# Probabilistic Tracklet Scoring and Inpainting for Multiple Object Tracking (Supplementary Material)

Fatemeh Saleh[1,2], Sadegh Aliakbarian[1,2], Hamid Rezatofighi[3], Mathieu Salzmann[4,5], Stephen Gould[1,2]
[1]Australian National University, [2]ACRV, [3]Monash University, [4]CVLab, EPFL, [5]ClearSpace
fatemehsadat.saleh@anu.edu.au

## 1. Architecture Details

As illustrated in Fig. 2 of the main paper, our novel model consists of two subnetworks, MA-Net and ArTIST. These two subnetworks are trained jointly. In this section, we introduce the architecture and implementation details of each of these subnetworks.

**MA-Net.** MA-Net is a recurrent autoencoder that is trained to capture the representation of motion of all agents in the scene. This is achieved by learning to reconstruct the motion of tracklets. The subnetwork consists of an encoder that takes as input a 4D motion velocity representation, passes it through a fully-connected layer with 128 hidden units and a ReLU non-linearity, followed by a single GRU with 256 hidden units. The last hidden state of this (encoder) GRU initializes the hidden state of the decoder's GRU. The decoder is based on a residual GRU network that learns the velocity of changes in motion. To this end, given the initial hidden state and a seed 4D motion velocity representation (the velocity of changes between the first two consecutive frames), the decoder reconstructs each tracklet autoregressively. On top of each GRU cell of the decoder, there exists a MLP that maps the hidden representation to a 4D output representation, *i.e.*, the reconstructed velocity of motion at each time-step.

**ArTIST.** ArTIST takes as input a 4D motion velocity representation and a 256D interaction representation. The motion velocity is first mapped to a higher dimension via a residual MLP, resulting in a 512D representation. We then combine this with the interaction representation through concatenation. The resulting representation is then passed through a fully-connected layer that maps it to a 512D representation, followed by a ReLU non-linearity. This then acts as the input to a single layer LSTM with 512 hidden units to process the sequence. The LSTM produces a residual 512D vector, which is appended to its input to generate the final representation. To map the output of the LSTM to a probability distribution for each component of the motion velocity, we use 4 fully-connected layers (mapping 512D to $K$D) followed by softmax activations, resulting in a $4 \times K$ representation, where $K = 1024$ is the number of clusters.

## 2. Implementation Details

We train our model on a single GTX 2080Ti GPU with the Adam optimizer [2] for 110K iterations. We use a learning rate of 0.001 and a mini-batch size of 256. To avoid exploding gradients, we use the gradient-clipping technique of [4] for all layers in the network. Since we use the ground-truth boxes during training, we apply random jitter to the boxes to simulate the noise produced by a detector. We train our model with sequences of arbitrary length (in range $[5, 100]$) in each mini-batch. During training, we use the teacher forcing technique of [7], in which ArTIST chooses with probability $P_{tf}$ whether to use its own output (a sampled bounding box) at the previous time-step or the ground-truth bounding box to compute the velocity at each time-step. We use $P_{tf} = 0.2$ for the frames occurring after 70% of the sequence length. For our online tracking pipeline, we terminate a tracklet if it has not been observed for 30 frames. For tracklet rejection in the case of inpainting, we use an IOU threshold of 0.5 and set $t_{TRS} = 1$ for low frame-rate videos and $t_{TRS} = 2$ for high frame-rate ones. During multinomial sampling, we sample $\mathcal{S} = 50$ candidate tracklets. Note that, we also use the PathTrack [3] dataset, containing more than 15,000 person trajectories in 720 sequences, to augment MOT benchmark datasets. We implemented our model using the Pytorch framework of [5].

## 3. ArTIST Pseudo-code for Tracking

In Algorithm 1, we provide the pseudo-code of our tracking algorithm. Following our discussion in Section 3 of the main paper, given the trained ArTIST model, detections, and current tracklets, this algorithm demonstrates how our approach updates tracklets at each time-step.

**Algorithm 1** ArTIST tracking at time $t$

**procedure** TRACKING($D^t$, $\mathbb{T}$, $S$)
  Cost = `zeros`($|\mathbb{T}|$, $|D^t|$) ▷ The cost matrix
  FullSeq = [ ] ▷ List of fully observed tracklets
  InpaintSeq = [ ] ▷ List of partially observed tracklets

  **for** $\mathcal{T}_j$ in $\mathbb{T}$ **do**
    $\Delta_j$ = `seq2vel`($\mathcal{T}_j$) ▷ Compute motion velocity
    $I_j$ = **agg**(MA-Net.`encode`($\mathbb{T} \setminus \{\mathcal{T}_j\}$)) ▷ Compute interaction representation

    **if** gap($\mathcal{T}_j$) == 0 **then** ▷ Handle tracklets with full observation
      FullSeq.`append`($j$)
      $p(bbox^t_{\mathcal{T}_j})$ = ArTIST($\Delta_j$, $I_j$) ▷ Compute the likelihood of next plausible bounding box
      **for** $d^t_i$ in $D^t$ **do** ▷ Compute the cost of assigning detections to $\mathcal{T}_j$
        Cost[$j$][$i$] = NLL($p(bbox^t_{\mathcal{T}_j})$, $d^t_i$)

    **if** gap($\mathcal{T}_j$) $> 0$ **then** ▷ Handle tracklets that require inpainting
      InpaintSeq.`append`($j$)
      $\widehat{\Delta}_{j,[1:S]}$ = `Inpaint`($\Delta_j$, $I_j$, gap($\mathcal{T}_j$), $S$) ▷ Inpaint $S$ continuations for gap($\mathcal{T}_j$) time-steps
      $\widehat{\Delta}_j$ = **TRS**($\widehat{\Delta}_{j,[1:S]}$, $D^t$) ▷ Choose the best of $S$ continuations
      $\Delta_j$ = [$\Delta_j$, $\widehat{\Delta}_j$] ▷ Synthesize full sequence
      $p(bbox^t_{\mathcal{T}_j})$ = ArTIST($\Delta_j$, $I_j$) ▷ Compute the likelihood of next plausible bounding box
      **for** $d^t_i$ in $D^t$ **do** ▷ Compute the cost of assigning detections to $\mathcal{T}_j$
        Cost[$j$][ $i$] = NLL($p(bbox^t_{\mathcal{T}_j})$, $d^t_i$)

  assign$_f$, uD$_f$, uT$_f$ = `Munkres`(Cost, $D^t$, $\mathbb{T}$, FullSeq) ▷ Assignment for fully observed tracklets
  assign$_i$, uD$_i$, uT$_i$ = `Munkres`(Cost, uD$_f$, $\mathbb{T}$, InpaintSeq) ▷ Assignment for inpainted tracklets (given the unassigned detections)
  assignment = `Combine`(assign$_f$, assign$_i$) ▷ Total assignment
  **update**($\mathbb{T}$, assignment) ▷ Update the tracklets at time-step $t$

## 4. Effect of Multinomial Sampling

In order to better show the effect of multinomial sampling, we provide a more detailed view of the ablation study provided in Table 6 of the main paper, as that in Table 1. This makes it clearer that the impact of our Multi+TRS sampling is larger on moving cameras than on static ones, significantly improving identity-preservation metrics, e.g., IDF1 and IDs.

Table 1. Per-sequence comparison between Top-1 and Multi+TRS.

| Sequence | Camera | FPS | Setting | MOTA↑ | IDF1↑ | IDs↓ |
|---|---|---|---|---|---|---|
| MOT17-04 | Static | 30 | Top-1 | 70.2 | 73.2 | 55 |
| | | | Multi+TRS | **71.0** | **74.6** | **41** |
| MOT17-05 | Moving | 14 | Top-1 | 57.5 | 65.0 | 38 |
| | | | Multi+TRS | **59.3** | **69.6** | **25** |
| MOT17-11 | Moving | 30 | Top-1 | 64.9 | 61.1 | 17 |
| | | | Multi+TRS | **66.3** | **70.0** | **13** |

## 5. Sampling Quality

We also evaluate the quality of the generated sequences. To this end, we compare deterministic sampling (top-1) with our stochastic sampling at inference in Table 2, in which we compute the mIoU between the entire sequence of generated bounding boxes and the GT ones. For the stochastic case, we compute the best score over 30 samples. We performed this experiment on 1K tracklets of varying length and giving variable observation lengths to the model. As can be seen in Table 2, the relative improvement over the deterministic case increases as the observation length decreases.

Table 2. Evaluating sampling quality for future tracklet generation on 1000 random variable-length tracklets of the MOT17 val.

| Observation | Future | Deterministic | Stochastic | Relative Improvement |
|---|---|---|---|---|
| 75% | 25% | 75.3% | **83.7%** | 11.1% |
| 50% | 50% | 64.0% | **75.3%** | 17.7% |
| 25% | 75% | 54.4% | **68.7%** | 26.3% |

## 6. Evaluation Metrics

Several metrics are commonly used to evaluate the quality of a tracking system [6, 1]. The main one is MOTA, which combines quantification of three error sources: false positives, false negatives and identity switches. A higher MOTA score implies better performance. Another important metric is IDF1, i.e., the ratio of correctly identified detections over the average number of ground-truth and com-

puted detections. The number of identity switches, IDs, is also frequently reported. Furthermore, the following metrics provide finer details on the performance of a tracking system: mostly tracked (MT) and mostly lost (ML), that are respectively the ratio of ground-truth trajectories that are covered/lost by the tracker for at least 80% of their respective life span; False positives (FP) and false negatives (FN). All metrics were computed using the official evaluation code provided by the MOTChallenge benchmark.

## References


[1] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *Journal on Image and Video Processing*, 2008:1, 2008. 2

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[3] Santiago Manen, Michael Gygli, Dengxin Dai, and Luc Van Gool. Pathtrack: Fast trajectory annotation with path supervision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 290–299, 2017. 1

[4] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013. 1

[5] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 1

[6] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*, pages 17–35. Springer, 2016. 2

[7] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989. 1