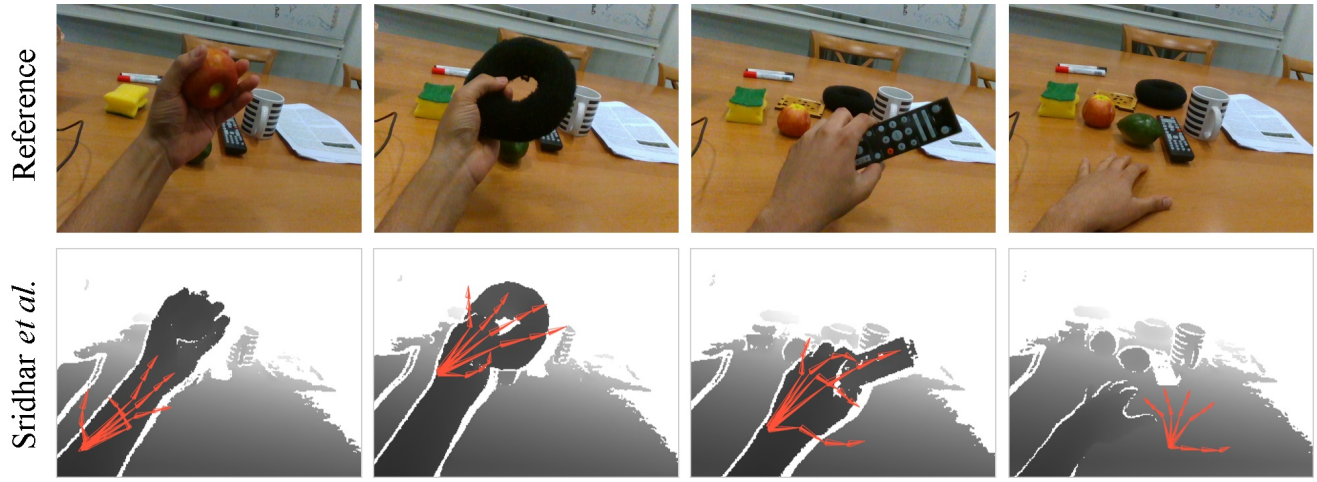


Figure 6: Qualitative results produced by the approach of Sridhar *et al*. [8] on our benchmark dataset *EgoDexter*. The close proximity of the arm to the camera and the interaction with objects and the environment leads to catastrophic failures.

Figure 7: Despite of being trained for egocentric views, our approach in fact generalizes to 3rd-person views. In addition, it is robust to the presence of other skin-colored body parts.

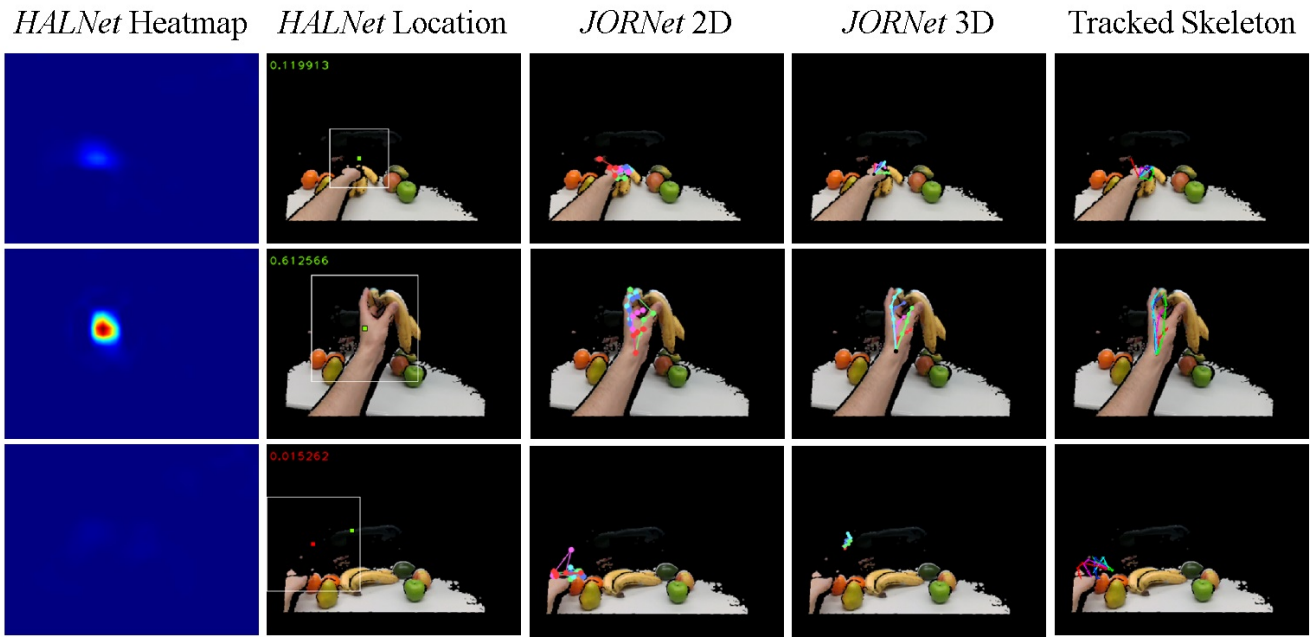


Figure 8: The failure cases of our method can be caused by different intermediate steps. The first two columns show the output from *HALNet*, a heatmap and its maximum location with corresponding confidence. Note that the root stabilization step improved the location in the last row (from green to red) but did not succeed. The third, fourth and fifth column shows 2D predictions, 3D predictions, and the tracked kinematic skeleton, respectively.

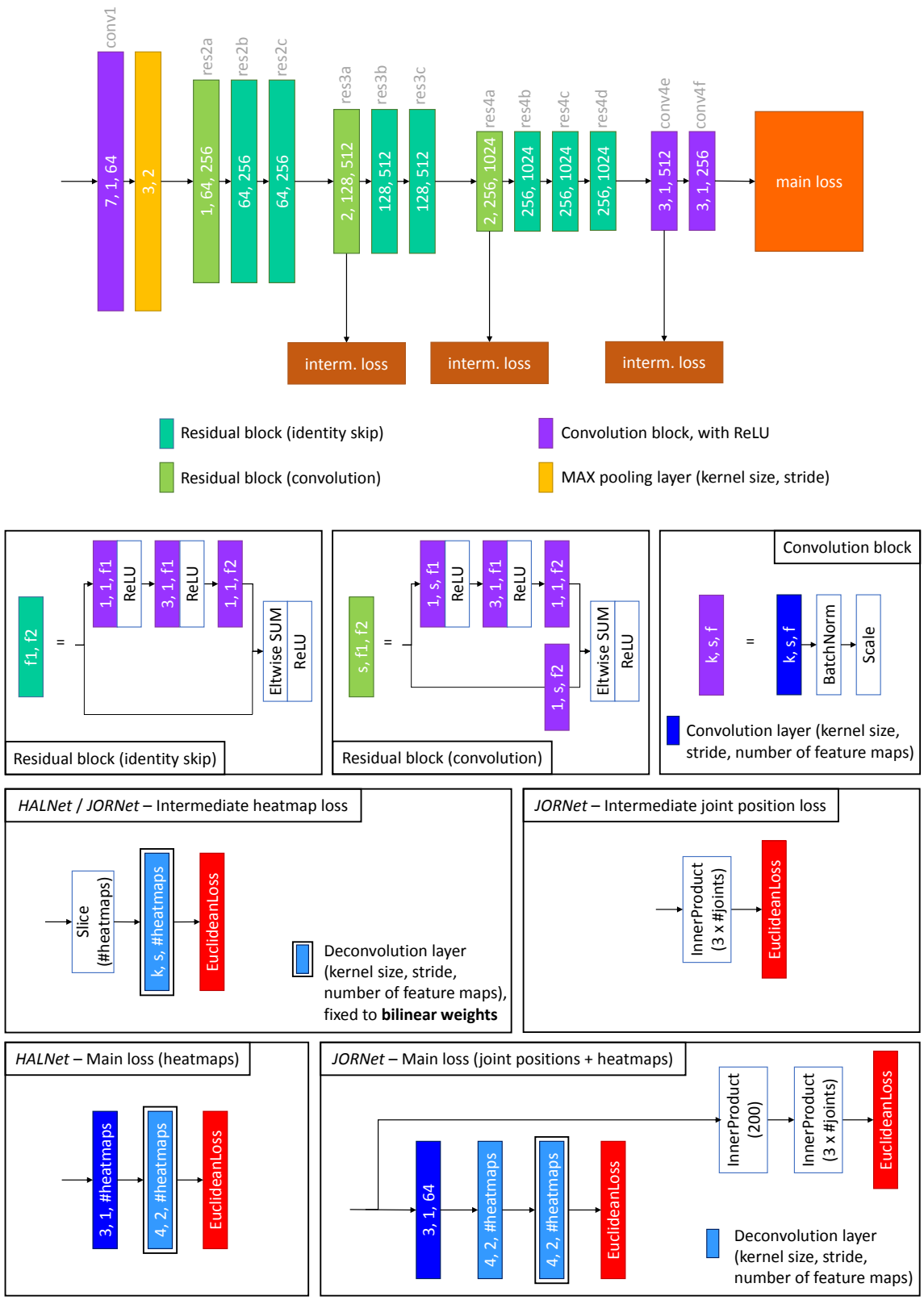


Figure 9: Our network architecture.

References

[1] K. Greff, R. K. Srivastava, and J. Schmidhuber. Highway and residual networks learn unrolled iterative estimation. In *International Conference on Learning Representations*, 2016. 2

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[3] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015. 2

[4] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 675–678, 2014. 2

[5] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, volume 2, page 6, 2015. 2

[6] D. Mehta, H. Rhodin, D. Casas, O. Sotnychenko, W. Xu, and C. Theobalt. Monocular 3d human pose estimation using transfer learning and improved CNN supervision. *arXiv preprint arXiv:1611.09813v2*, 2016. 2

[7] G. Rogez, J. S. Supancic, and D. Ramanan. First-person pose recognition using egocentric workspaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4325–4333, 2015. 1, 4

[8] S. Sridhar, F. Mueller, A. Oulasvirta, and C. Theobalt. Fast and Robust Hand Tracking Using Detection-Guided Optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 4