A. Supplementary Material for KeyTrack

A.1. Test Set Scores

We submitted to the PoseTrack 2017 test set twice. We first achieved a **60.1** MOTA score, but then decreased the TOKS box expansion value from $\alpha = 1.4$ to $\alpha = 1.25$. This increased our our score to **61.2**. $\alpha = 1.25$ is also the value we used on the 2018 Validation Set.

A.2. Additional Qualitative Results

We provide additional qualitative results of our model on the PoseTrack 18 Validation Set in Figure 10.

A.3. Keypoint Postprocessing

The post-processing performed when evaluating AP and MOTA is different. Specifically, we use a different keypoint confidence threshold, where keypoints above the threshold are kept and keypoints below the threshold are ignored. The confidence metric used is the per-keypoint confidence score from the pose estimator. The threshold optimal for MOTA is much higher than AP. Interestingly, ID Switches are not much worse, indicating the majority of the error stems from the estimation step. Results are in Table 4.

Confidence Threshold	AP	% IDSW	MOTA
0.05	**81.6**	1.0	42.0
0.35	79.6	0.9	63.3
0.5	76.7	0.9	66.5
0.57	74.3	**0.8**	**66.6**
0.6	72.8	0.9	66.0

Table 4. Effect of postprocessing on the 2018 Validation Set.

A.4. Implementation Details

Training To fine-tune the detector, separate models are fine-tuned on PoseTrack 17 and 18 datasets for 1 epoch with a learning rate of 1.9×10^{-3} and batch size of 4. Training was conducted on 4 NVIDIA GTX Titans. To fine tune the pose estimator, originally trained on COCO, we follow [47].

During tracking training, we use a linear warm up schedule for learning rate, warming up to 1×10^{-4}, for a fraction of 0.01 of total training steps, then linearly decay to 0 over 25 epochs. Batch size is 32. Cross entropy loss is used to train the model. Since there are more non-matching poses than matching poses in a pair of given frames, we use Pytorch's WeightedRandomSampler to sample from matching and non-matching classes equally, accounting for class imbalance. When assigning a track ID to a pose, we choose the maximum match score from the previous 4 timesteps. All models are trained on 1 NVIDIA GTX 1080Ti GPU.

Inference The detector processes images with a batch size of 1. The detections are fed to the pose estimator, which processes all of the bounding box detections for a frame in a single batch. Flip testing is used. Temporal OKS is computed for every frame with an OKS threshold of 0.35. The bounding box scores are ignored when computing OKS. Bounding boxes are thresholded at a minimum confidence score of 0.2, and keypoints are thresholded at a minimum confidence score of 0.1. We found decreasing the bounding box confidence and keypoint thresholds to 0 did not improve AP, but hurt runtime. Boxes are enlarged by factor $\alpha = 1.25$. All code is written in Python, and we use 1 NVIDIA GTX 1080ti. As done by [47, 21], we train on the PoseTrack 2017 Train and Validation sets before evaluating on the heldout Test Set.

Details of the Tracking Pipeline Analysis The ablation studies from 5.1 were conducted using the predicted keypoints and predicted boxes with our best model on the PoseTrack 2018 Validation Set. Match accuracy, the metric we use in Table 3 is similar to $1 - \%IDSW$, i.e. the IDs which are not switched. The methods would be in the same order if measured with IDSW.

A.5. Architecture Details

Detector and Estimator We use the implementation of the COCO-pretrained Hybrid Task Cascade Network [10] with Deformable Convolutions and Multi Scale predictions from [11]. For our pose estimator, we use the most accurate model from [47], HRNetW48-384x288.

Transformer Matching Network We use an effective image resolution of 24×18 for a total of 432 unique Position tokens. There are $|\mathcal{K}| = 15$ Type tokens and 4 Segment Tokens.

Each pair of poses has $2|\mathcal{K}|$ tokens total. These are projected to embeddings with dimension $[2|\mathcal{K}|, H]$, where $H = 128$ is the transformer hidden size (this is also the transformer intermediate size). The sum of each token's embedding is input to our Transformer Matching Network. The network's backbone consists of 4 transformers in series, each with 4 attention heads. We use a probability of 0.1 for dropout, applying it throughout our Network as [15]. Weights are initialized from a standard normal, $\mathcal{N}(0, 0.02)$. The output is pooled, then fed to a binary classification layer, $[H, 2]$. The network has a total of 0.41M parameters, we adapt code from [49]. A.5 gives details of our transformer, which follows the original architecture. The inputs are the hidden states, $[B, 2|\mathcal{K}|, H]$, where B is batch size, and an attention mask, $[B, 1, 1, 2|\mathcal{K}|]$. The extra dimensions in the attention mask are for broadcasting in matrix multiplication. The FLOP counts for our Transformer Matching Network are in Table 6.

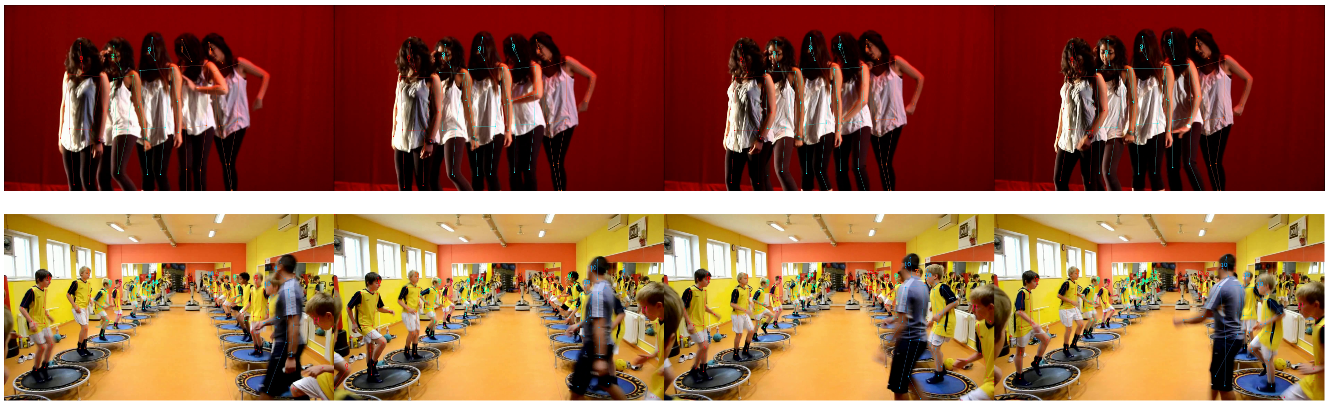
Figure 9. Two videos which highlight the limitations of our model. In the top example, the individuals are very near each other and are moving in a synchronized fashion. Thus, our model incorrectly ids people in the middle of the group. In the bottom row, a man walks in front of boys on trampolines. They are occluded for a few frames and are given incorrect ids after he walks away from them. Also, some of the individuals in the back are given incorrect ids because they are small, in close proximity, and moving in similar fashions.

element 1	op	element 2	output
hidden states $[32, 30, 128]$	x	W^Q $[128, 128]$	Q $[32, 30, 128]$
hidden states $[32, 30, 128]$	x	W^K $[128, 128]$	K $[32, 30, 128]$
hidden states $[32, 30, 128]$	x	W^V $[128, 128]$	V $[32, 30, 128]$
Q $[32, 30, 128]$	resize	-	$Q_{multi}[32, 4, 30, 32]$
K $[32, 30, 128]$	resize	-	$K_{multi}[32, 4, 32, 30]$
V $[32, 30, 128]$	resize	-	$V_{multi}[32, 4, 30, 32]$
A_{scores} $[32, 4, 30, 30]$	+	attention mask	A'_{scores} $[32, 4, 30, 30]$
A'_{scores} $[32, 4, 30, 30]$	softmax	-	A_{probs} $[32, 4, 30, 30]$
A_{probs} $[32, 4, 30, 30]$	dropout	-	A_{probs} $[32, 4, 30, 30]$
A_{probs} $[32, 4, 30, 30]$	x	$V_{multi}[32, 4, 30, 32]$	context $[32, 4, 30, 32]$
context $[32, 4, 30, 32]$	resize	-	context $[32, 30, 128]$

Table 5. A look inside our transformer. 32 is the batch size. x is matrix multiplication., Q, K, V are the query, key, and value, respectively. $W*$ are the learned weights corresponding to the query, key, or value. "multi" refers to a multi-headed representation. A_{scores} are the raw attention scores, and A_{probs} is the distribution of attention scores resulting from the softmax operation.

Network Module	Parameters (M)	FLOPS (M)
Embeddings	0.06	0.35
Transformers (x4)	0.40	5.84
Pooler	0.02	0.015
Classifier	0.01	0.02
Transformer Matching Network	0.41	6.20
GCN	0.1	1.30
Optical Flow	38.7	52.7×10^3

Table 6. FLOP and parameter comparison of our Transformer Matching Network to alternative tracking methods. The first four rows give details of each component of our network. (M) indicates millions. Its computational cost is similar to a GCN, only amounting to 1ms increase on the GPU, and much more efficient than Optical Flow. As we showed earlier, our method is more accurate than both alternatives.

We also give details about the other tracking methods we compare to in Table 4. Though our method is slightly more computationally expensive than the GCN, it is much more accurate. Both Transformers and the GCN are far less computationally expensive than Optical Flow.

CNN Pose Entailment Networks The input is projected to 64 channels in the first layer of the CNN. All convolutions use kernel size 3 and padding 1. BatchNorm is applied after each convolutional layer. The input is downsampled via a maxpooling operation with a stride of 2. The number of filters are doubled after downsampling. Two Linear layers complete the network. The hidden size is dependent on the resolution of the input image. The second layer outputs a binary classification, corresponding to the likelihood of the poses being a match or non-match.

The number of convolutions layers is equal to $log_2(n) - 1$, where n is the long edge of the input image. The batch size, learning rate, and number of training epochs are the same as those we used for the transformers. We experimented with other learning rates, but did not see improvement.

The scheme to color the "visual tokens" is accomplished by fixing the Hue and Saturation, then adjusting the Value via a linear interpolation from $0 - 100\%$ in increments of $\frac{100}{|\mathcal{K}|}$.

A.6. Limitations

Our approach can struggle to track people who are in close proximity and are moving in similar patterns. This is similar to how CNNs struggle with people in close proximity who look visually similar, such as the case where they are wearing the same uniform. Another challenging case for our model is people who are hidden for long periods of time. It is difficult to re-identify them without visual features, and we would need to take longer video clips into context than we currently do to successfully re-identify these individuals. We visualize both these failure modes in Figure 9.

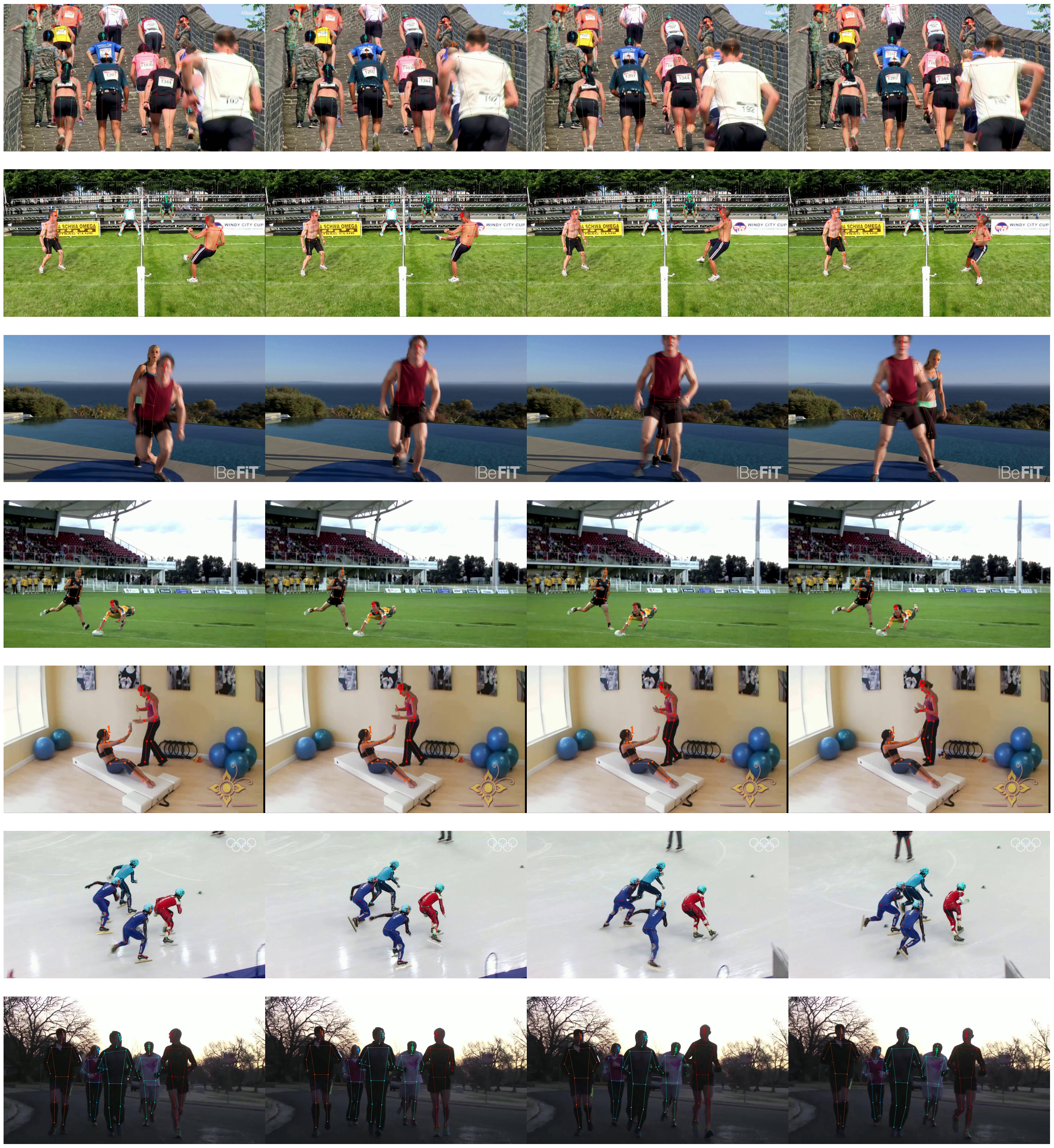
Figure 10. Additional qualitative results of our model succeeding in scenarios despite occlusions, unconventional poses, and varied lighting conditions. Every 4th frame is sampled so more extensive motion can be shown. Solid circles represent predicted keypoints. Hollow squares represent keypoint predictions that are not used due to low confidence.