Supplementary File of Paper: "TubeTK: Adopting Tubes to Track Multi-Object in a One-Step Training Model"

Bo Pang, Yizhuo Li, Yifan Zhang, Muchen Li†, Cewu Lu*
Shanghai Jiao Tong University, †Huazhong University of Science and Technology
{pangbo, liyizhuo, zhangyf_sjtu lucewu}@sjtu.edu.cn, muchenli@alumni.hust.edu.cn

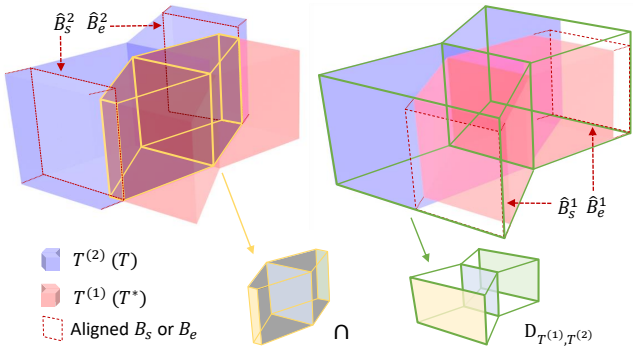

Figure 1. Visualization of the calculation process of Tube GIoU. The intersection and $D_{T^{(1)},T^{(2)}}$ of targets are also decahedrons, thus the volume of them can be calculated in the same way of Btubes.

1. Tube GIoU Algorithm

Tube GIoU is a 3D extension version of the original GIoU, from area to volume. To calculate it, we need to get the volume of the predicted and GT Btubes ($T^{(1)}$ and $T^{(2)}$), their intersection $\bigcap$, and their smallest enclosing tube D. Since Btubes are decahedrons, directly calculating their $\bigcap$ and D is difficult. But according to our regression method, B_m of T and $B_m^\star$ of $T^\star$ must be on the same video frame, which makes the calculating straightforward. As shown in Fig. 1, we can treat each Btube as two square frustums that share the same underside. Because B_m and $B_m^\star$ are on the same video frame, the intersection and enclosing convex object are also composed of two adjoining square frustums whose volumes are easy to calculate, as Alg. 1 shows.

2. Detailed Network Structure

The detailed network is shown in Fig. 2. We adopt the 3D-ResNet50 as the backbone and the temporal kernel size is the same with the spatial ones, i.e. if the original kernel size of 2D-ResNet is 3×3, then the 3D version's is 3×3×3.

*Corresponding author.

Algorithm 1 Tube IoU & Tube GIoU

Input: Btube $T^{(1)}$:
$\{B_s^{(1)}, B_m^{(1)}, B_e^{(1)}, d_s^{(1)}, d_e^{(1)}\}$
and the corresponding ground truth $T^{(2)}$:
$\{B_s^{(2)}, B_m^{(2)}, B_e^{(2)}, d_s^{(2)}, d_e^{(2)}\}$
Output: Tube IoU and GIoU between T and $T^{(2)}$

1: Aligning $B_s^{(2)}$ to the frame of $B_s^{(1)}$ as $\hat{B}_s^{(2)}$:
$\hat{B}_s^{(2)} = (B_m^{(2)} \times |d_s^{(2)} - d_s^{(1)}| + B_s^{(2)} \times |d_s^{(1)}|)/d_s^{(2)}$
Getting $\hat{B}_s^{(1)}$ by the same method.
2: Calculating the intersection $\mathcal{I}_s^a$ and the smallest enclosing box $\mathcal{E}_s^a$ between $B_s^{(1)}$ and $\hat{B}_s^{(2)}$, $\mathcal{I}_s^b$ and $\mathcal{E}_s^b$ between $\hat{B}_s^{(1)}$ and $B_s^{(2)}$, $\mathcal{I}_m$ and $\mathcal{E}_m$ between $B_m^{(1)}$ and $B_m^{(2)}$.
3: Calculating the tube intersection, volume and smallest enclosing tube. We assume $d_s^{(2)} \geq d_s^{(1)}$ and $d_e^{(2)} \geq d_e^{(1)}$:
$|\bigcap_s| = (|\mathcal{I}_s^a| + |\mathcal{I}_m| + \sqrt{|\mathcal{I}_s^a|} \times \sqrt{|\mathcal{I}_m|}) \times d_s^{(1)}$.
$V_{T^{(1)}}^s = (|B_s^{(1)}| + |B_m^{(1)}| + \sqrt{|B_s^{(1)}|} \times \sqrt{|B_m^{(1)}|}) \times d_s^{(1)}$.
$|D_{T^{(1)},T^{(2)}}^s| = (|\mathcal{E}_s^b| + |\mathcal{E}_m| + \sqrt{|\mathcal{E}_s^b|} \times \sqrt{|\mathcal{E}_m|}) \times d_s^{(2)}$.
In the same way, we can get $V_{T^{(2)}}^s$, $|\bigcap_e|$, $V_{T^{(1)}}^e$, $V_{T^{(2)}}^e$, and $|D_{T^{(1)},T^{(2)}}^e|$. And, $|D_{T^{(1)},T^{(2)}}| = |D_{T^{(1)},T^{(2)}}^s| + |D_{T^{(1)},T^{(2)}}^e|$
4: Calculating the Tube IoU:
$|\bigcup| = (V_{T^{(1)}}^s + V_{T^{(2)}}^s + V_{T^{(1)}}^e + V_{T^{(2)}}^e - |\bigcap_s| - |\bigcap_e|)$
$\text{TIoU} = (|\bigcap_s| + |\bigcap_e|)/|\bigcup|$
5: Calculating the Tube GIoU:
$\text{TGIoU} = \text{TIoU} - \frac{|D_{T^{(1)},T^{(2)}}| - |\bigcup|}{|D_{T^{(1)},T^{(2)}}|}$

We adopt the features from Layer-2, 3, 4 of the backbone as the FPN's inputs. And the FPN also has two external layers to enlarge the receptive fields. The task heads consist of 5 CNN layers and the number of hidden filters are 256.

3. Detailed Performance on MOT15, 16, 17

The detailed performances on every single video in MOT15, 16, 17 are shown in Tab. 1.

4. Usage of POI Detection Results

For our TubeTK does not need external detection results and we do not have a large dataset to train it, TubeTK does not achieve the SOTA results on MOT16 (2.1 MOTA gap).

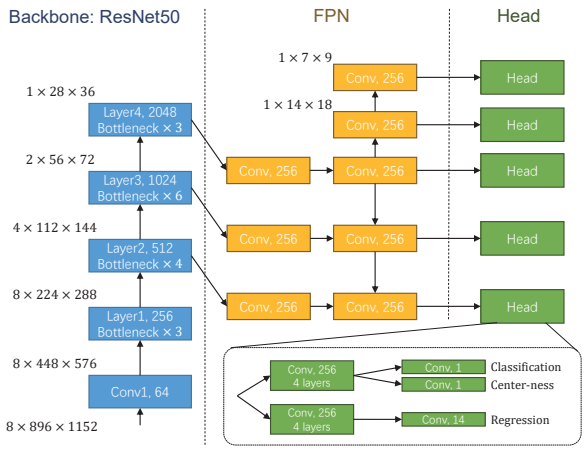

Figure 2. Network structure. All the layers have the 3D kernel and the temporal kernel size is the same with the spatial ones.

Table 1. Detailed results of every single video.

	Video	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓
MOT17	MOT17-01	47.9	44.9	6	10	167	3154	41
	MOT17-03	76.4	69.6	81	12	3181	21287	186
	MOT17-06	52.4	54.8	85	36	1609	3699	307
	MOT17-07	55.4	43.3	21	2	1944	5371	222
	MOT17-08	42.3	34.1	18	12	970	10889	319
	MOT17-12	50.3	49.4	28	23	494	3749	63
	MOT17-14	35.6	39.5	6	61	655	11012	241
MOT16 (w/o Dtr)	MOT16-01	48.9	45.5	8	9	175	3052	40
	MOT16-03	76.3	69.5	86	12	3741	20828	177
	MOT16-06	51.2	55.7	87	39	1863	3542	231
	MOT16-07	55.0	43.5	21	3	2225	4938	190
	MOT16-08	46.9	37.3	18	3	1694	6952	234
	MOT16-12	52.4	50.8	27	20	533	3366	51
	MOT16-14	35.8	39.8	7	61	731	10948	194
MOT16 (with POI)	MOT16-01	64.9	55.7	12	2	357	1839	49
	MOT16-03	78.8	72.3	101	12	3907	18163	148
	MOT16-06	53.7	57.5	96	35	1795	3275	268
	MOT16-07	59.4	46.1	22	2	1752	4693	182
	MOT16-08	47.7	39.1	19	3	1507	7040	207
	MOT16-12	54.8	57.1	32	19	490	3208	51
	MOT16-14	38.6	42.1	14	49	1736	9284	331
MOT15	TUD-Crossing	80.0	59.4	10	0	69	129	22
	PETS09-S2L2	69.0	40.7	21	1	759	1996	230
	ETH-Jelmoli	60.3	63.9	14	15	152	826	29
	ETH-Linthescher	66.5	63.0	68	61	417	2467	109
	ETH-Crossing	66.3	60.3	9	9	37	294	7
	AVG-TownCentre	67.0	70.8	105	26	670	1550	140
	ADL-Rundle-1	42.5	44.2	10	2	1632	3630	88
	ADL-Rundle-3	49.9	44.6	17	2	926	4067	96
	KITTI-16	61.3	66.4	8	1	150	486	23
	KITTI-19	47.8	53.8	13	10	595	2118	78
	Venice-1	61.0	54.2	8	3	349	1398	32

But this kind of comparison is not fair because the SOTA methods all adopt the POI detection* which is generated by the Faster-RCNN detector trained on a pretty large dataset.

For a fair comparison, although our method is not a track-by-detection (TBD) model, we propose a mechanism to use the POI detection results. Originally, TubeTK first generates a set of Btubes, then we link them with a greedy algorithm. To adopt POI, we first expend all the detection Bboxes in POI to short Btubes, then add these new Btubes to the set, and finally link all of them. Note that the expending method is just simply copying the Bbox's spatial position to the adjacent frames to form a straight Btube, thus, the expended Btubes do not contain any moving trend information like the predicted ones. This is really a weak method to use the external detection results, but what we want is to show the potential of our method in a relatively fair comparison.

*https://drive.google.com/open?id=0B5ACiy41McAHMjczS2p0dFg3emM