Related Work

Lucas-Kanade Method

$$I(x+\delta x, y+\delta y, z+\delta z, t+\delta t) = I(x,y,z,t)+\frac{\partial I}{\partial x}\delta x+\frac{\partial I}{\partial y}\delta y+\frac{\partial I}{\partial z}\delta z+\frac{\partial I}{\partial t}\delta t+H.O.T.$$

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial z}\delta z + \frac{\partial I}{\partial t}\delta t = 0$$

$$\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} \sum I_{x_i}^2 & \sum I_{x_i}I_{y_i} & \sum I_{x_i}I_{z_i} \\ \sum I_{x_i}I_{y_i} & \sum I_{y_i}^2 & \sum I_{y_i}I_{z_i} \\ \sum I_{x_i}I_{z_i} & \sum I_{y_i}I_{z_i} & \sum I_{z_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_{x_i}I_{t_i} \\ -\sum I_{y_i}I_{t_i} \\ -\sum I_{z_i}I_{t_i} \end{bmatrix}$$

Implementing Details

Markov Decision Process (MDP)

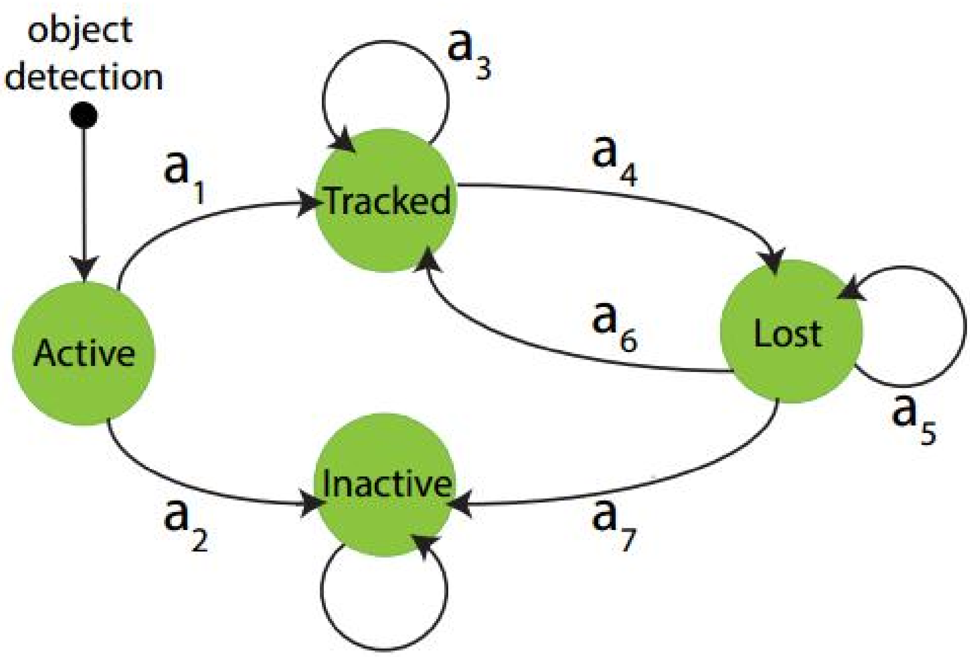


Figure 2. The target MDP in our framework.

Policy in an Active State

This decision making can be considered to be a preprocessing step before tracking. Strategies such as non-maximum suppression or thresholding detection scores are usually used. We train a SVM(offline):

$$R_{\text{Active}}(s,a) = y(a)\big(\mathbf{w}_{\text{Active}}^T \phi_{\text{Active}}(s) + b_{\text{Active}}\big),$$

Note that a false alarm from object detector can still be miss-classified and transfered to a tracked state, which will be handled by the MDP in the tracked and lost states.

Policy in a Tracked State

Whenever an object detection is transferred to a tracked target, we initialize the target template with the detection bounding box. Also, the MDP collects target's templates in the tracked frames to represent the history of the target.

In order to use the target template for tracking, we compute an optical flow from densely and uniformly sampled points inside the template to a new video frame.

For position u, we find its corresponding location v, then compute the backward flow of point v to the target template and obtain a new prediction u’.

$$e(\mathbf{u}) = \|\mathbf{u} - \mathbf{u}'\|^2$$

$e_{\text{medFB}} = \text{median}(\{e(\mathbf{u}_i)\}_{i=1}^{n})$ n is the number of points

If emedFB is larger than some threshold, the tracking is considered to be unstable.

the bounding box overlap $o(t_k, \mathcal{D}_k)$

$$o_{\text{mean}} = \text{mean}(\{o(t_k, \mathcal{D}_k)\}_{k=1}^{K})$$

$$R_{\text{Tracked}}(s, a) = \begin{cases} y(a), & \text{if } e_{\text{medFB}} < e_0 \text{ and } o_{\text{mean}} > o_0 \\ -y(a), & \text{otherwise,} \end{cases} \tag{2}$$

where e_0 and o_0 are specified thresholds, $y(a) = +1$ if action $a = a_3$, and $y(a) = -1$ if $a = a_4$ in Fig. 2. So the MDP keeps the target in a tracked state if e_{medFB} is smaller but o_{mean} is larger than certain thresholds respectively. Otherwise, the target is transfered to a lost state.

Template Updating

Online tracking methods update the appearance model whenever the tracker tracks the target. As a result, they are likely to accumulate tracking errors during the update, and drift from the target.

To solve this,, the template used in tracking remains unchanged if it is able to track the target. Whenever the template fails to track the target due to appearance change, the MDP transfers the target into a lost state. We store K templates as the history of the target being tracked. These K templates are used for data association in lost states. So we do not accumulate tracking errors, but reply on the data association to handle the appearance change and continue the tracking.

Policy in a Lost State

1. We simply mark a lost target as inactive and terminate the tracking if the target has been lost for more than TLost frames.
2. Data Association.

$$R_{\text{Lost}}(s,a) = y(a)\left(\max_{k=1}^{M} \left(\mathbf{w}^T \phi(t, d_k) + b \right) \right),$$

Reinforcement Learning

1. When the MDP associates the target to an object detection which is wrong according to the ground truth, then the target and the detection is added to the training set S of the binary classifier as a negative example.
2. When the MDP decides to not associate the target to any detection, but the target is visible and correctly detected by

a detection according to the ground truth, then the target and the detection is added to the training set as a positive example.

3.

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{k=1}^{M}\xi_k$$

$$\text{s.t.} \quad y_k\left(\mathbf{w}^T\phi(t_k, d_k) + b\right) \geq 1 - \xi_k, \xi_k \geq 0, \forall k$$

3. So we employ a Reinforcement Learning to obtain a max-margin classifier for data association

Feature Representation

Type	Notation	Feature Description
FB error	$\phi_1, \cdots, \phi_5$	Mean of the median forward-backward errors from the entire, left half, right half, upper half and lower half of the templates in optical flow
NCC	ϕ_6	Mean of the median Normalized Correlation Coefficients (NCC) between image patches around the matched points in optical flow
	ϕ_7	Mean of the NCC between image patches of the detection and the predicted bounding boxes from optical flow
Height ratio	ϕ_8	Mean of the ratios in bounding box height between the detection and the predicted bounding boxes from optical flow
	ϕ_9	Ratio in bounding box height between the target and the detection
Overlap	ϕ_{10}	Mean of the bounding box overlaps between the detection and the predicted bounding boxes from optical flow
Score	ϕ_{11}	Normalized detection score
Distance	ϕ_{12}	Euclidean distance between the centers of the target and the detection after motion prediction of the target with a linear velocity model

Table 1. Our feature representation for data association.

Others

The similarity scores are used in the Hungarian algorithm to obtain the assignment between detections and lost targets.

Results

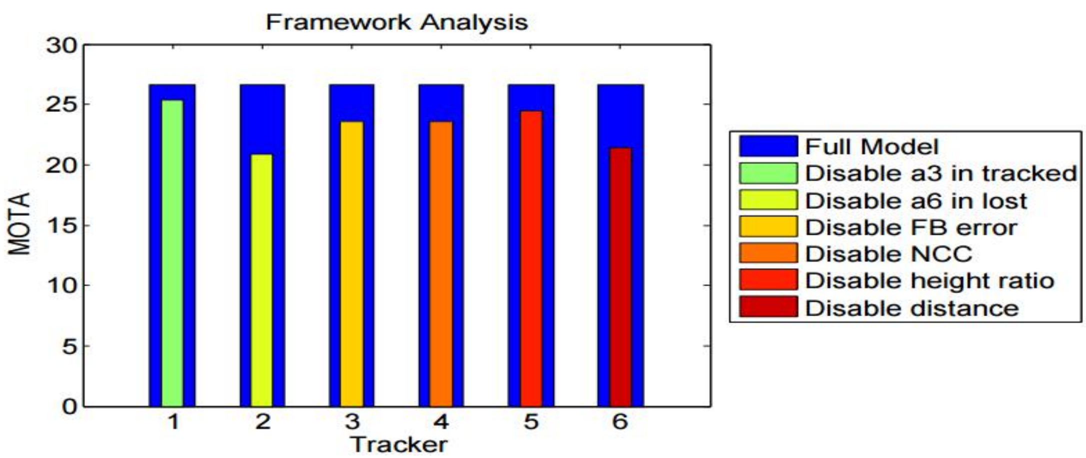
1.

K	MOTA	MOTP	MT	ML	FP	FN	IDS	Frag
1	24.7	73.2	10.3	55.1	3,597	13,651	147	303
2	25.7	73.5	9.8	53.4	3,548	13,485	121	349
3	23.0	73.6	8.5	56.0	3,727	13,907	134	325
4	26.3	**73.9**	9.8	53.8	3,191	13,726	**91**	300
5	**26.7**	73.7	**12.0**	53.0	3,386	**13,415**	111	331
6	19.5	73.7	5.6	68.8	3,393	14,920	269	321
7	26.1	73.6	10.7	55.6	3,092	13,838	132	306
8	25.8	73.8	10.7	55.6	3,221	13,785	122	305
9	**26.7**	73.6	**12.0**	**51.7**	3,290	13,491	133	328
10	26.6	73.8	9.8	55.1	**2,691**	14,130	123	**276**
11	25.3	73.5	**12.0**	52.1	3,672	13,436	136	317
12	24.8	73.4	11.5	55.6	3,637	13,585	139	321

Table 3. Tracking performance in terms of the number of templates on the validation set.

We observe two peaks for the tracking performance. One is around using 5 templates, and the other is around using 9 templates.

2.



3.

Tracker	Tracking Mode	Learning Mode	MOTA	MOTP	MT	ML	FP	FN	IDS	Frag	Hz
DP_NMS [36]	Batch	N/A	14.5	70.8	6.0%	40.8%	13,171	34,814	4,537	3,090	**444.8**
TC_ODAL [4]	Online	Online	15.1	70.5	3.2%	55.8%	12,970	38,538	**637**	1,716	1.7
TBD [14]	Batch	Offline	15.9	70.9	6.4%	47.9%	14,943	34,777	1,939	1,963	0.7
SMOT [12]	Batch	N/A	18.2	71.2	2.8%	54.8%	8,780	40,310	1,148	2,132	2.7
RMOT [42]	Online	N/A	18.6	69.6	5.3%	53.3%	12,473	36,835	684	1,282	7.9
CEM [27]	Batch	N/A	19.3	70.7	8.5%	46.5%	14,180	34,591	813	1,023	1.1
SegTrack [26]	Batch	Offline	22.5	**71.7**	5.8%	63.9%	**7,890**	39,020	697	**737**	0.2
MotiCon [23]	Batch	Offline	23.1	70.9	4.7%	52.0%	10,404	35,844	1,018	1,061	1.4
MDP OFL (Ours)	Online	Offline	30.1	71.6	10.4%	41.3%	8,789	33,479	690	1,301	0.8
MDP REL (Ours)	Online	Online	**30.3**	71.3	**13.0%**	**38.4%**	9,717	**32,422**	680	1,500	1.1

Table 4. Tracking performance on the test set of the MOT Benchmark. More comparisons are available at [2].

Some Ideas

1. When the k(the number of history target) increase, the tracking performance fluctuate. So I think the performance should improve as the k increase. Maybe this indicates that our model is underfit.

2. The policy in an Active State predict y(a) only according to a normalized 5D feature (2D coordinates , width , height, confidence) using a SVM. I recommend a CNN training offline taking the whole patch of the image of the bounding box as the input and serve as the binary classifier.

3. In the Reinforcement Learning part, we manually picked the

feature.Also I thing we can employ a CNN to do the binary classify instead of the hand-pick features so we can choose features automatically and fine-tune the CNN according to the validation set.