Fast Visual Object Tracking using Ellipse Fitting for Rotated Bounding Boxes

York University, Toronto, Canada

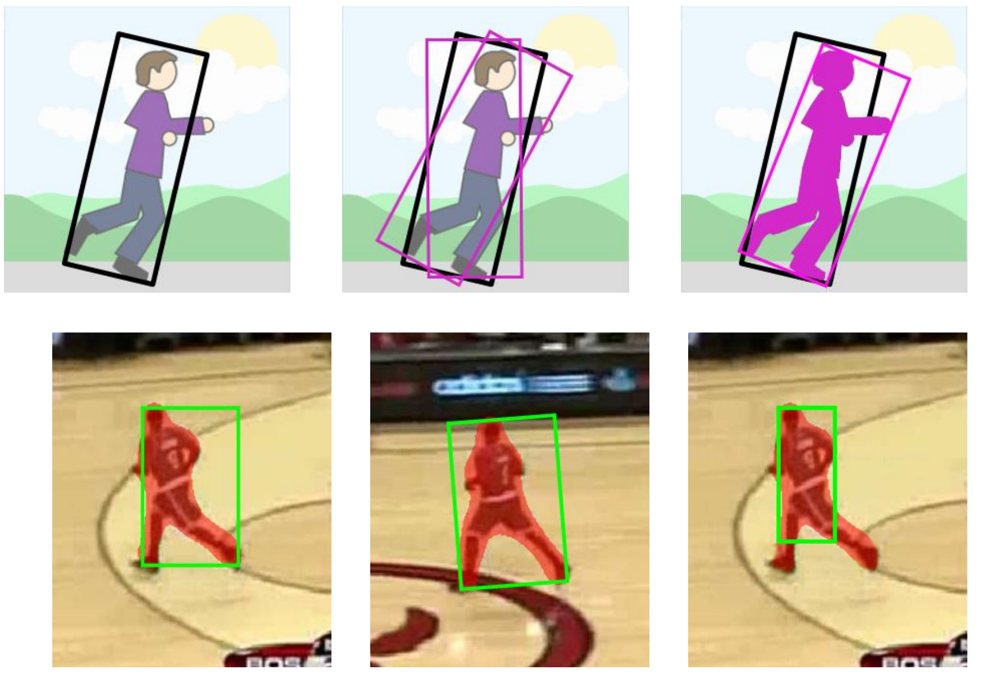
Bao Xin Chen, and John K. Tsotsos

YORK UNIVERSITÉ UNIVERSITY · Centre for Vision Research · Tsotsos Lab Active and Attentive Vision

Introduction

- ❖ An algorithm that uses **ellipse fitting** to estimate the bounding box rotation angle and size
- ❖ ~ **2% EAO** improvement on VOT2016, VOT2018, and VOT2019
- ❖ Retains a fast-tracking frame rate (**80 fps**) with GPU
- ❖ the source code[1] is available as an additional package to PySOT[2]

Motivation



- ❖ Rotated bounding boxes started in VOT2014
- ❖ Using Magic Numbers (e.g., ±10°, $\pm\pi/8$)
- ❖ Fitting bounding boxes on a Mask
  - Axis-aligned (very fast)
  - Min area bounding box (also fast)
  - VOT2016 automatic bounding box (5 fps)

Our approach

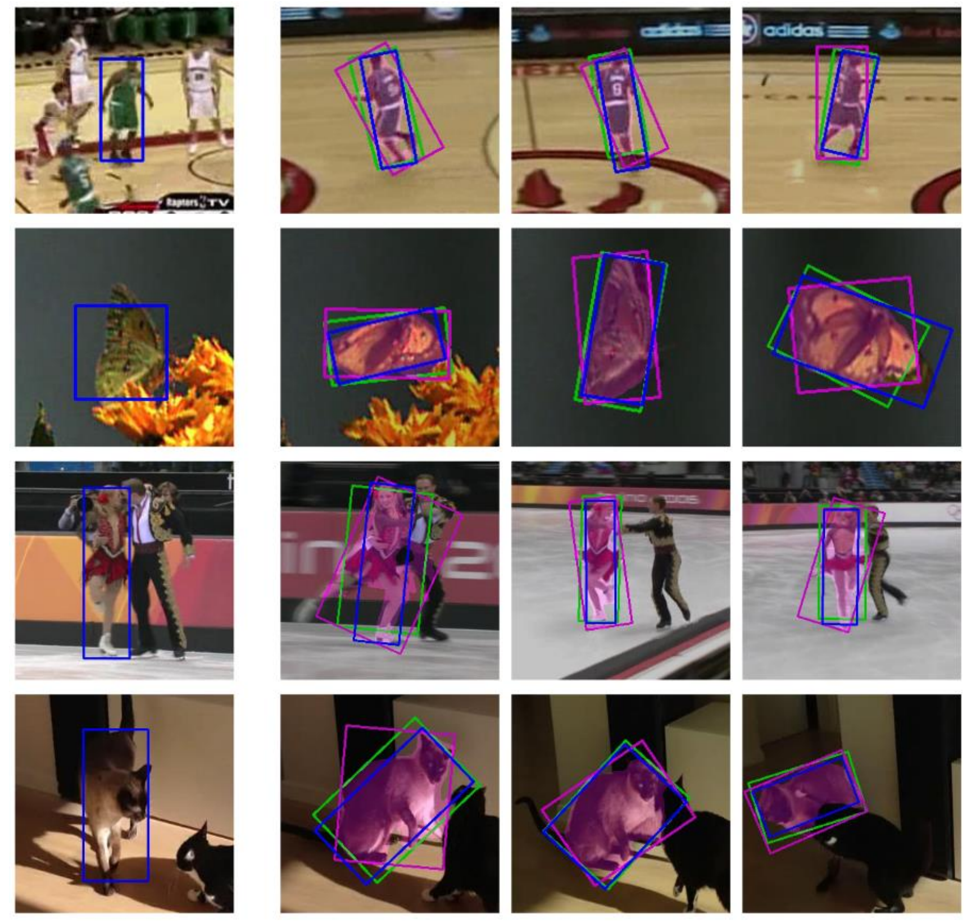
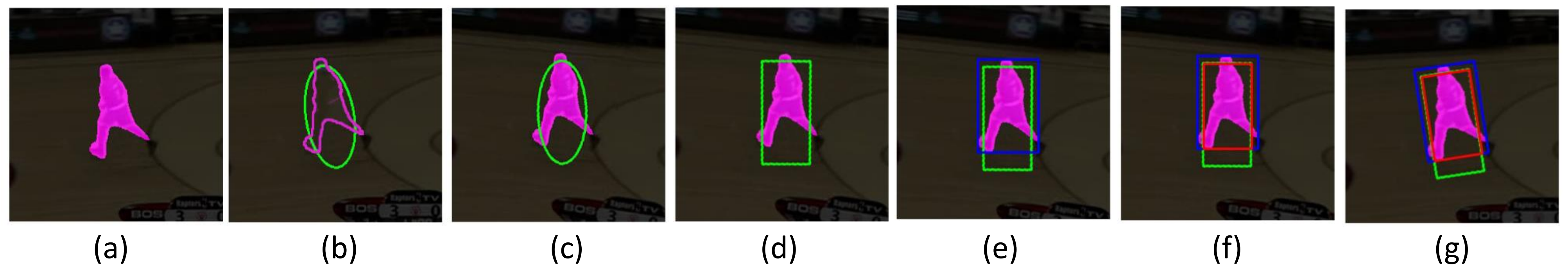


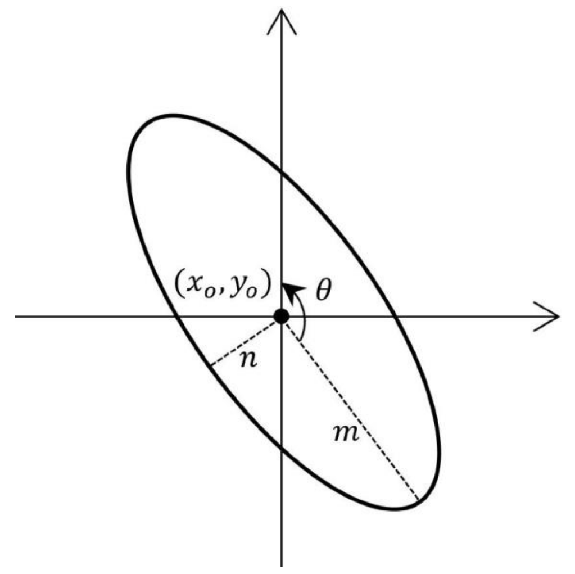
Figure 1. Our approach **SiamMask_E** yields lager IoU between the ground truth (blue) and its prediction (green) than the original SiamMask (magenta). SiamMask_E predicts a higher accuracy on the orientation of the bounding boxes which improves the average overlap accuracy (A) and expected average overlap (EAO).

Our approach (7 steps)



a) Take a target mask as input.

b) Apply an ellipse fitting algorithm B2AC on edge of the mask, then determine the center of the ellipse and the rotation angle.



- Ellipse formula

$$F(x,y) = ax^2 + bxy + cy^2 + dx + ey + f = 0$$

$$\text{where, } b^2 - 4ac < 0$$

- Giving a set of points minimize

$$\min_{\mathbf{a}} \sum_{i=1}^{N} F(x_i, y_i)^2$$

c) Compute the affine transformation matrix using the rotation angle and the center from the ellipse, then apply the transformation on the ellipse center.

$$M = \begin{bmatrix} cos\Theta & sin\Theta & (1-cos\Theta)x_o - sin\Theta y_o \\ -sin\Theta & cos\Theta & sin\Theta x_o - (1-cos\Theta)y_o \end{bmatrix}$$

d) Apply a rectangular rotated bounding box (green) on the ellipse.

e) Draw a min-max axis-aligned bounding box (blue) on the transformed mask.

f) Calculate the intersection of the blue box and green box to form a new bounding box (red).

g) Calculate the inverse of the affine transformation matrix, then apply transformation to convert back to the original image coordinate and output the red box.

$$M^{-1} = \begin{bmatrix} cos\Theta & sin\Theta & (1-cos\Theta)x_o - sin\Theta y_o \\ -sin\Theta & cos\Theta & sin\Theta x_o - (1-cos\Theta)y_o \end{bmatrix}^{-1}$$

[1] https://github.com/baoxinchen/siammask_e

[2] https://github.com/STVIR/pysot

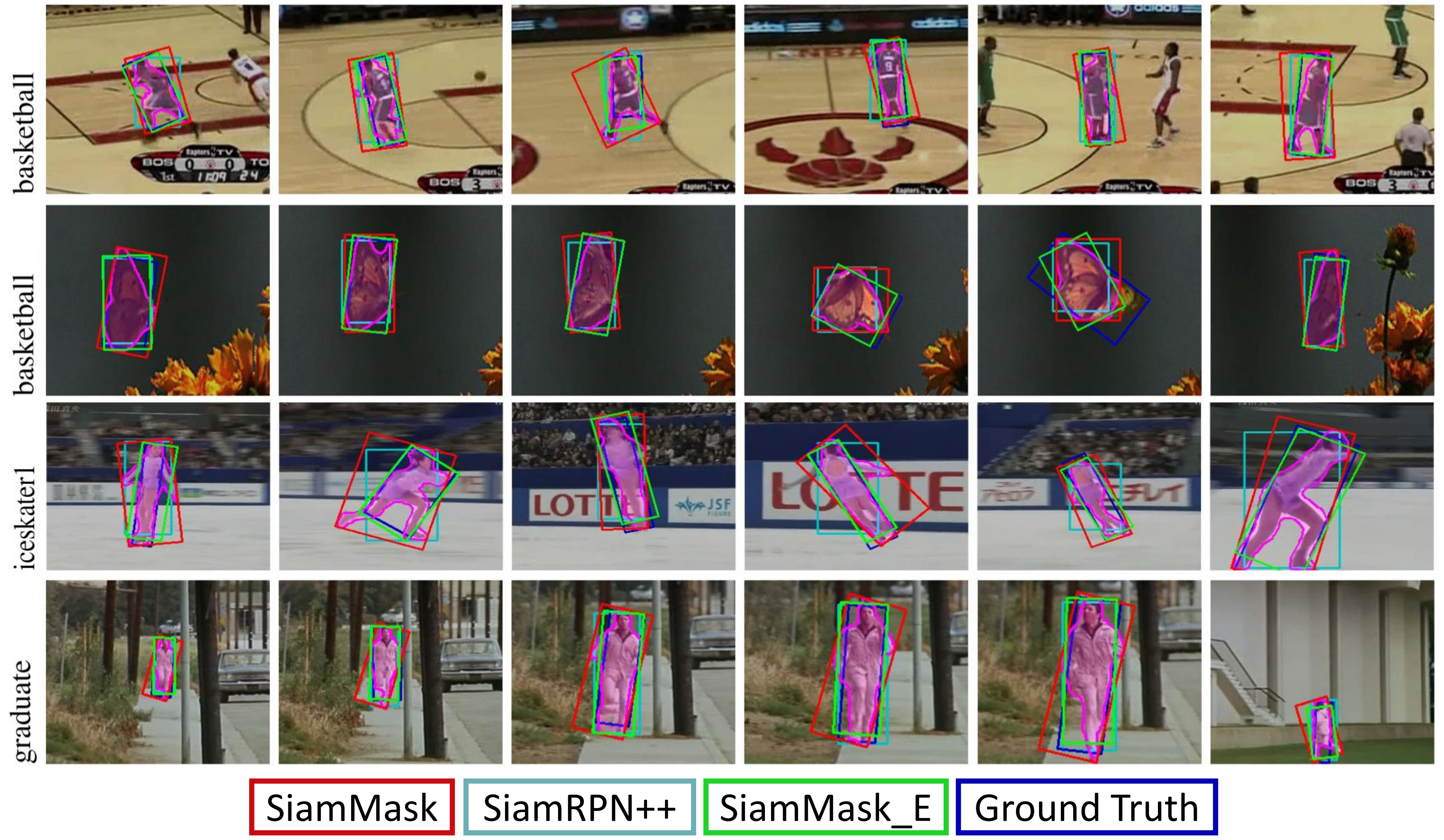
Code

Paper

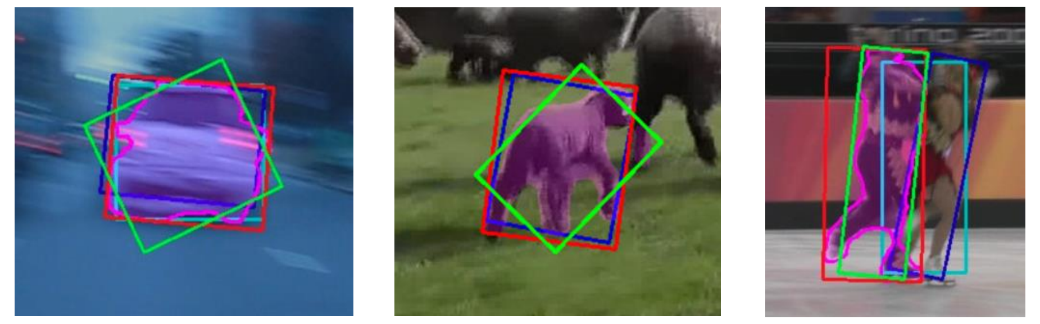
Results

	VOT2019			VOT2018			VOT2016			
	A↑	R↓	EAO↑	A↑	R↓	EAO↑	A↑	R↓	EAO↑	Speed↑
SiamRPN++	0.595	**0.467**	0.287	0.601	**0.234**	0.415	0.642	**0.196**	0.464	46 fps
SiamMask	0.596	**0.467**	0.283	0.598	0.248	0.406	0.621	0.214	0.436	87 fps
SiamMask-Opt*	-	-	-	0.642	0.295	0.387	0.670	0.233	0.442	5 fps
SiamMask_E (Ours)	0.625	0.482	0.298	0.627	0.248	0.427	0.645	0.210	0.452	85 fps
SiamMask_E_Ref (Ours)	**0.652**	0.487	**0.309**	**0.655**	0.253	**0.446**	**0.677**	0.224	**0.466**	80 fps

Table 1. Comparing with the state-of-the-art Siamese trackers on VOT2019, VOT2018, and VOT2016. Our tracker SiamMask_E with Ref outperforms other trackers in terms of average overlap accuracy (A) and expected average overlap (EAO). ↑ stands for the higher the best, and ↓ stands for the lower the best. * the numbers are reported in the original paper.



Failure cases



- ❖ Rotational symmetry shapes
- ❖ Some shapes are difficult to determine
- ❖ Adapts error from Siamese approach