Pedestrian Detection

Kaist Dataset

세종대학교
SEJONG UNIVERSITY

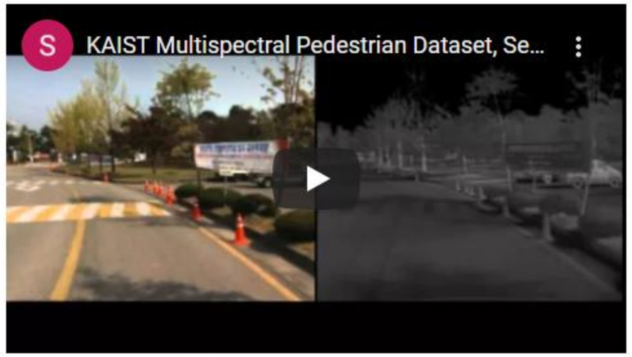
Sejong RCV

KAIST 데이터셋 정보

https://sites.google.com/site/pedestrianbenchmark/data-format

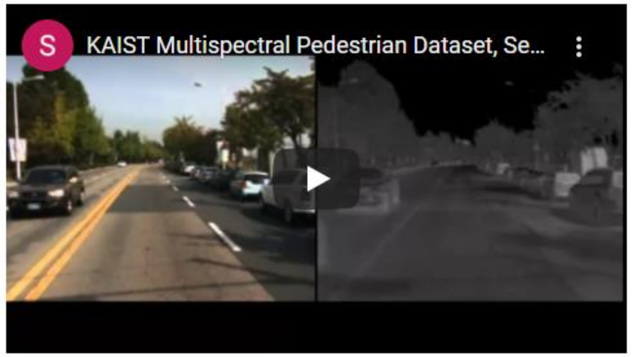
[Training]

- Set00, Day - Campus



Size : 5.92GB / Length : 17,498 frames / xx unique pedestrians / xx unique cyclists

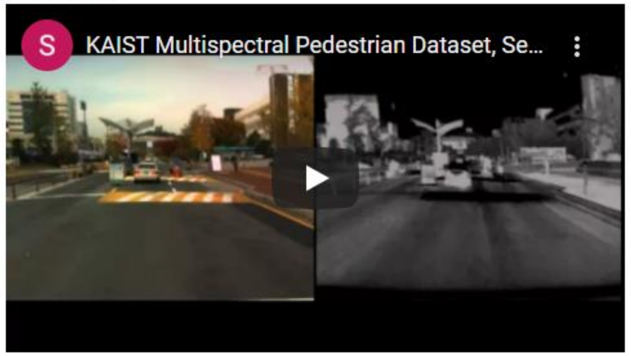
- Set01, Day - Road



Size : 2.82GB / Length : 8,035 frames / xx unique pedestrians / xx unique cyclists

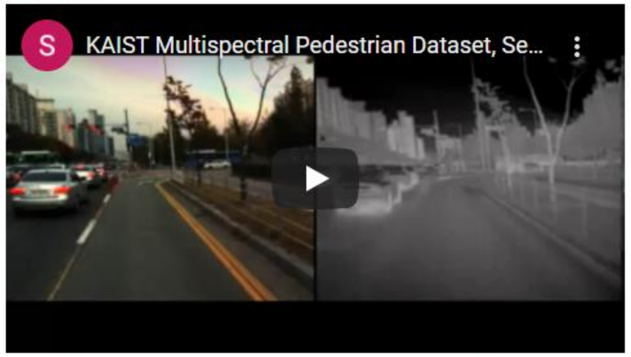
[Testing]

- Set06, Day - Campus



Size : 4.78GB / Length : 12,988 frames / xx unique pedestrians / xx unique cyclists

- Set07, Day - Road

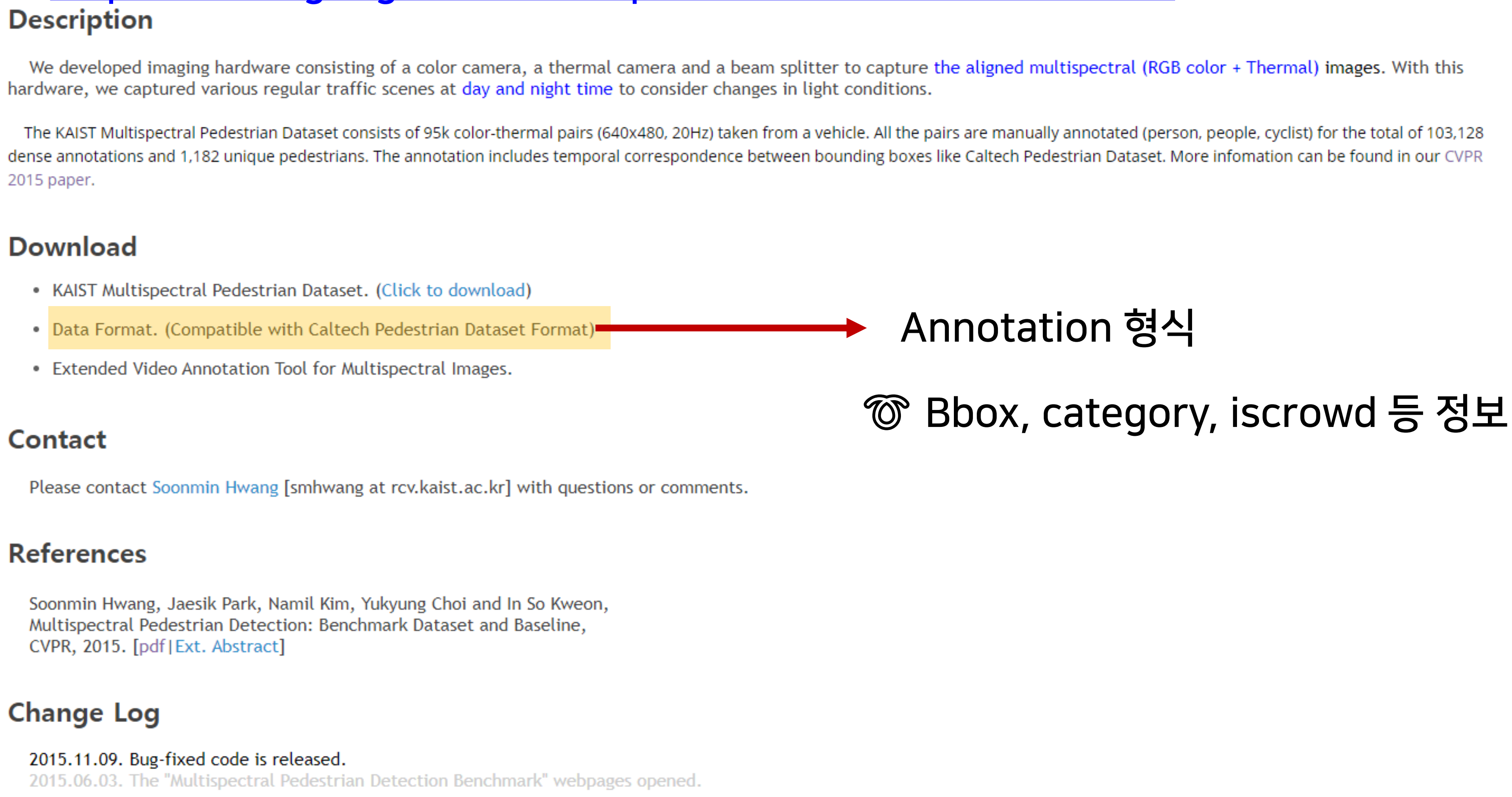


Size : 3.04GB / Length : 8,141 frames / xx unique pedestrians / xx unique cyclists

☑ 해당 사이트에서 set 정보를 비롯하여 이미지를 영상으로 확인할 수 있음

KAIST 데이터셋 정보

https://sites.google.com/site/pedestrianbenchmark/home



Description

We developed imaging hardware consisting of a color camera, a thermal camera and a beam splitter to capture the aligned multispectral (RGB color + Thermal) images. With this hardware, we captured various regular traffic scenes at day and night time to consider changes in light conditions.

The KAIST Multispectral Pedestrian Dataset consists of 95k color-thermal pairs (640x480, 20Hz) taken from a vehicle. All the pairs are manually annotated (person, people, cyclist) for the total of 103,128 dense annotations and 1,182 unique pedestrians. The annotation includes temporal correspondence between bounding boxes like Caltech Pedestrian Dataset. More infomation can be found in our CVPR 2015 paper.

Download

- KAIST Multispectral Pedestrian Dataset. (Click to download)
- Data Format. (Compatible with Caltech Pedestrian Dataset Format)
- Extended Video Annotation Tool for Multispectral Images.

Contact

Please contact Soonmin Hwang [smhwang at rcv.kaist.ac.kr] with questions or comments.

References

Soonmin Hwang, Jaesik Park, Namil Kim, Yukyung Choi and In So Kweon,
Multispectral Pedestrian Detection: Benchmark Dataset and Baseline,
CVPR, 2015. [pdf | Ext. Abstract]

Change Log

2015.11.09. Bug-fixed code is released.
2015.06.03. The "Multispectral Pedestrian Detection Benchmark" webpages opened.

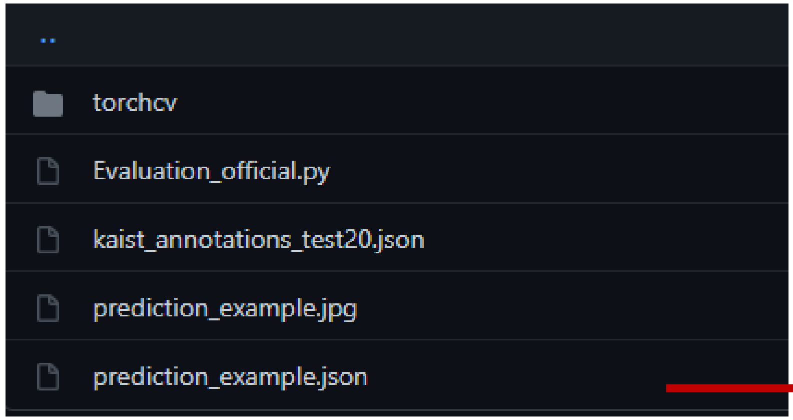
$\longrightarrow$ Annotation 형식

➰ Bbox, category, iscrowd 등 정보 확인 필요 ➰

✅ 그 외 다양한 정보들을 확인할 수 있으므로 들어가보고 데이터셋을 둘러볼 것을 추천 🔥

예측값에 대한 json file 생성하기

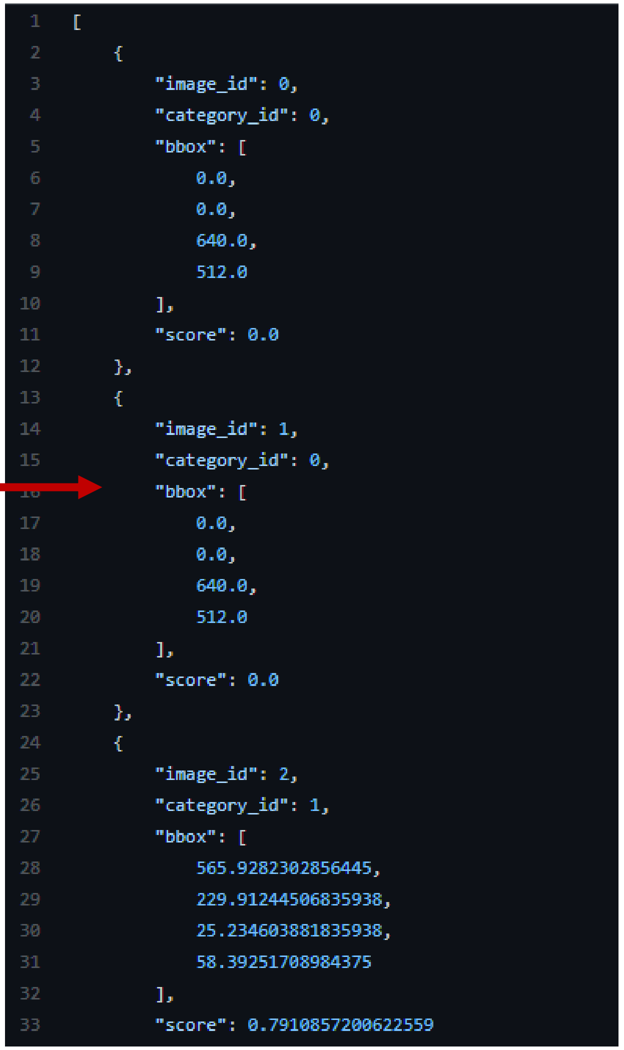
▼ 깃허브에 올라온 3주차 > official_Evaluation 파일리스트



```
..
torchcv
Evaluation_official.py
kaist_annotations_test20.json
prediction_example.jpg
prediction_example.json
```

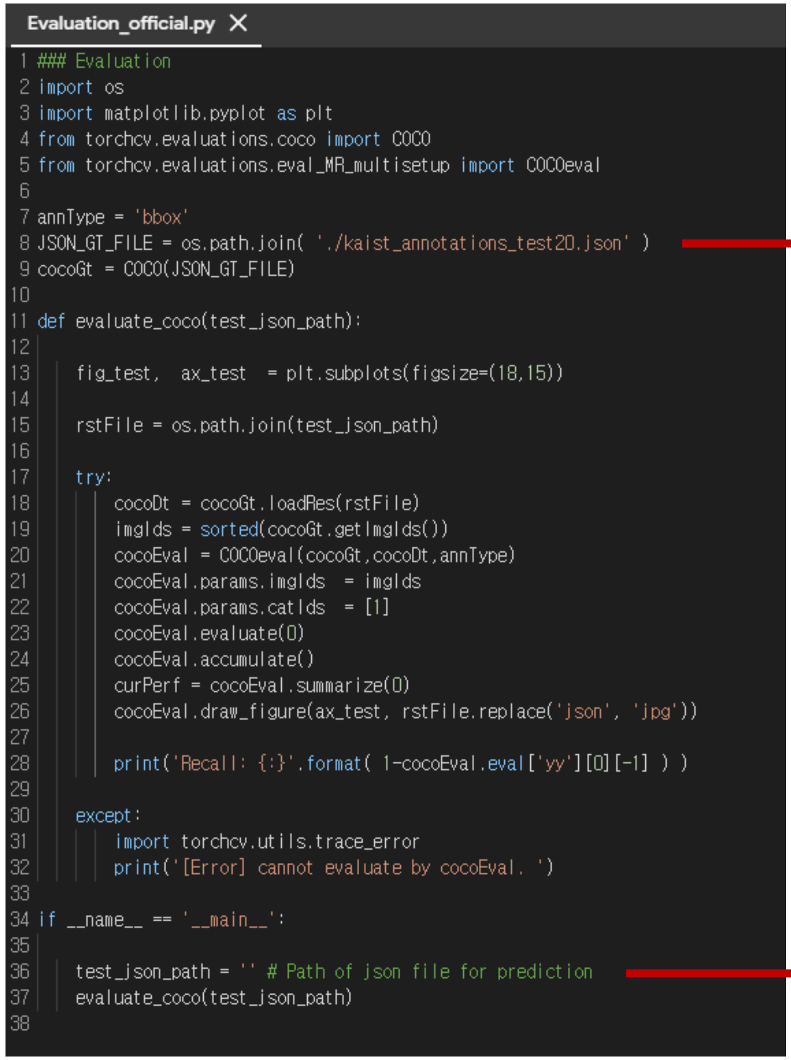
Hint.
Detect 부분의 코드를 사용해서 test에 대한 image_id 및 bbox 좌표 등에 대한 하나의 json 파일로 만들기

Json 파일의 형식은 predict_example.json을 확인 바람
한 이미지 당 "image_id" , "category_id", "bbox", "score" 필요



```
[
    {
        "image_id": 0,
        "category_id": 0,
        "bbox": [
            0.0,
            0.0,
            640.0,
            512.0
        ],
        "score": 0.0
    },
    {
        "image_id": 1,
        "category_id": 0,
        "bbox": [
            0.0,
            0.0,
            640.0,
            512.0
        ],
        "score": 0.0
    },
    {
        "image_id": 2,
        "category_id": 1,
        "bbox": [
            565.9282302856445,
            229.91244506835938,
            25.234603881835938,
            58.39251708984375
        ],
        "score": 0.7910857200622559
```

평가 코드 돌리기



Evaluation_official.py

```python
### Evaluation
import os
import matplotlib.pyplot as plt
from torchcv.evaluations.coco import COCO
from torchcv.evaluations.eval_MR_multisetup import COCOeval

annType = 'bbox'
JSON_GT_FILE = os.path.join( './kaist_annotations_test20.json' )
cocoGt = COCO(JSON_GT_FILE)

def evaluate_coco(test_json_path):

    fig_test,  ax_test  = plt.subplots(figsize=(18,15))

    rstFile = os.path.join(test_json_path)

    try:
        cocoDt = cocoGt.loadRes(rstFile)
        imgIds = sorted(cocoGt.getImgIds())
        cocoEval = COCOeval(cocoGt,cocoDt,annType)
        cocoEval.params.imgIds  = imgIds
        cocoEval.params.catIds  = [1]
        cocoEval.evaluate(0)
        cocoEval.accumulate()
        curPerf = cocoEval.summarize(0)
        cocoEval.draw_figure(ax_test, rstFile.replace('json', 'jpg'))

        print('Recall: {:}'.format( 1-cocoEval.eval['yy'][0][-1] ) )

    except:
        import torchcv.utils.trace_error
        print('[Error] cannot evaluate by cocoEval. ')

if __name__ == '__main__':

    test_json_path = '' # Path of json file for prediction
    evaluate_coco(test_json_path)

```

Test에 대한 GT json file.
모델이 예측한 정보와 비교하기 위해 필요함.
제공되는 Evaluation_official 폴더 안에 함께 있기 때문에,
굳이 경로를 바꿔주지 않아도 됨.

앞서 생성한 json 파일의 경로를 넣으면 됨 !

✓ 경로 설정 후 ! python Evaluation_official.py 로 코드 돌리면 성능 확인 가능