Tracking Emerges by Looking Around Static Scenes, with Neural 3D Mapping – Supplementary Material

Adam W. Harley, Shrinidhi Kowshika Lakshmikanth,
Paul Schydlo, and Katerina Fragkiadaki

Carnegie Mellon University, Pittsburgh PA, 15213, USA
{aharley,kowshika,pschydlo,katef}@cs.cmu.edu

1 Implementation details

1.1 Neural 3D mapping

Our 3D convolutional network f has the following architecture. The input is a 4-channel 3D grid shaped $128\times32\times128$. The four channels are RGB and occupancy. "Unoccupied" voxels have zeros in all channels. This input is passed through a 3D convolutional network, which is an encoder-decoder with skip connections. There are three encoder layers: each have stride-2 3D convolutions, with kernel size $4 \times 4 \times 4$, and the output channel dimensions are $64, 128, 192$. There are three decoder layers: the first two have stride-2 transposed convolutions with stride 2 and $4 \times 4 \times 4$ kernels with 256 output channels; after each of these we concatenate the same-resolution layer from the encoder half; the last layer has a $1 \times 1 \times 1$ kernel and 64 channels.

At training time f encodes a metric space sized $32 \times 4 \times 32$ meters. At test time we "zoom in" on the area of interest (i.e., the search region), and encode a space sized $16 \times 4 \times 16$ meters.

We train with the Adam optimizer [5] and train for $200,000$ iterations at a learning rate of $1e-4$.

1.2 Baselines

– **Unsupervised 3D flow [3].** The original work by Harley et al. [3] only estimated flow for pairs of frames. We extend this into a 3D tracker by deploying the flow module in an object-centric manner, and by "chaining" the flows together across time. We effectively compute the flow from the first frame *directly* to every other frame in the sequence, but with an alignment step designed to keep the object within the field of view of the flow module. Specifically, given the object location in the first frame, we extract a volume of flow vectors in the object region, and run RANSAC to find a rigid transformation that explains the majority of the flow field. We apply this rigid transform to the box, yielding the object's location in the next frame. Then, we back-warp the next frame according to the box transformation, to re-center the voxel grid onto the object. Estimating the flow between the original frame and the newly backwarped frame yields the *residual* flow field; adding this residual to the original flow provides the new cumulative motion of the object. We repeat these steps across the length of the input video, back-warping according to the cumulative flow and estimating the residual.

- **2.5D dense object nets [2].** While the original work used a margin loss for contrastive learning, we update this to a cross entropy loss with a large offline dictionary and a coupled "slow" encoder, consistent with state-of-the-art contrastive learning [4]. We extend this model into a 3D tracker as follows. First, we use the available depth data to "unproject" the learned embeddings into sparse 3D scene maps. Then, we use the same tracking pipeline as our own model (with search regions and soft argmaxes and RANSAC). The differences between this model and our own are (1) it learns a 2D CNN instead of a 3D one, and (2) its 3D output is constrained to the locations observed in the depth map, instead of producing features densely across the 3D grid.
- **2.5D tracking by colorization [7,6].** We use the latest iteration of this method [6], which uses the LAB colorspace at input and output, and color dropout at training time; this outperforms the quantized-color cross entropy loss of the original work. We extend this model into a 3D tracker in the same way that we extended the "dense object nets" baseline. Therefore, the only difference between the two 2.5D methods is in the training: this method can train on arbitrary data but has an objective that only indirectly encourages correspondence (through an RGB reconstruction loss), while "2.5D dense object nets" requires static scenes and directly encourages correspondences (through a contrastive loss).
- **3D neural mapping with random features.** We use the same inputs, architecture, outputs, and the same hyperparameters for resolution and search regions, but do not train the parameters.
- **3D fully convolutional siamese tracker (supervised) [1].** We use the same resolutions and feature encoders for this model as we do for our main model. The cosine window adds a zero-velocity bias into the model, and makes it less likely for the model's outputs to make large jumps [1].

2 Detailed results

In Tables 1 and 2, we provide the numerical values of the data plotted in the main paper. The notation IOU@N denotes 3D intersection over union at the Nth frame of the video. Note that IOU@0 = 1.0 in this task, since tracking is initialized with a ground-truth box in the zeroth frame.

Table 1. Single-object tracking accuracy, in mean IOU across time, in CARLA.

Method	IOU@2	IOU@4	IOU@6	IOU@8
Zero motion	0.63	0.33	0.21	0.17
Random 3D neural mapping	0.13	0.07	0.05	0.03
Random 3D siamese	0.05	0.02	0.01	0.00
Random 3D siamese + cosine	0.48	0.22	0.10	0.05
2.5D colorization	0.41	0.29	0.25	0.19
2.5D dense object nets	0.66	0.47	0.39	0.33
Harley et al. [3]	0.58	0.47	0.38	0.29
Ours	0.80	0.69	0.66	0.61
3D siamese (supervised)	0.69	0.65	0.63	0.61
3D siamese + cosine (supervised)	0.75	0.71	0.69	0.65

Table 2. Single-object tracking accuracy, in mean IOU across time, in KITTI.

Method	IOU@2	IOU@4	IOU@6	IOU@8
Zero motion	0.59	0.28	0.18	0.13
Random 3D neural mapping	0.10	0.04	0.03	0.02
Random 3D siamese	0.00	0.01	0.00	0.00
Random 3D siamese + cosine	0.40	0.11	0.03	0.01
2.5D colorization	0.30	0.20	0.11	0.10
2.5D dense object nets	0.68	0.55	0.40	0.31
Harley et al. [3]	0.55	0.42	0.39	0.22
Ours	0.66	0.58	0.52	0.46
3D siamese (supervised)	0.61	0.60	0.58	0.58
3D siamese + cosine (supervised)	0.60	0.56	0.55	0.55

References

1. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: European conference on computer vision. pp. 850–865. Springer (2016)
2. Florence, P.R., Manuelli, L., Tedrake, R.: Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In: CoRL (2018)
3. Harley, A.W., Lakshmikanth, S.K., Li, F., Zhou, X., Tung, H.Y.F., Fragkiadaki, K.: Learning from unlabelled videos using contrastive predictive neural 3d mapping. In: ICLR (2020)
4. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: CVPR (2020)
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Lai, Z., Lu, E., Xie, W.: MAST: A memory-augmented self-supervised tracker. In: CVPR (2020)
7. Vondrick, C., Shrivastava, A., Fathi, A., Guadarrama, S., Murphy, K.: Tracking emerges by colorizing videos. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 391–408 (2018)