




# Sử dụng tool “Near miss accident prediction using dashcam video”

Created by	 Khánh Huỳnh Hoàng
Created time	@July 24, 2023 1:22 PM
Tags	Product

## Sử dụng tool với Python cell trong Jupyter.notebook hoặc Google Colabs

**LƯU Ý: TOOL ĐÒI HỎI CÓ GPU**

**Clone github:**

```
!git clone https://github.com/keeganhuyh/Near-miss-accident-prediction-using-dashcam-videos.git
```

**Cài đặt các thư viện cần thiết và set up môi trường chạy:**

```
!pip install filterpy
!pip install pykml
```

```
%cd /content/Near-miss-accident-prediction-using-dashcam-videos/DeepHough
%cd model/_cdht
!python setup.py build
!python setup.py install --user
```

**Chạy file run.py**

```
%cd /content/Near-miss-accident-prediction-using-dashcam-videos
!python run.py --folderpath '/content/Near-miss-accident-prediction-using-dashcam-videos/Runs/sample_741376'\
               --videopath '/content/Near-miss-accident-prediction-using-dashcam-videos/Runs/sample_741376/sample_741376.mp4'\
               --KML_file_path ''
```



### INPUT

- **Chú thích:** có hai tham số cần thiết để tool chạy là VIDEO và FILE KML. Trong đó, video được truyền vào tool bằng đường dẫn đến Video và tương tự với file KML.
  - folderpath: là đường dẫn đến thư mục chính chứa video và file KML
  - videopath: đường dẫn đến video đầu vào
  - KML\_file\_path: đường dẫn đến file KML (vì sample này không có file KML, nên em chuẩn bị sẵn một file vận tốc trích từ file csv lớn nên không cần truyền file KML)



## OUTPUT:

Những mục sau sẽ được lưu vào “**folderpath**”

- **file txt:** vanishing\_point\_list
- **file\_velocity:** velocity\_list
- **predict\_collision\_vehicle\_each\_frame:** risk\_json
- **trajectory\_file:** data\_json
- **video\_visuallize:** output\_video.mp4



## Format Input

1) VIDEO được đọc bằng OpenCV. Nên bất kì loại video nào có thể được đọc bởi OpenCV đều có thể được sử dụng trong tool này.

2) Vận tốc của ego car có thể được truyền vào tool bằng hai cách:

- File KML tiêu chuẩn: tool sẽ tự đọc được nếu có sẵn file KML
- File txt: mỗi dòng chứa vận tốc của xe đơn vị tính (m/s). Nếu video có tổng cộng 223 frame thì sẽ có 223 dòng ứng với vận tốc ở từng frame trong video.

⇒ Lưu cả hai input trên vào một folder (ví dụ như: **sample\_741376**) được sử dụng trong repo này. Và truyền đường dẫn vào model khi sử dụng



## Các hàm chính trong file run.py

```
def MakeInput(folderpath):
    fps, frame_count = ProcessVideo(folderpath, videopath)
    VanishingPointDetection(output_path, vnp_output_path, videopath, stage=2)

fps, frame_count = MakeInput(folderpath, vnp_output_path, videopath, stage=2)
```

Hàm **MakeInput**:

- Xử lí video thành đầu vào cho model Deephough
- Chạy model Deephough trên video “*videopath*” và Trích xuất ra những vanishing point sau đó lưu vào “*vnp\_output\_path*”

```
TrajectoryAndMakingVideo(videopath, vnp_output_path, veclocity_path, json_file_path, fps, (1280, 720))
```

#### Hàm **TrajectoryAndMakingVideo**:

- Chạy model Yolo để detect những phương tiện trên đường
- Trajectory vị trí những xe từ những công cụ (CameraCalibration, ObjectSpeedEstimate, Sort) trên đường rồi lưu vào *"json\_file\_path"*
- Dự đoán va chạm với model LSTM rồi lưu vào metadata *"json\_file\_path"*

```
risk_json_file = makeJson(frame_count, json_file_path, risk_json_path)
```

#### Hàm **makeJson**:

- Trích xuất những phương tiện có khả năng va chạm từ *"json\_file\_path"* thành file json *"risk\_json\_path"*

```
ExtractVideo('tam', frame_count, fps, video_path)
```

#### Hàm **ExtractVideo**:

- Visualize video và lưu vào *"video\_path"*

---

## Tác giả

Giáo viên hướng dẫn: Thầy Đào Minh Sơn

Sinh viên thực hiện: Huỳnh Hoàng Khánh (Khoa Khoa học máy tính - UIT)