Parallel Tracking and Verifying: A Framework for Real-Time and High Accuracy Visual Tracking

Heng Fan and Haibin Ling

Project & code: http://www.dabi.temple.edu/~hbling/code/PTAV/ptav.htm

Problem

Goal: to locate an arbitrary target in a video with its initial position.

- *Model-free:* agnostic to the object's class
- *Single-object tracking*

Main challenges:

- *Appearance variations:* occlusion, scale changes, rotation, deformation, illumination variations, …
- *Real-time requirement*

Background

To deal with *appearance variations*:

- *Deep learning based solution:*
  - *Pros: robust to appearance variations*
  - *Cons: high computation burden*

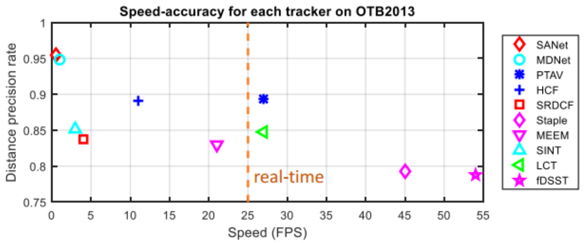
Representatives: SANet (CVPRW'17), MDNet (CVPR'16), HCF (ICCV'15), SINT (CVPR'16), C-COT (ECCV'16), …

To meet *real-time requirement*:

- *Using simple hand-crafted features*
  - *Pros: efficient computation, easily running real-time*
  - *Cons: sensitive to appearance variations*

Representatives: KCF (TPAMI'15), MOSSE (CVPR'10), fDSST(TPAMI'17), Staple (CVPR'16), …

Summary



- *Real-time & high quality trackers remain scarce*

Motivations

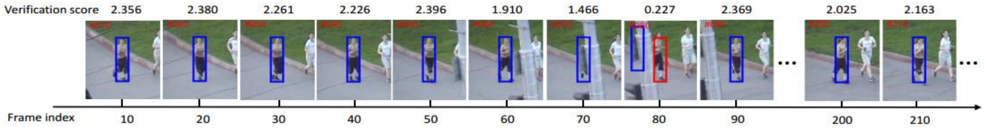
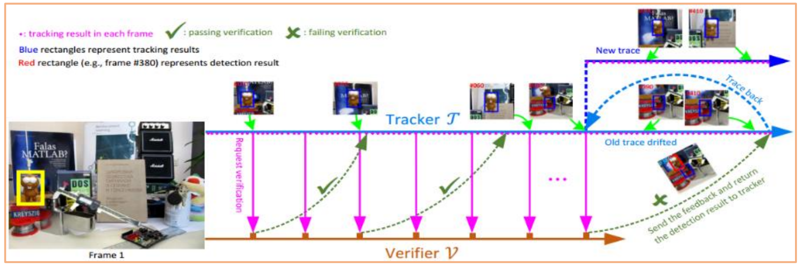


Figure: Verifying tracking results on a typical sequence. In most time, tracker works well.

- In most time, the target moves smoothly and its appearance changes slowly, and simple but efficient trackers work fine (see the above figure).
- Multi-thread computing has benefited computer vision systems, with notably in visual SLAM (*simultaneous localization and mapping*). By splitting tracking and mapping into two parallel threads, PTAV (*parallel tracking and mapping*) provides one of the most popular SLAM frameworks.
- Analogous to PTAM in which mapping is not required for every frame; nor does verifying in our task

Parallel tracking and verifying (PTAV): a general tracking framework



- Main idea
  - *Decompose visual tracking into two tasks, i.e., fast tracking and reliable verifying, processed by two parallel threads with necessary interactions (see above figure)*
- Components in PTAV
  - *A (fast) tracker $\mathcal{T}$*
    - *Perform efficiently (real-time)*
    - *Send verification request to $\mathcal{V}$*
    - *Allowed to make mistakes*
    - *Response to feedback from $\mathcal{V}$ by adjusting tracking model*
    - *Remain all intermediate states for fast rolling back*
  - *A (reliable) verifier $\mathcal{V}$*
    - *Perform relatively slowly but accurately*
    - *Receive request from $\mathcal{T}$*
    - *Return feedback to $\mathcal{T}$*, and correct it (if necessary)

PTAV: detailed workflow

Algorithm 1: Parallel Tracking and Verifying (PTAV)

```
Initialize the tracking thread for tracker T;
Initialize the verifying thread for verifier V;
Run T (Alg. 2) and V (Alg. 3) till the end of tracking;
```

Algorithm 3: Verifying Thread $\mathcal{V}$

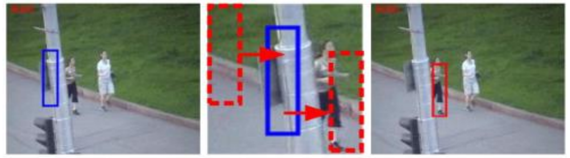
```
while not ended do
    if received request from T then
        Verifying the tracking result;
        if verification failed then
            Provide correction information;
        end
        Send verification result s to T;
    end
end
```

Algorithm 2: Tracking Thread $\mathcal{T}$

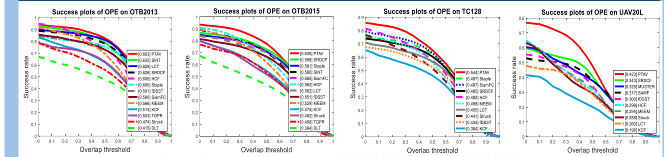
```
while Current frame is valid do
    if received a message s from V then
        if verification passed then
            Update tracking model (optional);
        else
            Correct tracking;
            Trace back and reset current frame;
            Resume tracking;
        end
    else
        Tracking on the current frame;
        if time for verification then
            Send the current result to V to verify;
        end
    end
    Current frame ← next frame;
end
```

PTAV: implementation

- Implementation of tracker $\mathcal{T}$
  - *Correlation filter based tracker: fDSST (Danelljan et al. PMAI'17)*
  - *Store all intermediate status*
- Implementation of verifier $\mathcal{V}$
  - *Siamese networks: SINT (CVPR'16)*
- How to correct $\mathcal{T}$
  - *$\mathcal{V}$ performs sliding window detection*



Experimental results



Speed comparisons on OTB2013 and OTB2015

			Deep trackers				Correlation-filters based trackers					Representative trackers		
		PTAV	HCF	SINT	DLT	SiamFC	SRDCF	Staple	LCT	fDSST	KCF	MEEM	TGPR	Struck
OTB2013	*speed (fps)*	27	11	3	9	46	4	45	27	54	245	21	1	10
OTB2015	*speed (fps)*	25	10	2	8	43	4	43	25	51	243	21	1	10

Ablation study of PTAV

- Different verification interval V

	$V=5$	$V=10$	$V=15$
DPR (%)	89.7	89.4	87.9
Speed (*fps*)	23	27	29

- Two threads V.S. one

Threads	OTB2013	OTB2015	TC128	UAV20L
One	16	14	11	15
Two	27	25	21	25

- Different tracker $\mathcal{T}$: fDSST V.S. KCF

		PTAV with fDSST	PTAV with KCF
OTB2013	DP (%)	89.4	80.4
	OS (%)	82.7	66.3
	Speed (*fps*)	27	24
OTB2015	DP (%)	84.9	73.5
	OS (%)	77.6	57.9
	Speed (*fps*)	25	21

Conclusion

- Decompose tracking into two separate tasks (i.e., fast tracking and slow verifying)
- The (fast) tracking (slow) verifying work asynchronously
- PTAV enjoys both high efficiency provided by $\mathcal{T}$ and the strong discriminative power provided by $\mathcal{V}$
- PTAV is a very flexible framework with great rooms for improvement and generalization.

[1] M. Danelljan, G. Hager, F. Khan, and M. Felsberg, Discriminative scale space tracking, *TPAMI*, 2017.
[2] R. Tao, E. Gavves, A. Smeulders, Siamese instance search for tracking, in *CVPR*, 2016.