

Graph Convolutional Tracking (GCT)

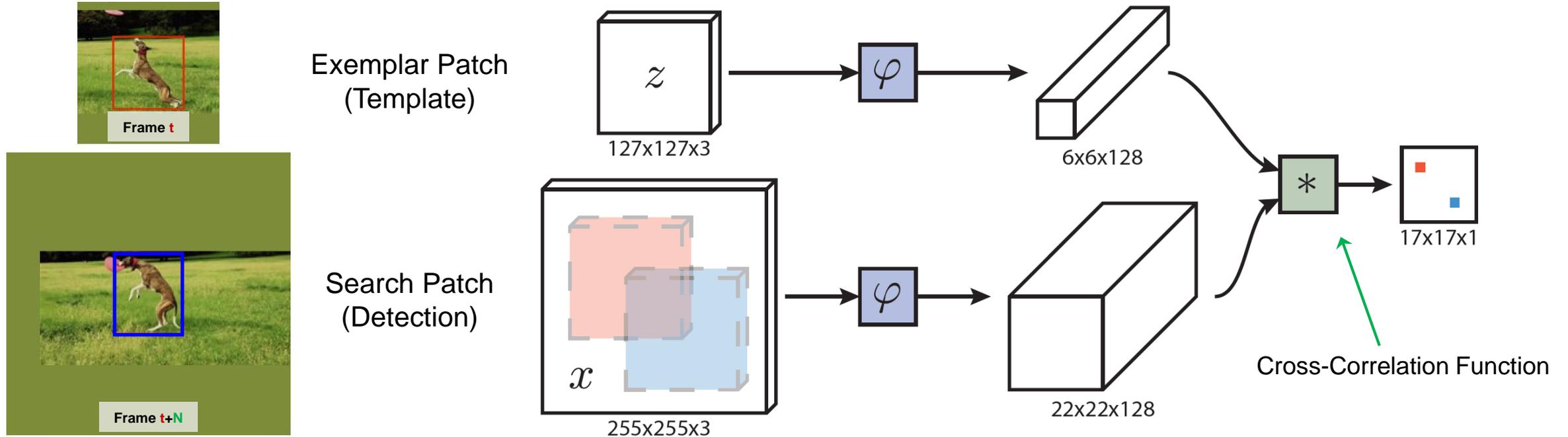
Kyuewang Lee

(2017-28945)

Dept. of ECE, SNU

Siamese Tracking (SiamFC)

- **SiamFC** : *Fully-Convolutional Siamese Networks for Object Tracking (ECCVw 2016)*



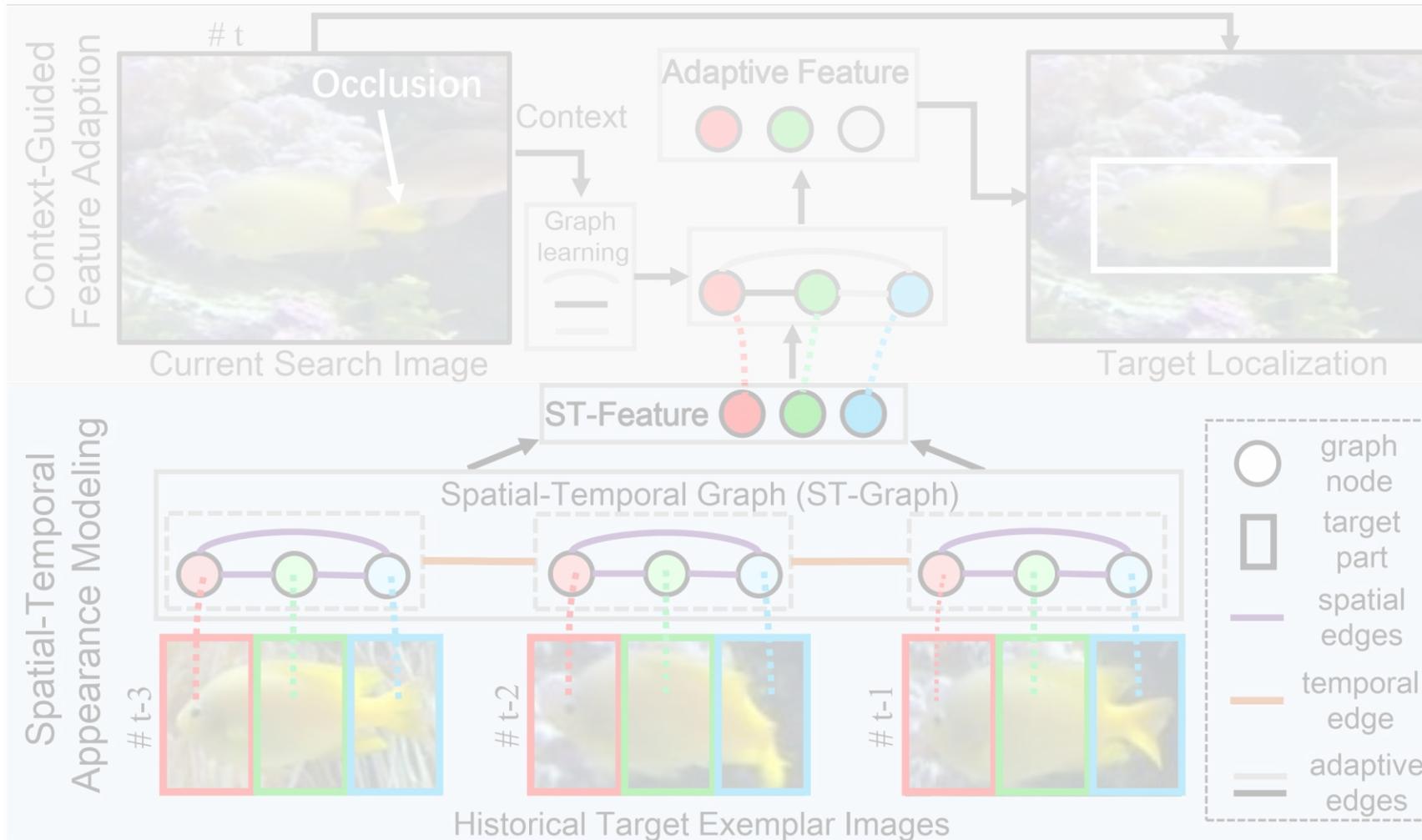
$$f(z, x) = g(\varphi(z), \varphi(x)) = \varphi(z) * \varphi(x) + \underbrace{b \cdot \vec{1}}_{\text{bias term}}$$

Graph Convolutional Tracking (GCT) Contributions

- The first work of an end-to-end Graph Convolutional Tracking algorithm
- Spatial-Temporal GCN (**ST-GCN**) and ConText GCN (**CT-GCN**)
 - ✓ ST-GCN learns *spatial-temporal feature* of the target, of previous frames
 - ✓ CT-GCN learns *context feature* of the target, in the current frame
- Performs favorably against **state-of-the-art trackers**, also runs in **real-time (50fps)**

Proposed Framework (GCT) : ST-GCN & CT-GCN

- **GCT** : *Graph Convolutional Tracking (CVPR 2019)*



Proposed Framework (GCT) : Formulation

- Siamese Tracking (**SiamFC** algorithm)

$$\begin{aligned} f(z, x) &= \phi(z) \star \phi(x) + b \\ &= \mathbf{Z} \star \mathbf{X} + b, \end{aligned}$$

- Graph Convolutional Tracking Formulation

$$f(z_{t-T:t-1}, x_t) = \psi_{GCN}(\mathbf{Z}_{t-T:t-1}, \mathbf{X}_t) \star \mathbf{X}_t + b,$$

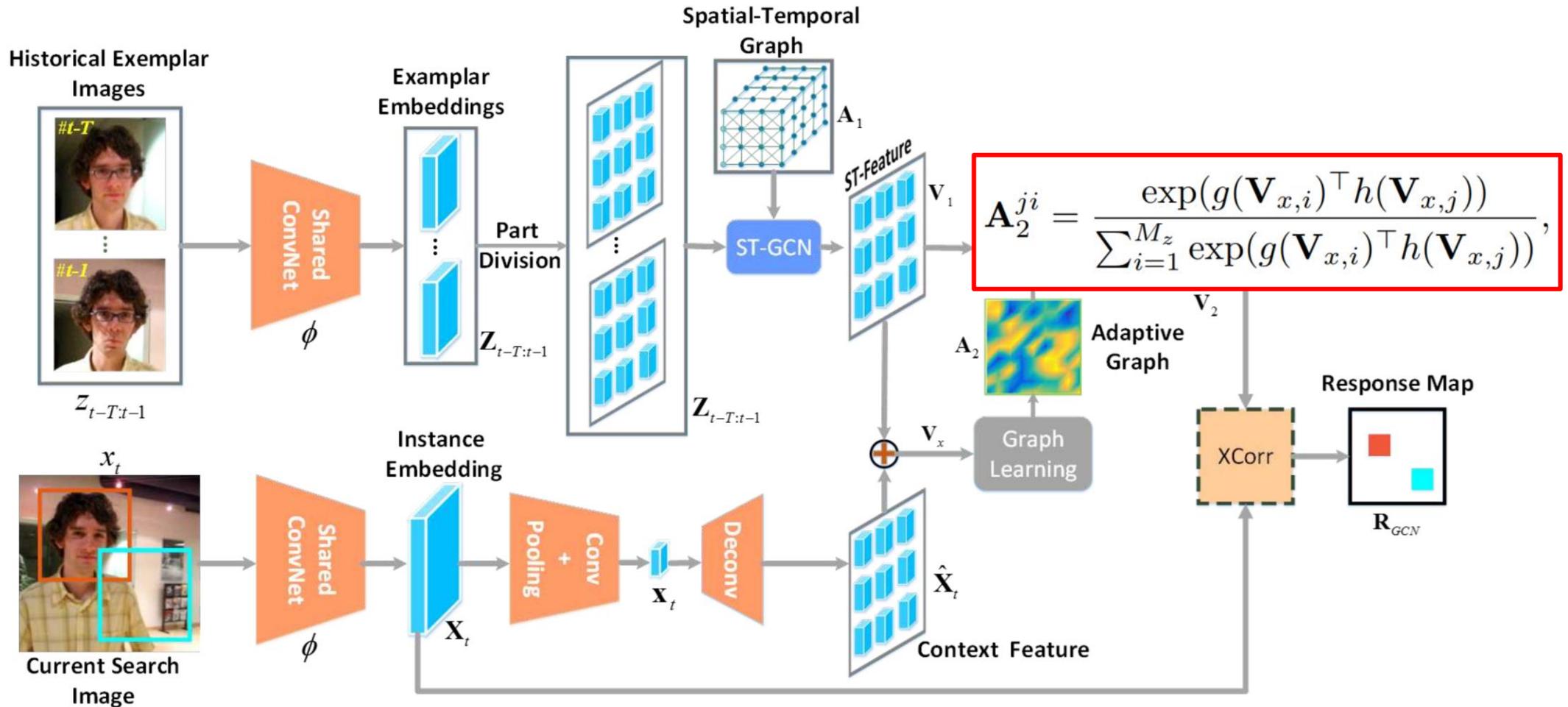
- Proposed Siamese Tracking Formulation (**GCT** algorithm)

$$f(z_{t-T:t-1}, x_t) = \psi_2(\psi_1(\mathbf{Z}_{t-T:t-1}), \mathbf{X}_t) \star \mathbf{X}_t + b,$$

$$\text{where } \begin{cases} \psi_1 & \text{Spatial-Temporal GCN (**ST-GCN**)} \\ \psi_2 & \text{ConText GCN (**CT-GCN**)} \end{cases}$$

Proposed Framework (GCT) : SiamFC-like Structure

- **GCT** : *Graph Convolutional Tracking (CVPR 2019)*



Experiment Settings and Datasets

- HW / SW

- ✓ Intel E5-2687 3.0GHz (CPU)
- ✓ 256GB RAM
- ✓ GeForce NVIDIA 1080Ti (GPU)
- ✓ Python + TensorFlow

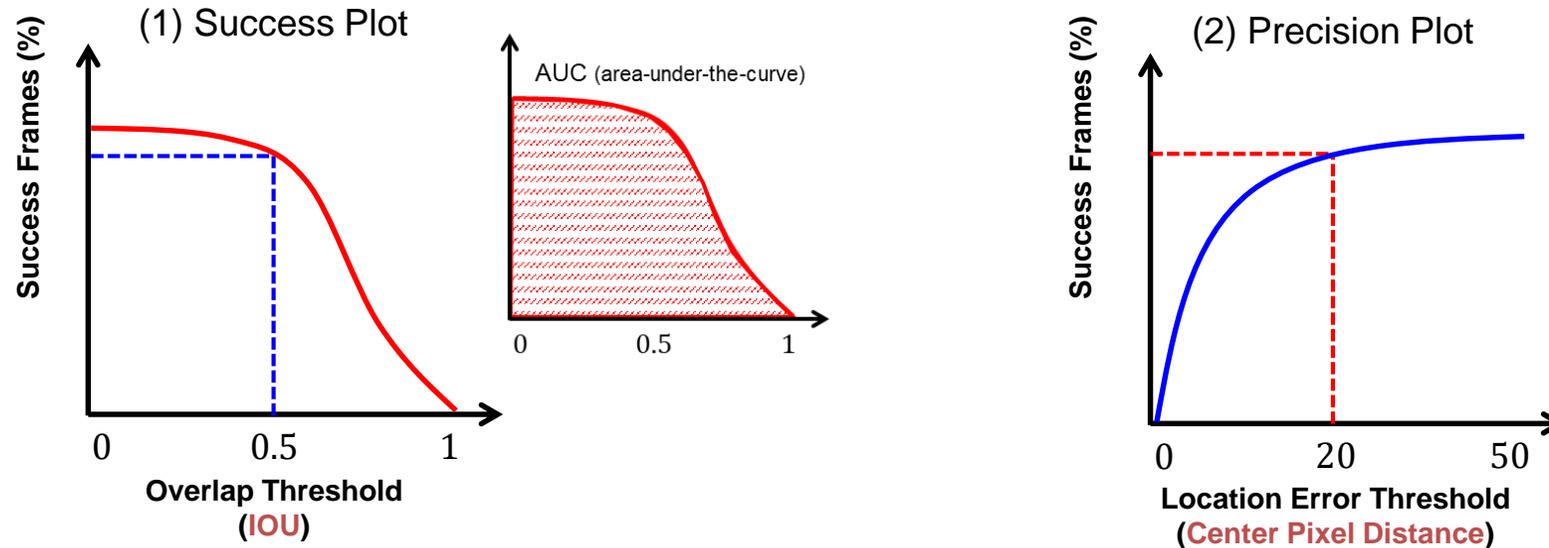
- Training Datasets

- ✓ ImageNet
 - For pre-training shared Siamese Network (*modified AlexNet*)
- ✓ ImageNet Large Scale Visual Recognition Challenge 2015 (*ILSVRC 2015*)
 - For offline training (end-to-end) GCT

- Tracking Benchmark Datasets (Test)

- ✓ OTB-50
- ✓ OTB-100
- ✓ VOT 2017
- ✓ UAV123

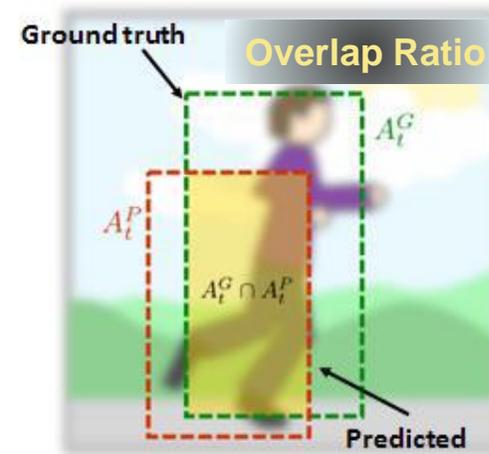
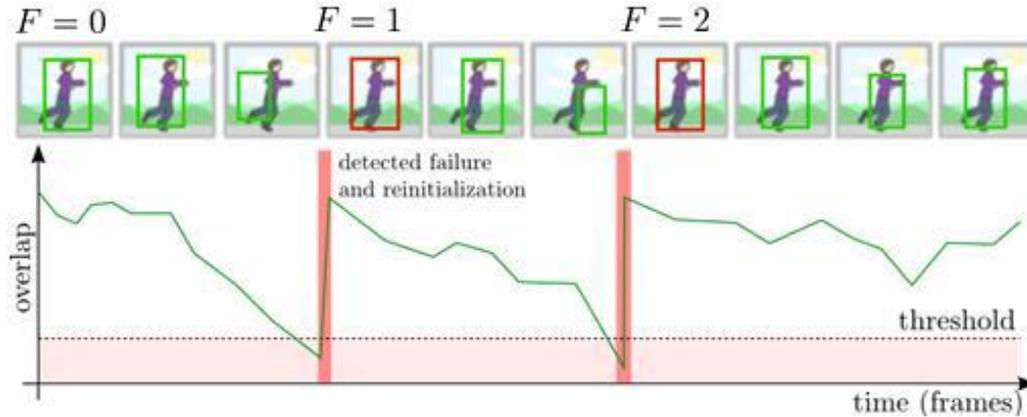
Evaluation Metrics (OTB metric)



Evaluation Metrics (VOT metric)

- **Robustness (R, ↓)** *failure rate*

How many times the tracking failed? (re-initialize to the ground-truth BBOX when failure is detected)



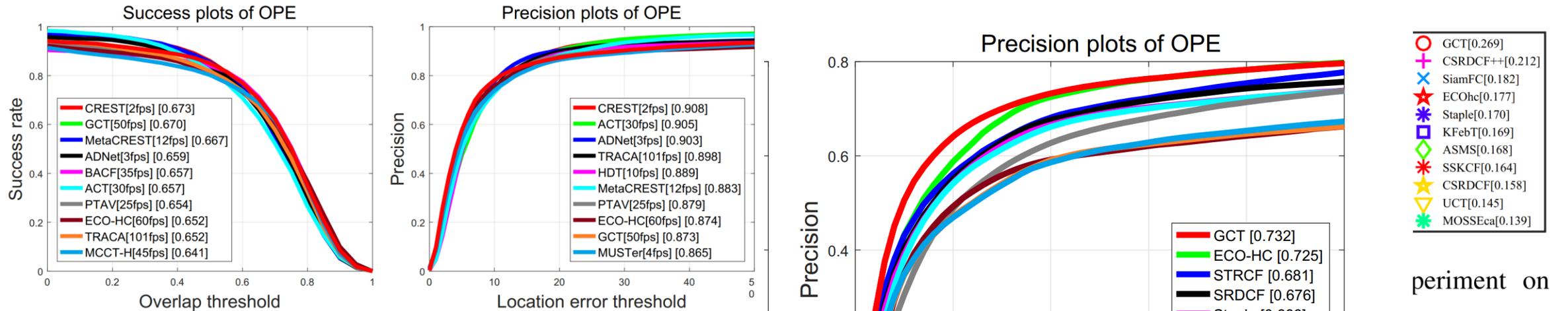
- **Accuracy (A, ↑)** *success rate*

Average overlap ratio under **successful tracking situation**
(re-initialize to the ground-truth BBOX when failure is detected)

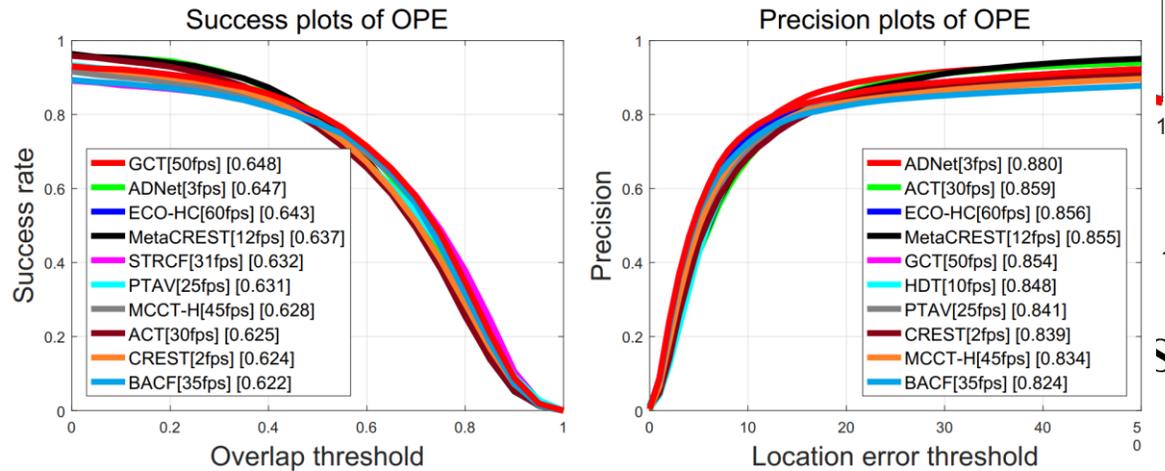
- **Expected Average Overlap (EAO, ↑)** *overall performance*

Comprehensive performance of **Accuracy (A)** and **Robustness (R)**

Experimental Results : Benchmark Results



(a) Results for OTB-2013 benchmark [70]



(b) Results for OTB-2015 benchmark [71]

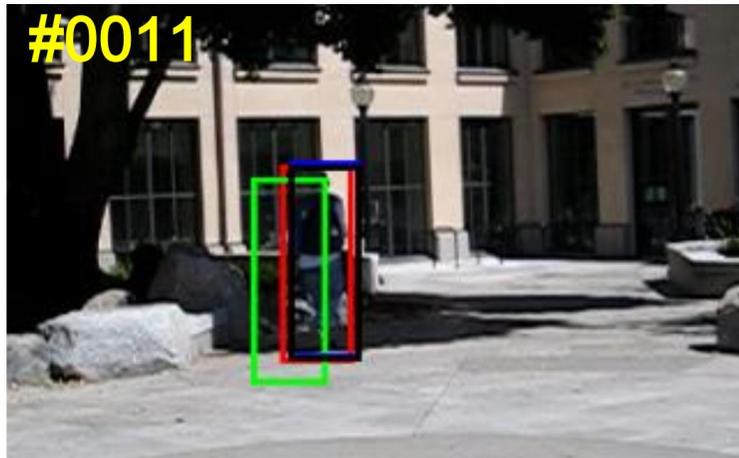
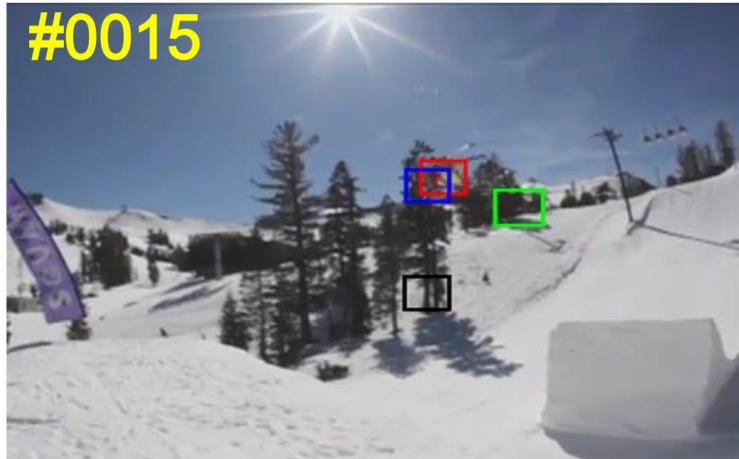
the UAV123 benchmark [40]. Our
s favorably.

Experimental Results : Ablation Studies

Table 2. Analysis of our approach on the OTB-2013 and OTB-2015. The impact of progressively integrating one component at the time, from left to right, is displayed.

	SiamFC	\implies S-GCN	\implies ST-GCN	\implies CT-GCN
AUC { OTB-2013(%)	60.7	62.5	64.9	67.0
OTB-2015(%)	57.7	60.2	63.5	64.8
FPS(OTB-2015)	76.1	66.7	58.6	49.8

Experimental Results : Qualitative Results



— GCT

— SiamFC

— TRACA

— ECO-HC

Source Code Explanation

- PyTorch Training version for GCT Algorithm

[<https://github.com/kyuewang17/GCT-KYLE>] : Private Repository로 생성하여서, **접근 권한**이 필요합니다
(접근 권한은 메일로 GitHub ID를 주시면 부여해 드리도록 하겠습니다. E-mail : kyuewang5056@gmail.com)
(혹은 이메일을 주시면 zip파일로 압축하여서 소스 코드 보내 드릴 수도 있습니다)

- ✓ Dependencies

- GitHub README.md 파일에 표시되어 있습니다.

- ✓ Source Code Referenced from: [[Link](#)]

- 제가 참조한 해당 코드는 미완성입니다. 해당 코드는 학습 코드와 test 코드가 누락되어 있고, 데이터를 load하는 핵심적인 부분이 누락되어 있습니다. (원 저자들은 코드를 아직까지 공개하지 않은 상태입니다)
- 그래서 저는 해당 code 부분에서 end-to-end GCT 네트워크를 논문을 참조하여 완성하였고, 나머지 부분은 PyTorch로 구현된 [SiamFC source](#)를 참조하여 학습(train) 코드 부분만 완성하였습니다.
- 또한 해당 코드에서는 Shared Convnet (Siamese Network backbone)으로 AlexNet을 사용하였다고 하는데, 원래의 모델이 아닌 layer의 dimension이 수정된 버전을 ImageNet에서 미리 학습(pre-train)하여, 학습시에는 ILSVRC 2017 데이터셋으로 backbone 부분을 fine-tuning 하는 형태로 수정된 AlexNet을 이용하였다고 합니다.
- 따라서, 본 기말 프로젝트에서는 제가 구현한 학습 코드가 실제로 학습이 문제 없이 되는지 까지만 나름대로의 scope를 정하여서 프로젝트를 수행하였습니다.

Training Python Message (Screen-Capture)

```
GCT-KYLE ~PycharmProjects/GCT-KYLE
├── .git
├── config
└── DATA
Run: train
/home/kyle/anaconda3/envs/gct_env/bin/python /home/kyle/PycharmProjects/GCT-KYLE/train.py
/home/kyle/anaconda3/envs/gct_env/lib/python3.6/site-packages/sklearn/externals/cloudpickle/cloudpickle.py:47: DeprecationWarning: the imp module is deprecated
---- Starting Train Script of Graph Convolutional Tracker ----
---- Retrieving Metadata for ILSVRC datasets... may take about up to 20 minutes at first time!! ----
---- [Done!] Metadata Retrieved! ----
Done
Setup Time: 0h34m40.49s
Using Exponential Learning Rate Decay of 1
Epoch sizes: 50 in train
Starting training for 50 epoch(s)
Epoch 1/50
Training Start
100% [██████████] 50/50 [03:57<00:00, 4.76s/it, loss=263.437]
[profiling][train] Mean load_data time: 3.7542837047576905
[profiling][train] Mean batch time: 1.0007285070419312
Epoch 2/50
Training Start
100% [██████████] 50/50 [03:58<00:00, 4.33s/it, loss=0.000]
[profiling][train] Mean load_data time: 3.7456852865219115
[profiling][train] Mean batch time: 1.02216730594635
Epoch 3/50
Training Start
100% [██████████] 50/50 [04:03<00:00, 4.66s/it, loss=0.000]
[profiling][train] Mean load_data time: 3.8656660413742063
[profiling][train] Mean batch time: 1.0007685537338258
Epoch 4/50
Training Start
100% [██████████] 50/50 [04:01<00:00, 4.66s/it, loss=0.000]
[profiling][train] Mean load_data time: 3.8262573528287994
[profiling][train] Mean batch time: 1.013555555343628
Epoch 5/50
Training Start
100% [██████████] 50/50 [04:04<00:00, 5.18s/it, loss=0.000]
[profiling][train] Mean load_data time: 3.8525683164596556
[profiling][train] Mean batch time: 1.033919506072990
Epoch 6/50
Training Start
100% [██████████] 50/50 [04:13<00:00, 5.08s/it, loss=0.000]
[profiling][train] Mean load_data time: 4.02801393032074
[profiling][train] Mean batch time: 1.0336401460647504
Epoch 7/50
Training Start
100% [██████████] 50/50 [04:13<00:00, 4.99s/it, loss=0.000]
[profiling][train] Mean load_data time: 4.021969518661499
[profiling][train] Mean batch time: 1.051731114387512
Epoch 8/50
Training Start
100% [██████████] 50/50 [04:13<00:00, 4.75s/it, loss=0.000]
[profiling][train] Mean load_data time: 3.992976250648485
[profiling][train] Mean batch time: 1.0686417293548585
Epoch 9/50
Training Start
100% [██████████] 50/50 [04:10<00:00, 4.95s/it, loss=0.000]
[profiling][train] Mean load_data time: 4.015799546241761
[profiling][train] Mean batch time: 0.9993943405151368
Epoch 10/50
Training Start
100% [██████████] 50/50 [04:09<00:00, 5.01s/it, loss=0.000]
[profiling][train] Mean load_data time: 3.9260040082168578
[profiling][train] Mean batch time: 1.0488302087783814
Epoch 11/50
Training Start
100% [██████████] 50/50 [04:11<00:00, 4.95s/it, loss=0.000]
[profiling][train] Mean load_data time: 4.001342267990112
[profiling][train] Mean batch time: 1.0212286949157714
Epoch 12/50
Training Start
100% [██████████] 50/50 [04:04<00:00, 5.00s/it, loss=0.000]
[profiling][train] Mean load_data time: 3.887248840332031
[profiling][train] Mean batch time: 0.9969201803287397
Epoch 13/50
Training Start
100% [██████████] 50/50 [04:06<00:00, 4.78s/it, loss=0.000]
[profiling][train] Mean load_data time: 3.859667897244263
```

• 학습 결과 설명

- ✓ 이전 슬라이드에서 언급한대로, 학습 코드가 에러 없이 구동되는지 여부를 목표로 하였습니다.
- ✓ 기존 알고리즘은 shared conv-net을 ImageNet 이미지 데이터셋에 미리 학습을 시키고, ILSVRC 2017 데이터셋에 대하여 학습을 시키는 반면, 해당 코드에서는 pre-training 과정 없이 학습을 진행하였습니다.
- ✓ Pre-training 과정이 없이 학습한 결과, loss 자체는 1 epoch만에 0으로 완전 수렴하였으나, 이는 학습 데이터셋에 대하여 weight가 과도하게 over-fitting되었을 것입니다.
- ✓ 추후에 해당 코드를 발전시켜서 완전한 pre-training이 가능하고, visual tracking benchmark 데이터셋에서도 test 가능한 코드를 계속 만들 계획입니다.

• 코드 구성 설명

- ✓ GitHub Repository에 < README.md > 파일로 주요 코드에 대한 설명이 나와 있습니다.

Thank You!