RWR 4013

Digital Twins for Smart Cities

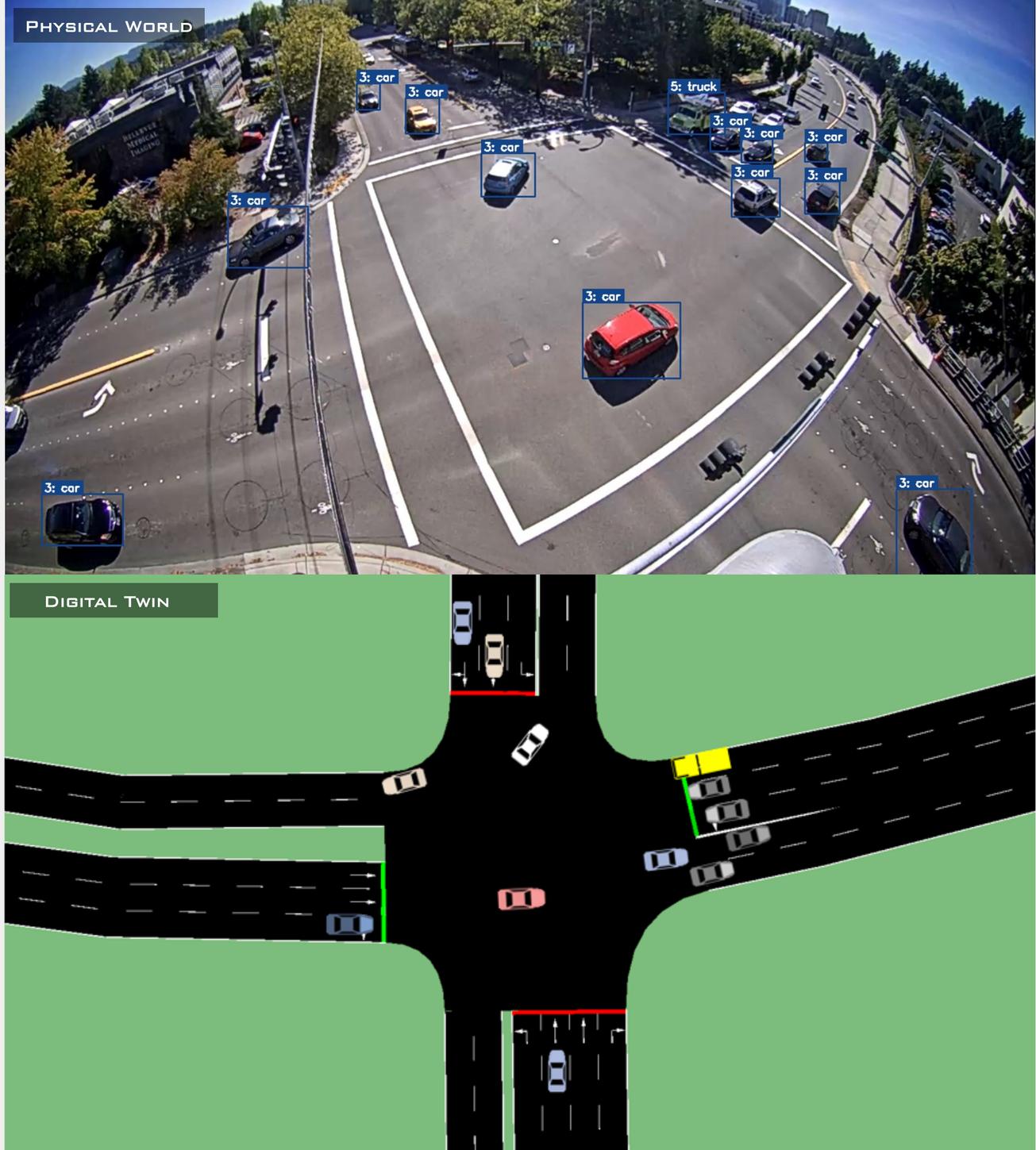
Dr. Ahmad Mohammadi

Week 3 | Session 1:
Computer Vision II
(Object Tracking)

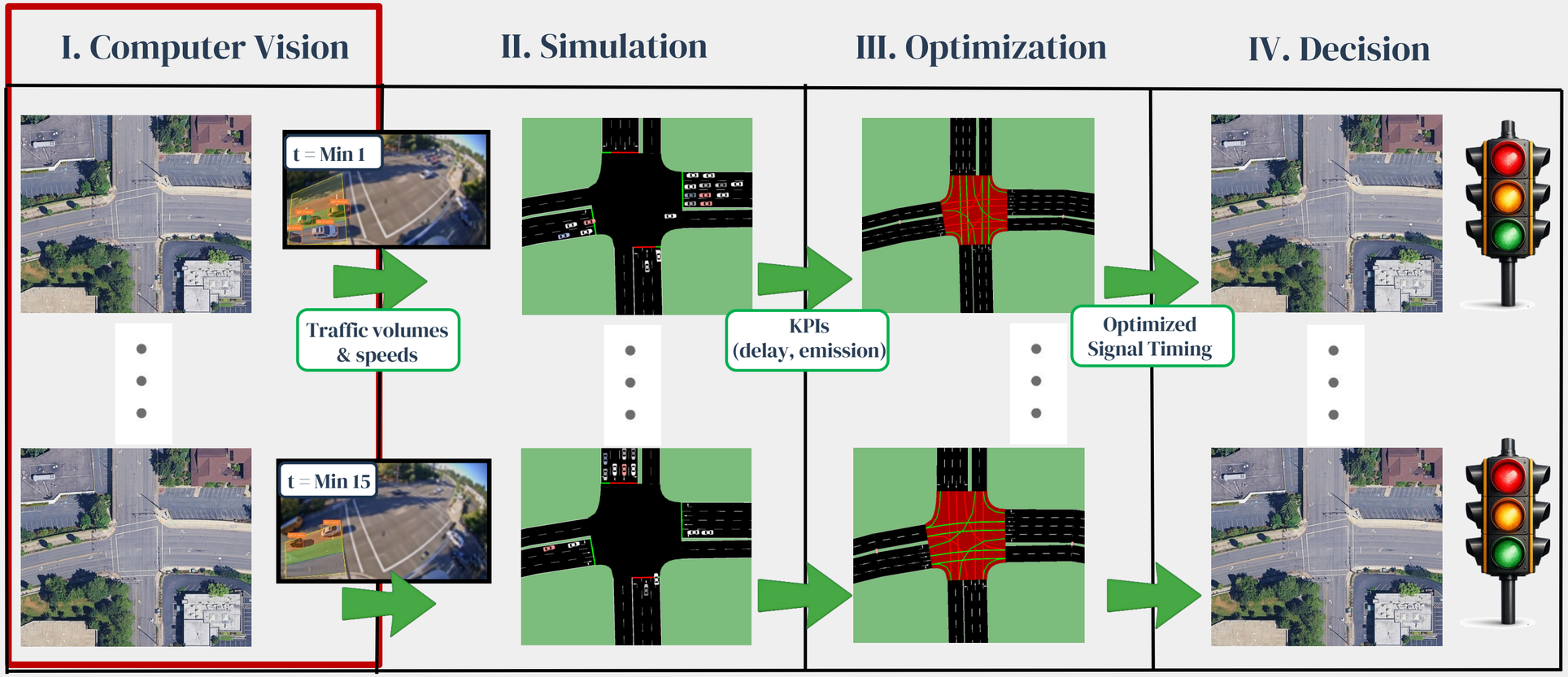
Fall 2026

RoadwayVR

roadwayvr.github.io/DigitalTwinsforSmartCities

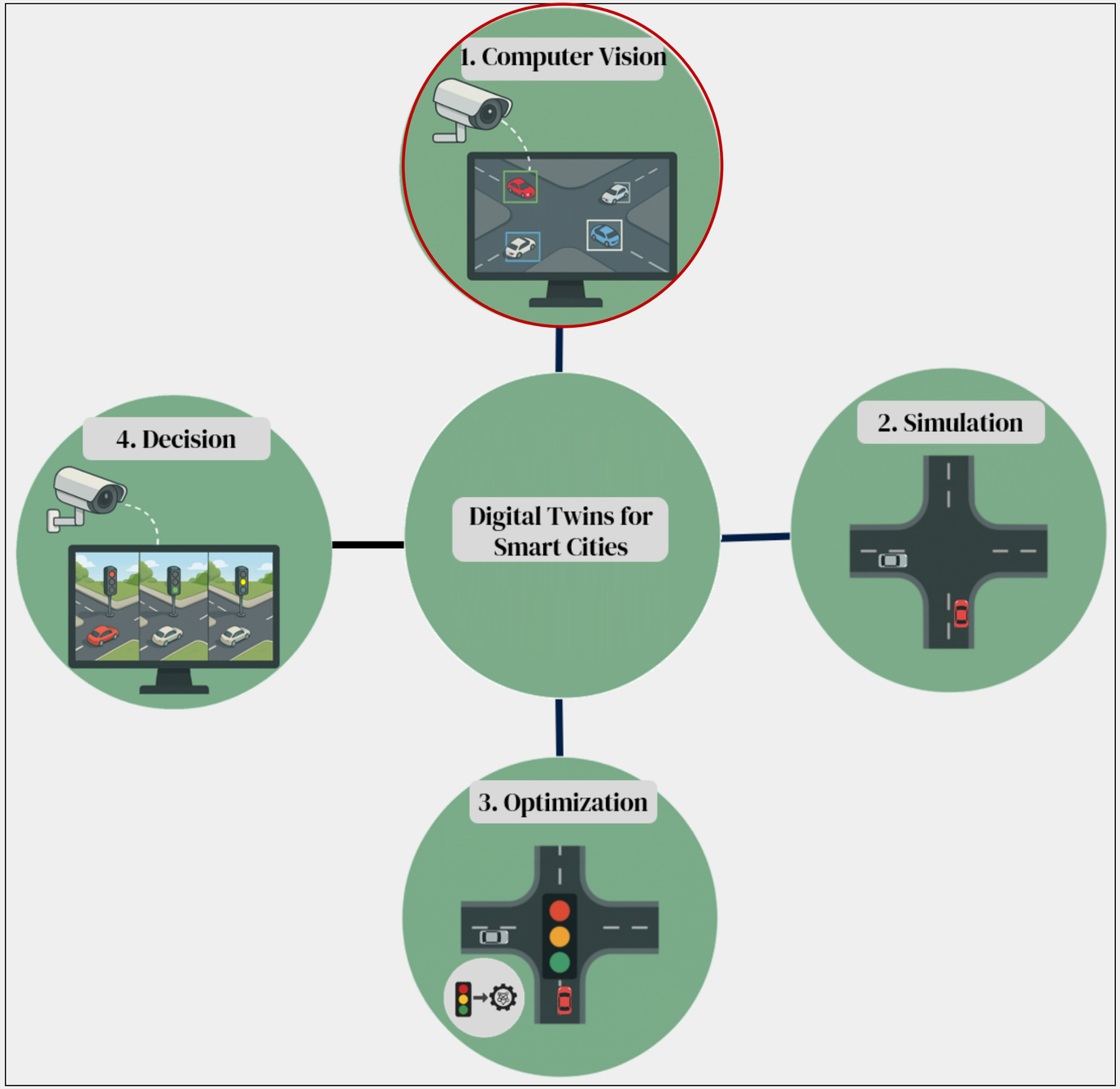


Overview of Course Syllabus in One Shot



Agenda

- ❑ Object Detection vs Object Tracking
- ❑ Bird’s-Eye View (Homography)
- ❑ Multi-Object Tracking (ByteTrack)
- ❑ SuperVision (Roboflow)
- ❑ SimJamComputerVision
- ❑ Object Detection and Tracking Using SimJamCV



Object Detection vs Object Tracking

Object Detection

Output: bounding box + class + confidence score (per frame)

“Confidence can change each frame.”

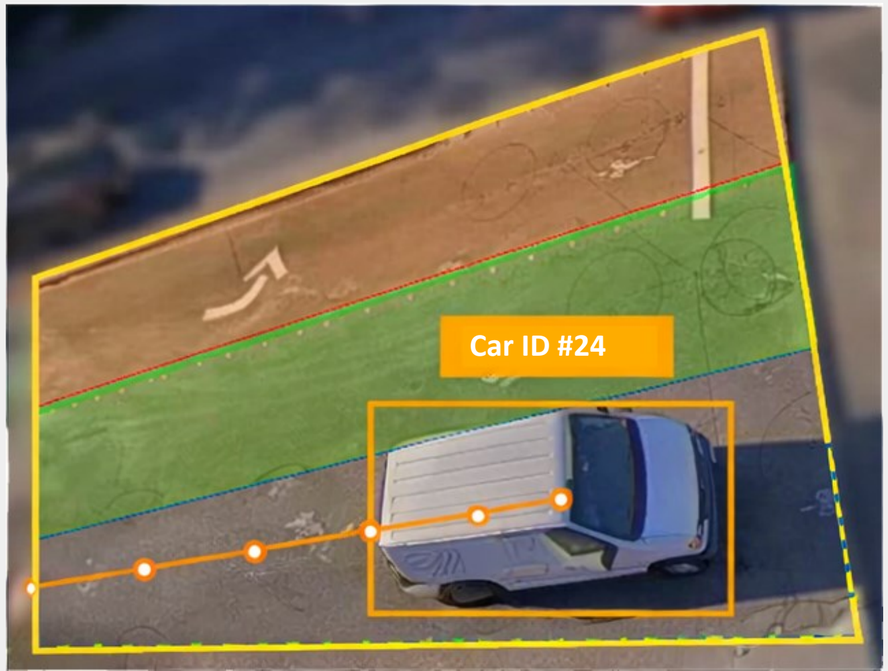
Time = 45 sec Time = 46 sec Time = 47 sec

Object Tracking

Output: object ID + trajectory over time (and often a box each frame)

“Keeps the same ID even as the object moves.”

Time = 45 sec Time = 46 sec Time = 47 sec



Object Detection vs Object Tracking

Object detection (per frame)

Answers: **“What is in this frame, and where is it?”**

Outputs:

- **bounding box** (x, y, w, h)
- **class / object type** (car, person, etc.)
- **confidence score** (how sure the detector is)

Notes:

- It **does no guarantee** that “Car in frame 45” is the ***same*** car as “Car in frame 46”.

Object tracking (across frames)

Answers: **“Which object is which over time, and how did it move?”**

Outputs:

- **object ID** (e.g., #24)
- **trajectory over time** (a sequence of positions)

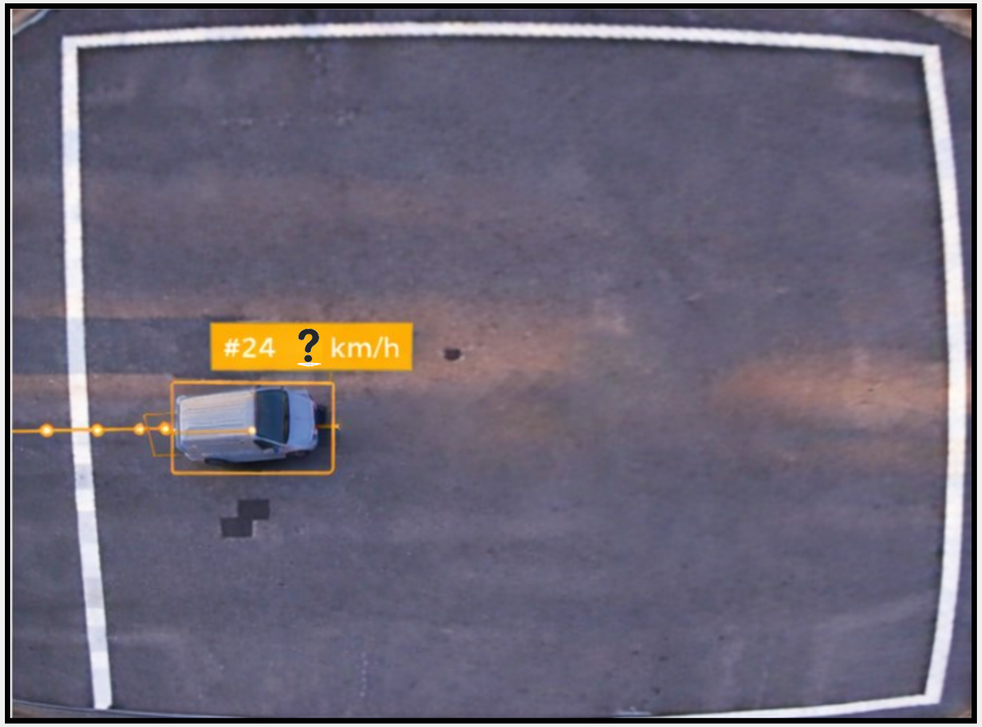
Notes:

- The trajectory is the continuous **center points (x, y)** of the box in each frame
- It guarantees that “Car in frame 45” is the *same* car as “Car in frame 46” by giving identical ID (but can sometimes switch IDs in difficult cases).
- We can use to estimate speed

Which View is Better for Speed Estimation?

Angled camera view

Bird’s-eye view



Which View is Better for Speed Estimation?

Bird’s-eye view makes distance measurement more uniform → more accurate speed estimation.

Distances are more uniform:

You can convert pixels to meters more accurately, so speed can be estimated in m/s or km/h.

Bird’s-eye view

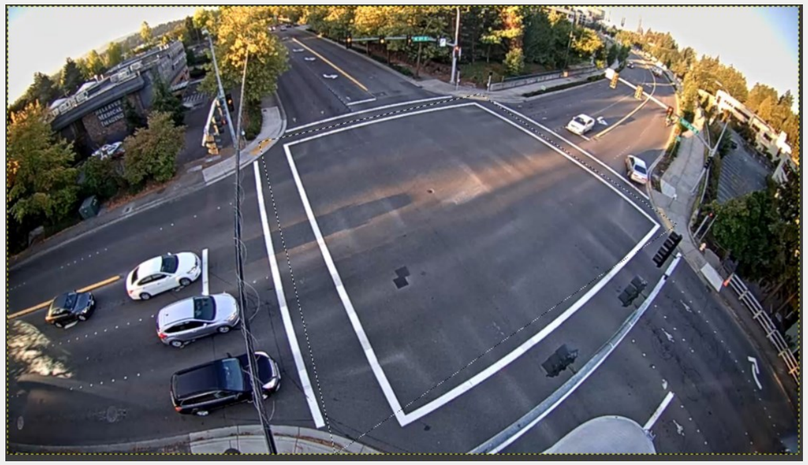
Bird’s-Eye View (Homography) for Speed

Two Steps for Speed Estimation:

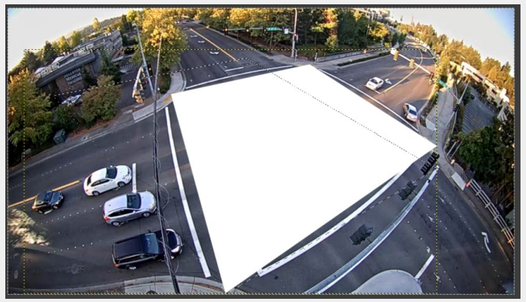
1. **Homography:** Map image pixels to the road plane (i.e., transform angled view to bird’s-eye view)

2. **Compute speed:** Measure distance (meters) over time → compute speed (m/s or km/h).

Real-world image

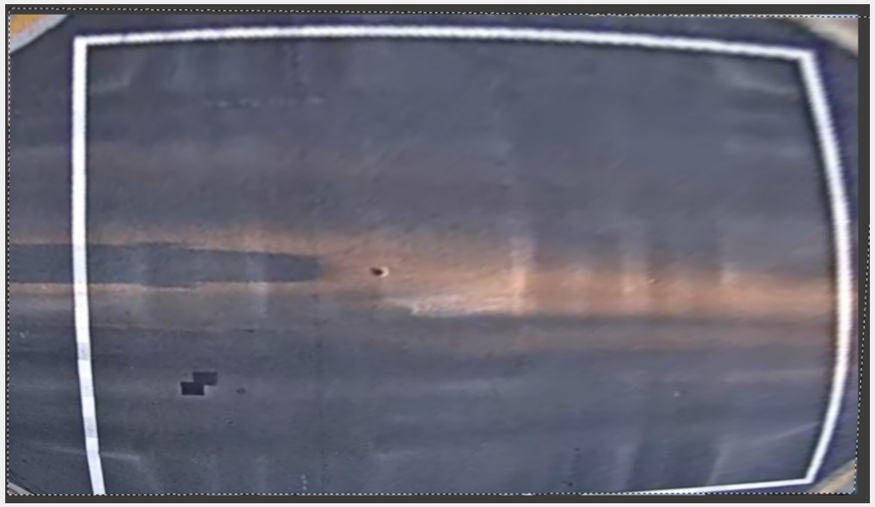


Area of Interest



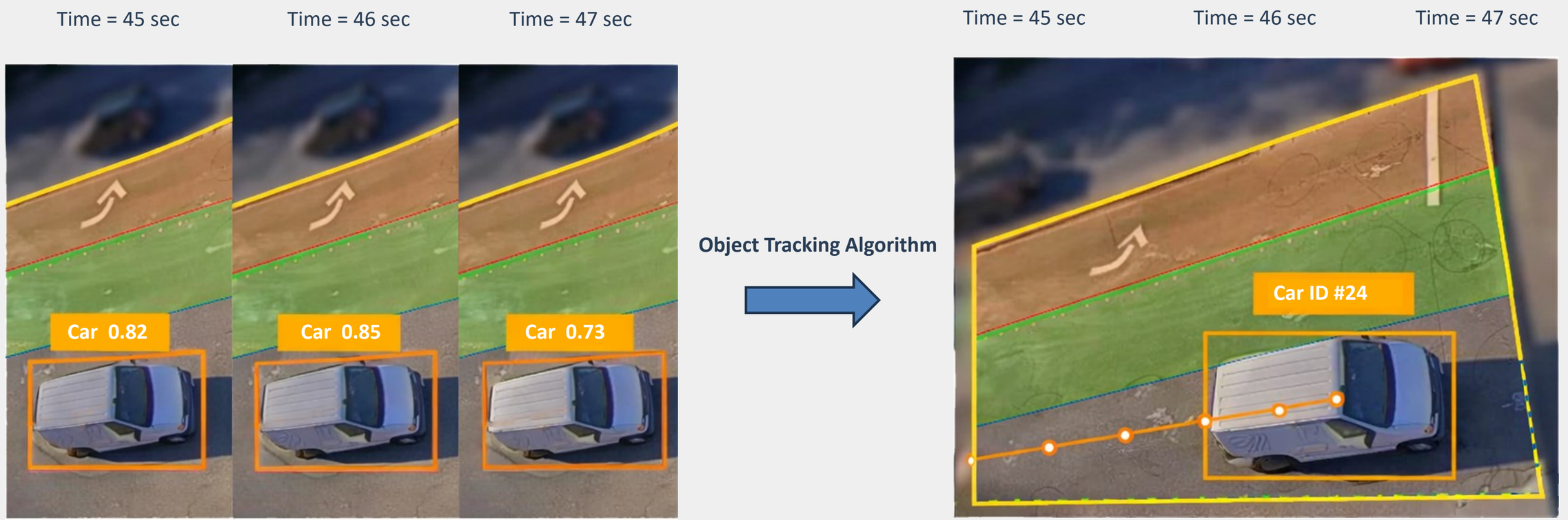
Homography

Bird’s-eye view



Multi-Object Tracking (ByteTrack)

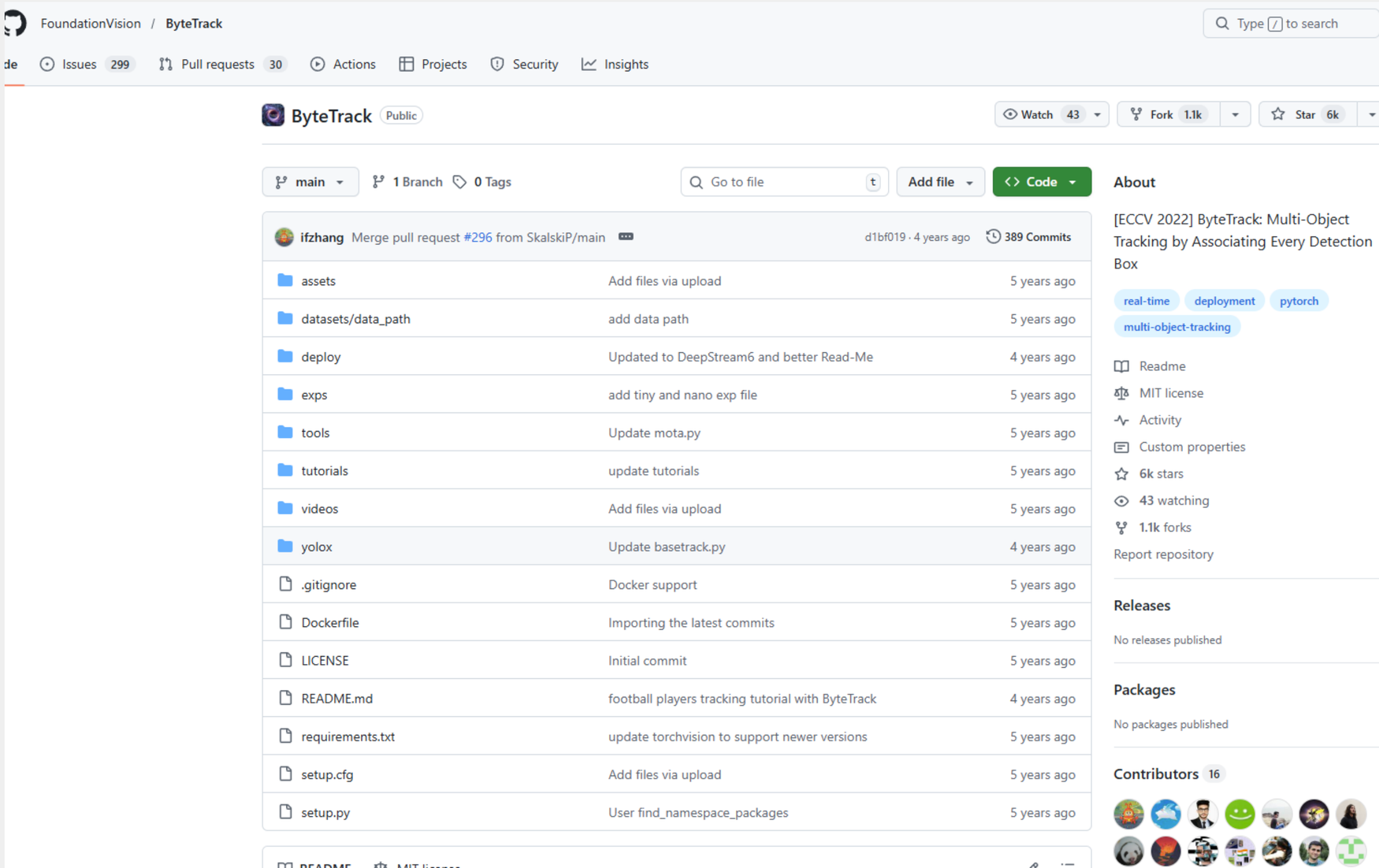
- **ByteTrack** is a popular multi-object tracking method that **links detections across video frames over time**
- It is a **classical rule-based tracking algorithm** (it does not "learn" like a neural network; it **associates** detections using motion + matching rules).



Multi-Object Tracking (ByteTrack)

➢ ByteTrack Method

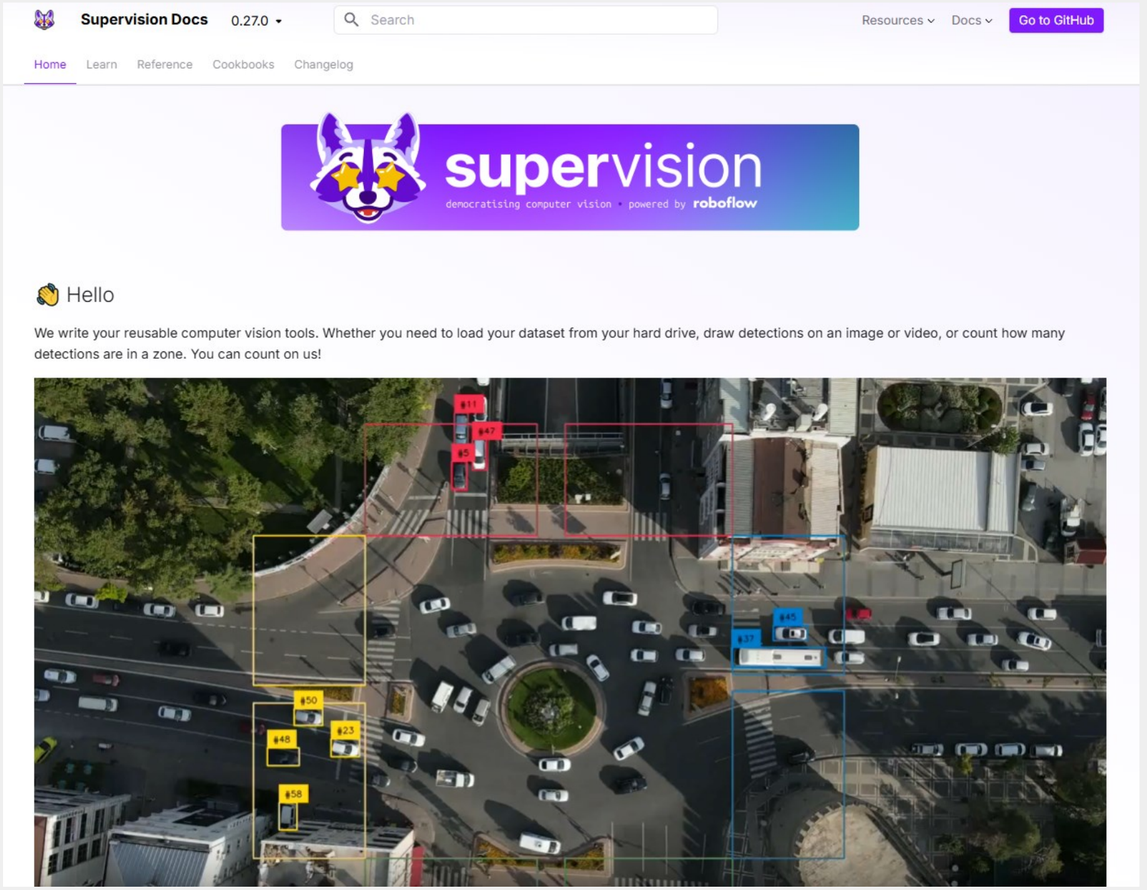
https://github.com/FoundationVision/ByteTrack



Supervision (Roboflow)

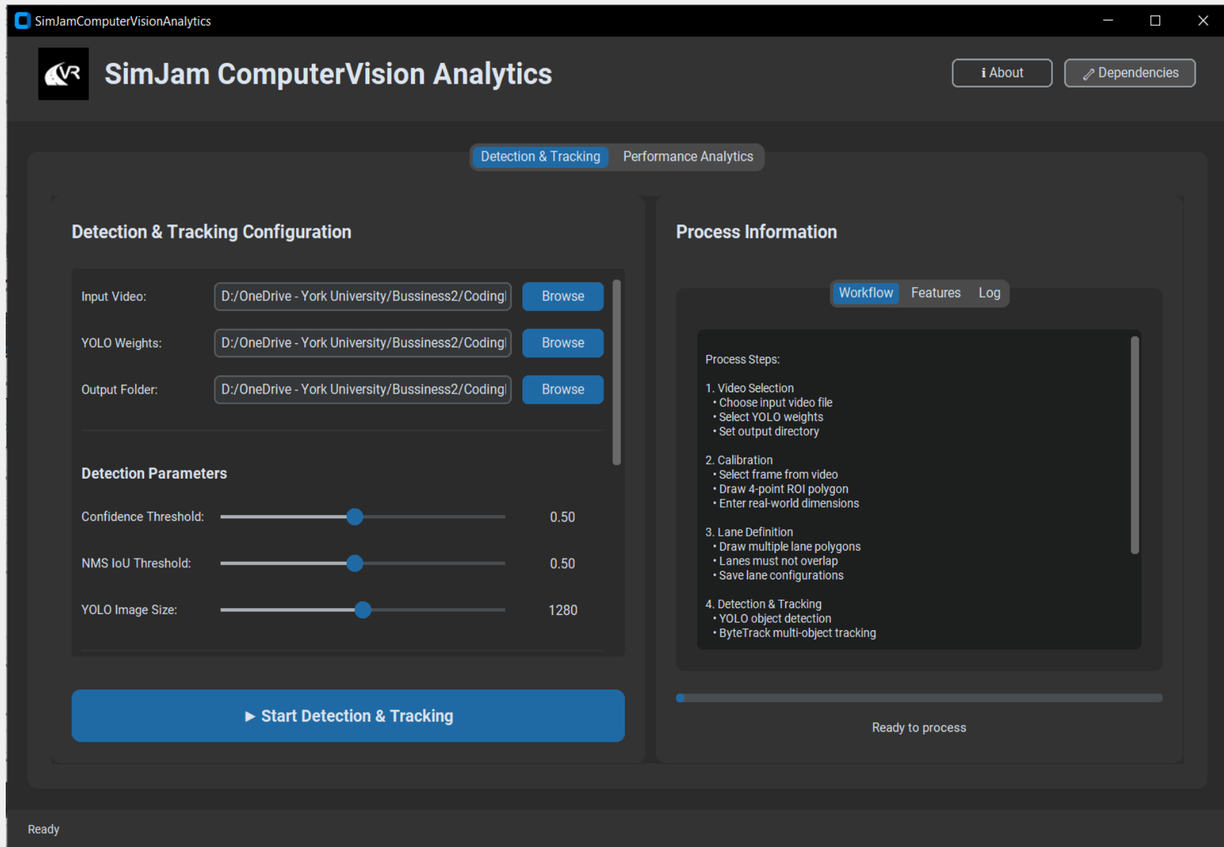
- **Supervision:** a Python library for CV development (tracking & visualization).

https://supervision.roboflow.com/latest/



SimJamComputerVision

- **SimJamComputerVision:** A no-code educational tool by Dr. Ahmad for automatic object detection and tracking.
- **Input:** Video
- **Output:** Object Count, Object Type, Speed

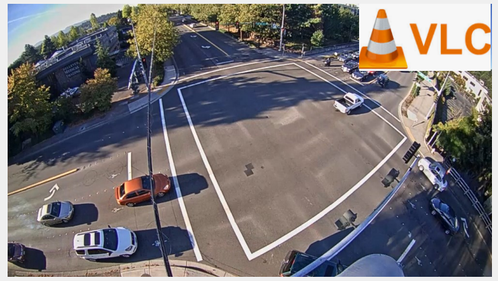


- *Powered by YOLO + ByteTrack (via Supervision).*

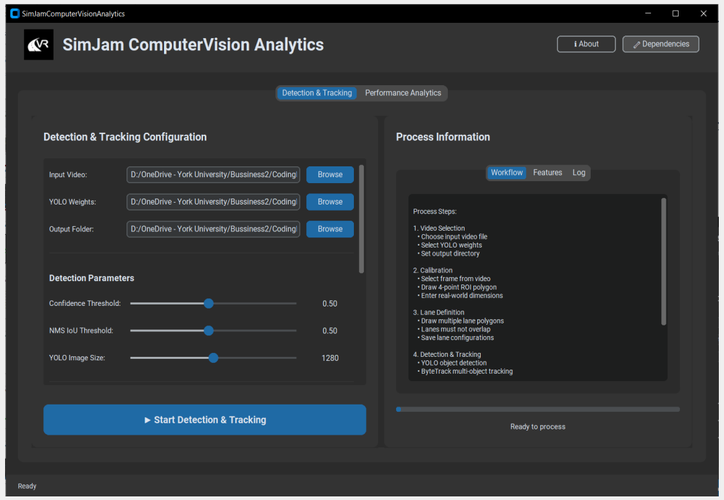
SimJamComputerVision

Input

Video

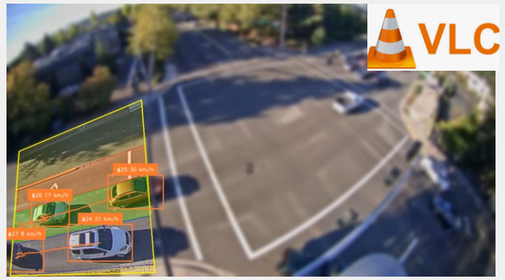


Application



Output

1



2

	A	B	C	D	E	F	G
1	frame	time_s	vehicle_id	x_m	y_m	img_x	img_y
153	24	0.828	5	8.434	5.408	214.7	453.2
154	24	0.828	2	13.458	8.481	143.6	556.6
155	24	0.828	4	8.459	10.723	100.4	485.7
156	24	0.828	1	18.412	5.113	224.4	637.7
157	24	0.828	6	13.085	1.623	320.8	515.1
158	24	0.828	3	3.903	12.637	69	429.7
159	24	0.828	7	18.272	13.953	33.3	653.6
160	25	0.862	5	8.439	5.413	214.5	453.3
161	25	0.862	4	8.463	10.723	100.4	485.8
162	25	0.862	2	13.455	8.482	143.6	556.6
163	25	0.862	1	18.419	5.107	224.5	637.8
164	25	0.862	6	13.093	1.627	320.7	515.3
165	25	0.862	3	3.987	12.868	65	432.3
166	25	0.862	7	18.275	13.936	33.6	653.7
167	26	0.897	5	8.44	5.413	214.5	453.3
168	26	0.897	4	8.463	10.723	100.4	485.8
169	26	0.897	2	13.455	8.482	143.6	556.6
170	26	0.897	1	18.419	5.107	224.5	637.8
171	26	0.897	6	13.093	1.627	320.7	515.3
172	26	0.897	3	3.987	12.869	65	432.3
173	26	0.897	7	18.274	13.936	33.6	653.6
174	27	0.931	5	8.441	5.412	214.6	453.3
175	27	0.931	2	13.454	8.482	143.6	556.6
176	27	0.931	4	8.467	10.721	100.4	485.8
177	27	0.931	6	13.095	1.627	320.7	515.3
178	27	0.931	1	18.418	5.111	224.4	637.8
179	27	0.931	3	4.023	13.025	62.3	433.7
180	27	0.931	7	18.316	13.936	33.5	654.3

3

	A	B	C	D	E	F	G	H	I	J	K	L
1	lane_id	n_vehicles	avg_delay	avg_speed	avg_stopp	measured	measured	estimated	green_tim	green_rati	vc_ratio	los
2	1	20	2.203631	20.82616	1.8761	382.1372	544.3658	272.1829	0	0.5	1.403972	A
3	2	18	1.640515	29.68999	3.054056	343.9235	895.2255	447.6127	0	0.5	0.768351	A
4	3	15	3.473889	11.11831	5.669	286.6029	688.3804	344.1902	0	0.5	0.832688	A

Object Detection and Tracking Using SimJamCV

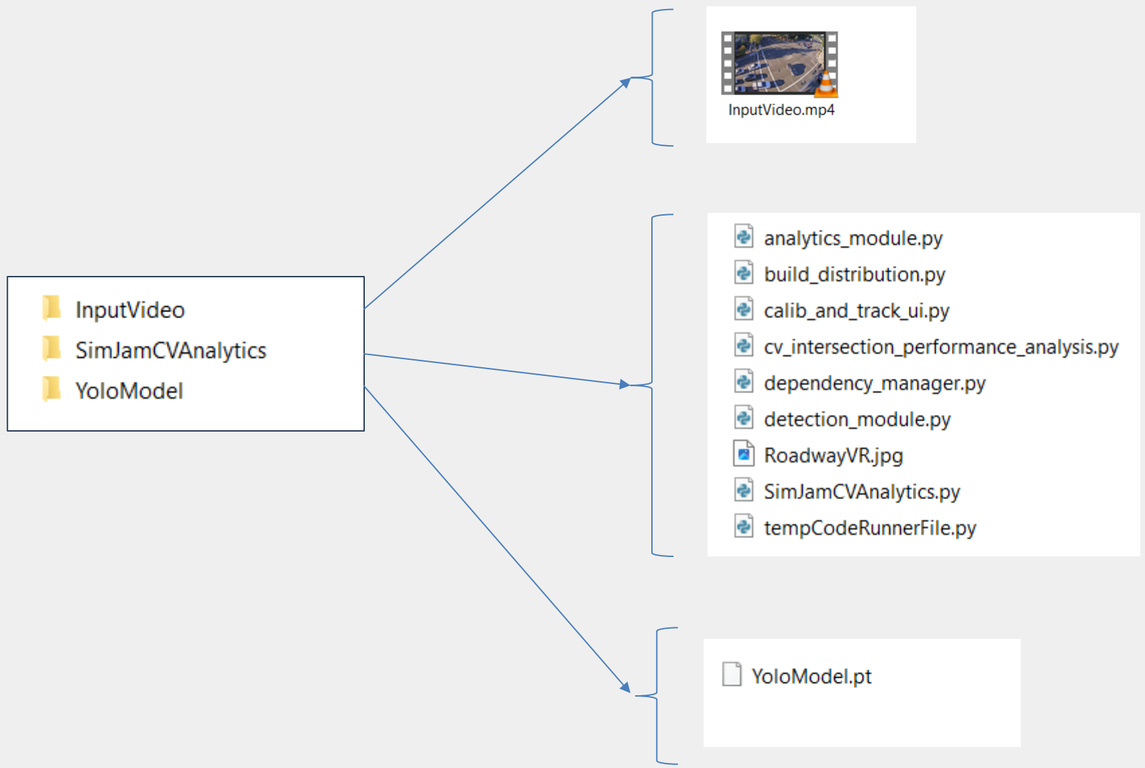
Steps:

1. Download Week3a.Material.zip and Extract it
2. Run SimJamCVAnalytics.py
3. Install Dependencies
4. Choose input video file
5. Adjust Detection/Tracking Parameters
6. Select YOLO Algorithm

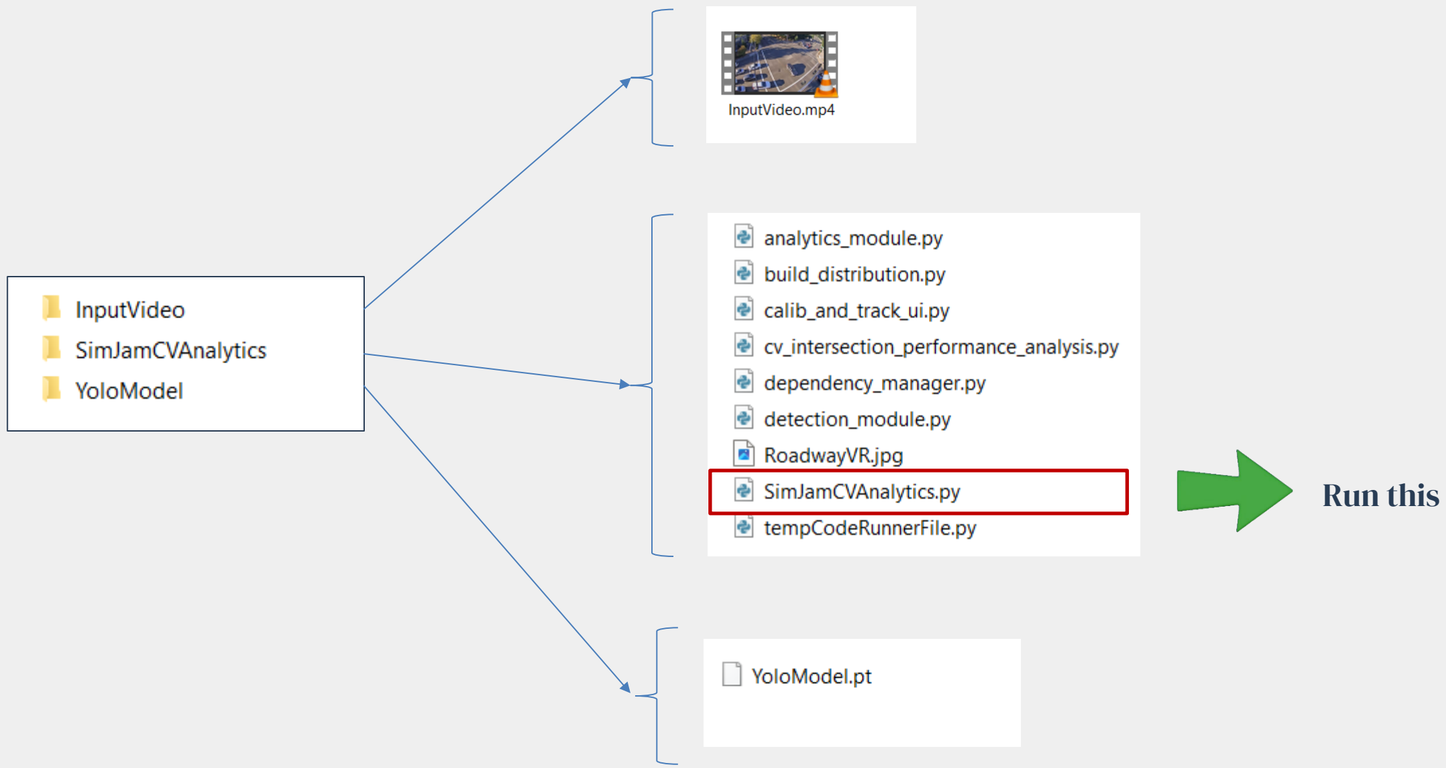
Steps:

7. Set output directory
8. Run Application
9. Preprocessing
10. Algorithm Running
11. Understanding Output
12. Performance Analytics

Step 1. Download Week3.Material.zip and Extract it



Step 2. Run SimJamCVAnalytics.py

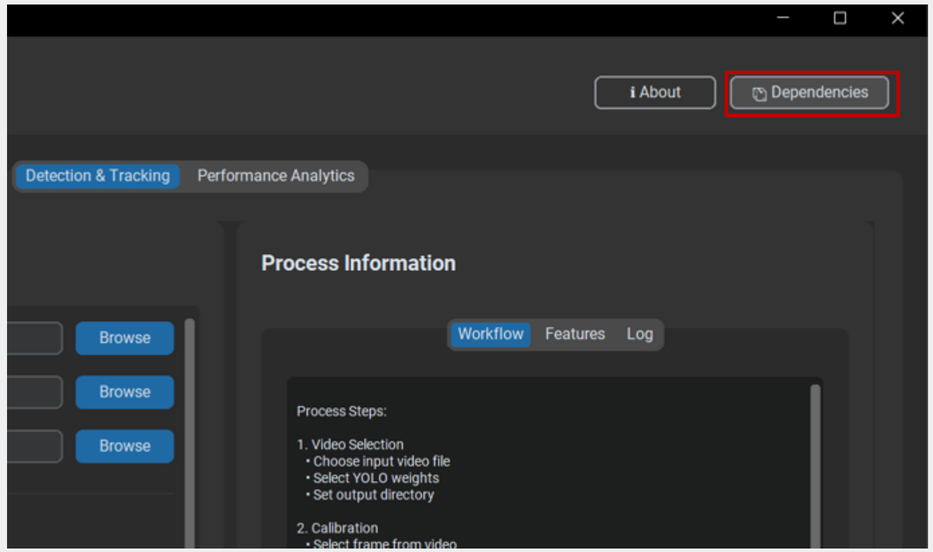


Step 3. Install Dependencies

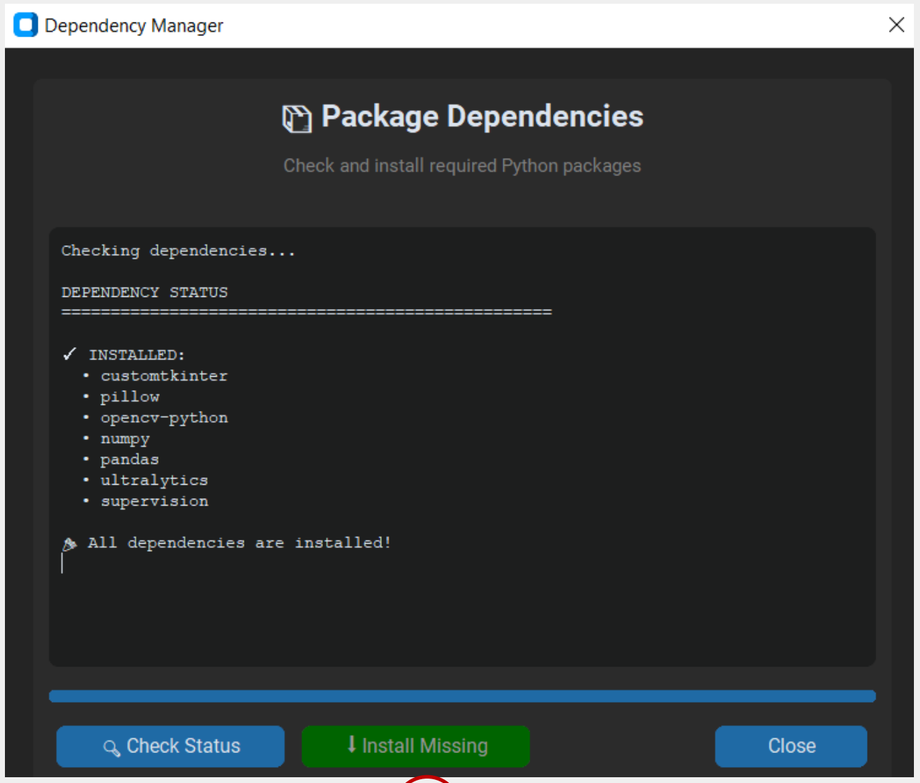
1. After Running SimJamCVAnalytics.py

2. Install Dependencies

1

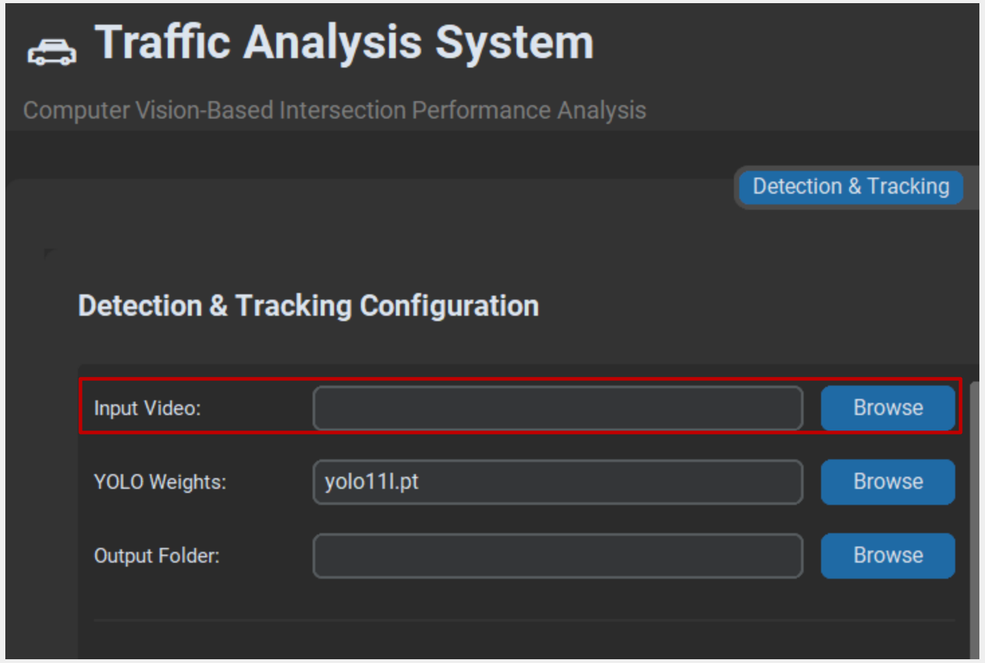


2



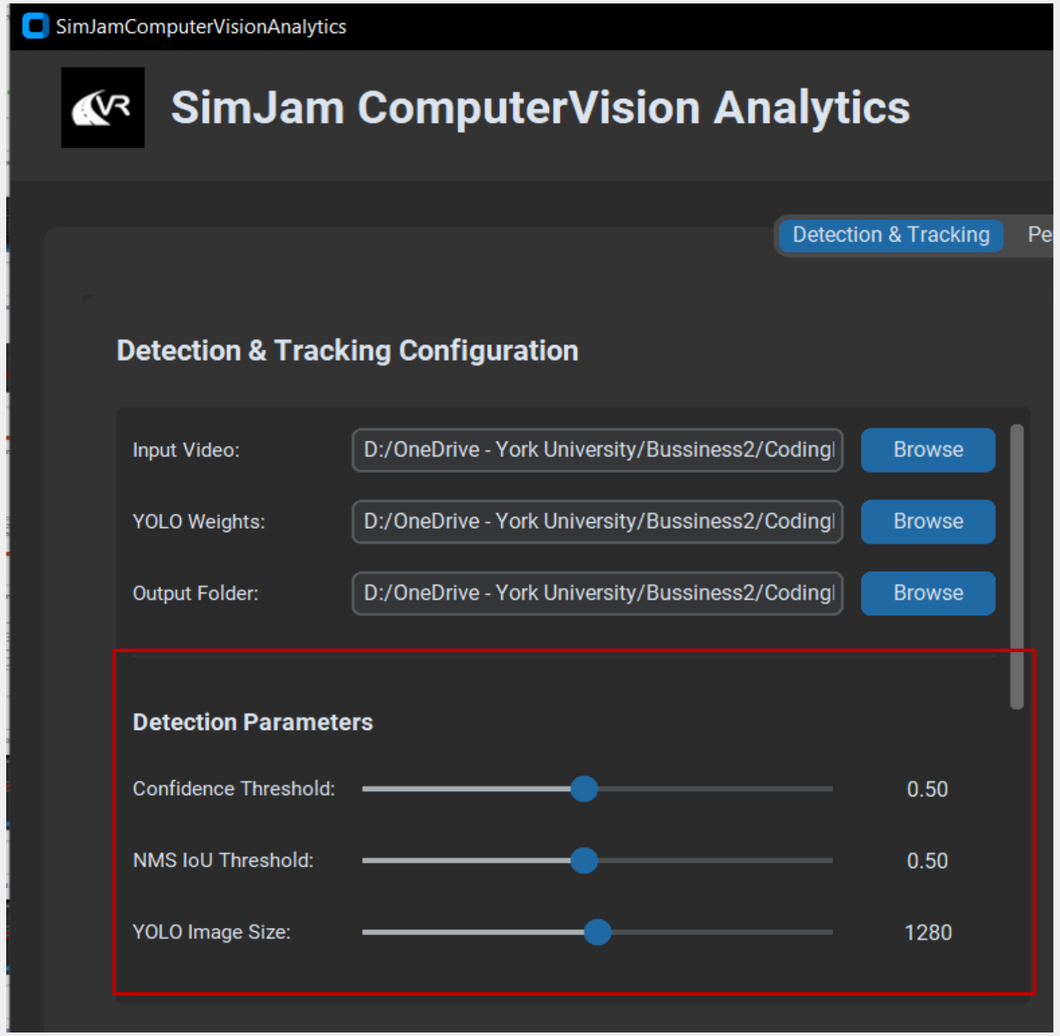
3

Step 4. Choose Input Video File



1. Choose input video file
2. Select YOLO Algorithm
3. Set output directory

Step 5. Adjust Detection/Tracking Parameters



1. **Confidence Threshold (0.50):** Algorithm assigns each detection a confidence score (0-1). Only detections above this threshold are kept.
2. **NMS IoU Threshold (0.50):** Controls duplicate detection removal (Non-Maximum Suppression)
3. **YOLO Image Size (1280):** Resizes video frames before feeding to YOLO

Step 5. Adjust Detection/Tracking Parameters

1

1. Confidence Threshold (0.50)

- **What it does**: Sets the minimum confidence score required for a detection to be accepted
- **Range**: 0.10 to 0.95
- **How it works**: YOLO assigns each detection a confidence score (0-1). Only detections **above** this threshold are kept.
- **Example**:

0.50 = "I'm 50% sure this is a vehicle" → **ACCEPTED**

0.30 = "I'm 30% sure" → **REJECTED**

- **Adjust when**:

Too many false detections (shadows, signs) → **Increase** (try 0.65-0.75)

Missing real vehicles → **Decrease** (try 0.35-0.45)

2

2. NMS IoU Threshold (0.50)

- **What it does**: Controls how duplicate detections are removed (Non-Maximum Suppression)
- **Range**: 0.10 to 0.95
- **How it works**: When multiple boxes detect the same vehicle, NMS keeps only the best one. This threshold defines how much boxes must overlap to be considered duplicates.
- **Example**:

  0.50 = Boxes overlapping >**50% overlap** are duplicates → Remove weaker one

  0.70 = More lenient, allows more overlapping boxes
- **Adjust when**:

  Multiple boxes on same vehicle → **Decrease** (try 0.30-0.40)

  Adjacent vehicles merged into one → **Increase** (try 0.65-0.75)

3

3. YOLO Image Size (1280)

- **What it does**: Resizes video frames before feeding to YOLO
- **Range**: 640 to 1920 pixels
- **How it works**: YOLO processes images at this resolution (largest side), while preserving the aspect ratio.
- **Example**:

  1280 = Better accuracy, slower processing

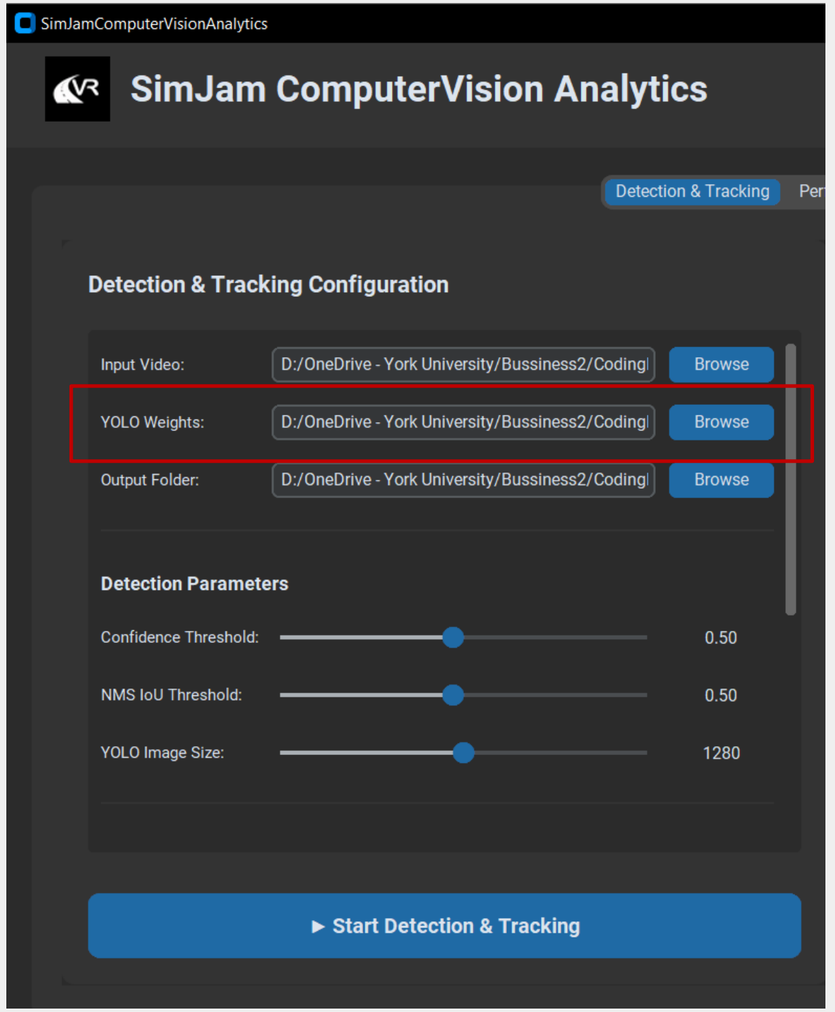
  640 = Faster processing, may miss small/distant vehicles
- **Adjust when**:

  Slow processing → **Decrease** (try 960 or 640)

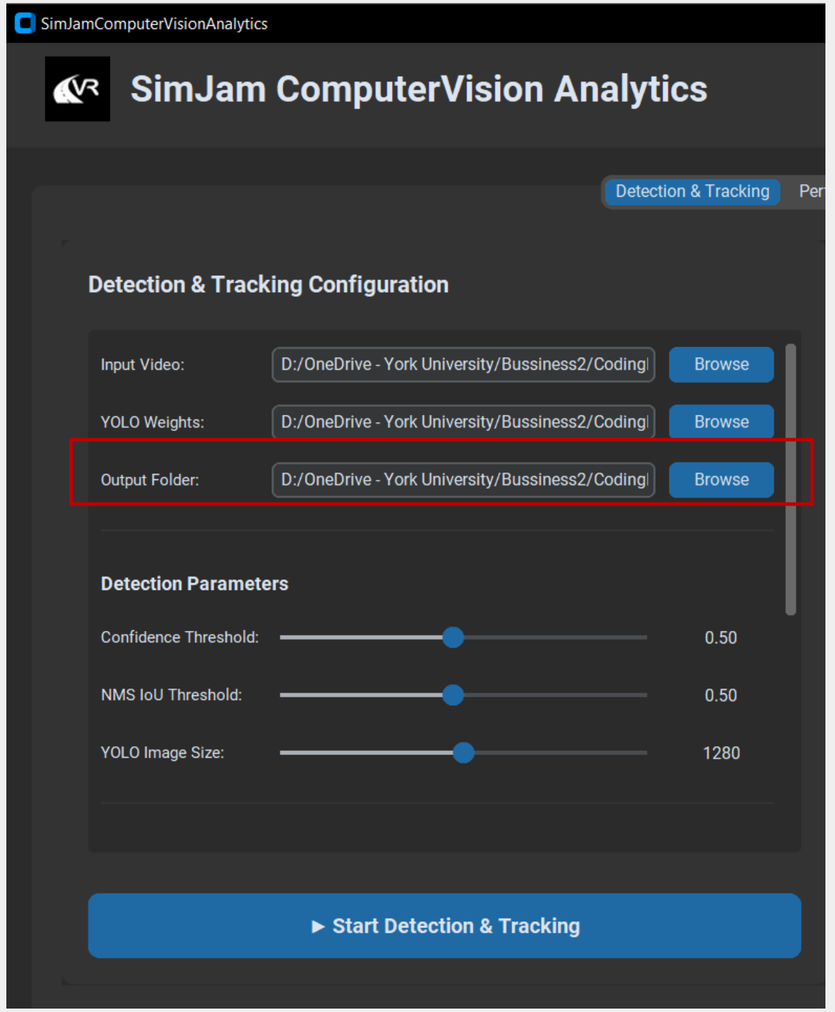
  Missing distant vehicles → **Increase** (try 1920)

  GPU memory errors → **Decrease** significantly

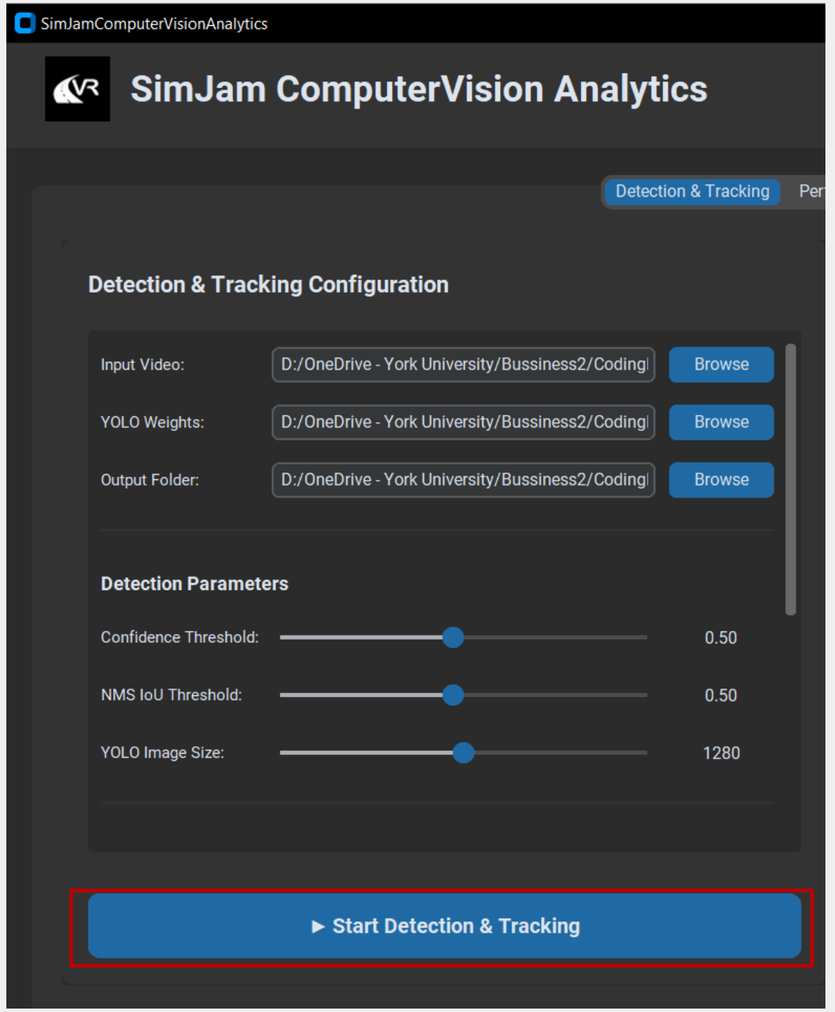
Step 6. Select YOLO Algorithm



Step 7. Set Output Directory

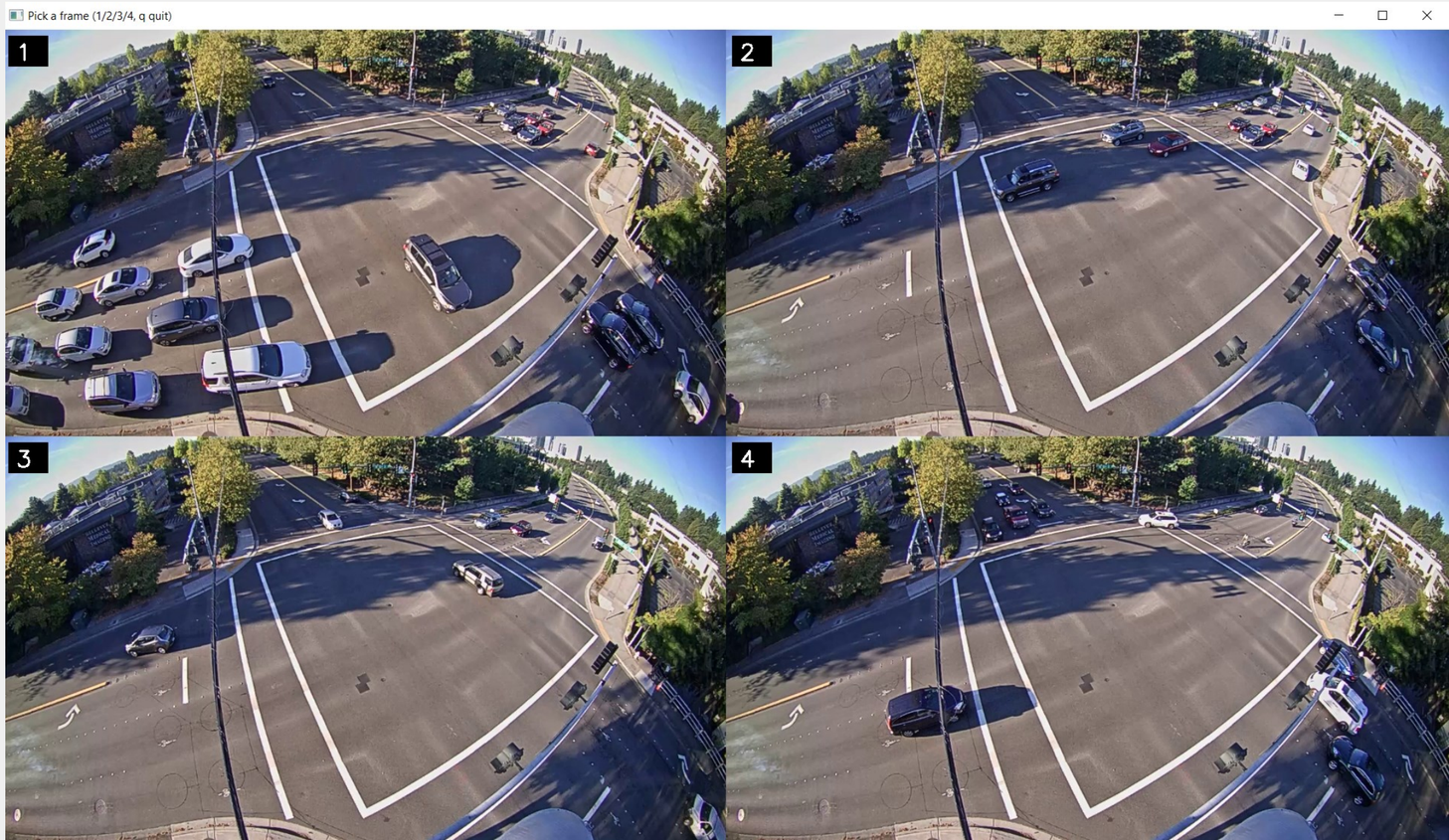


Step 8. Run Application



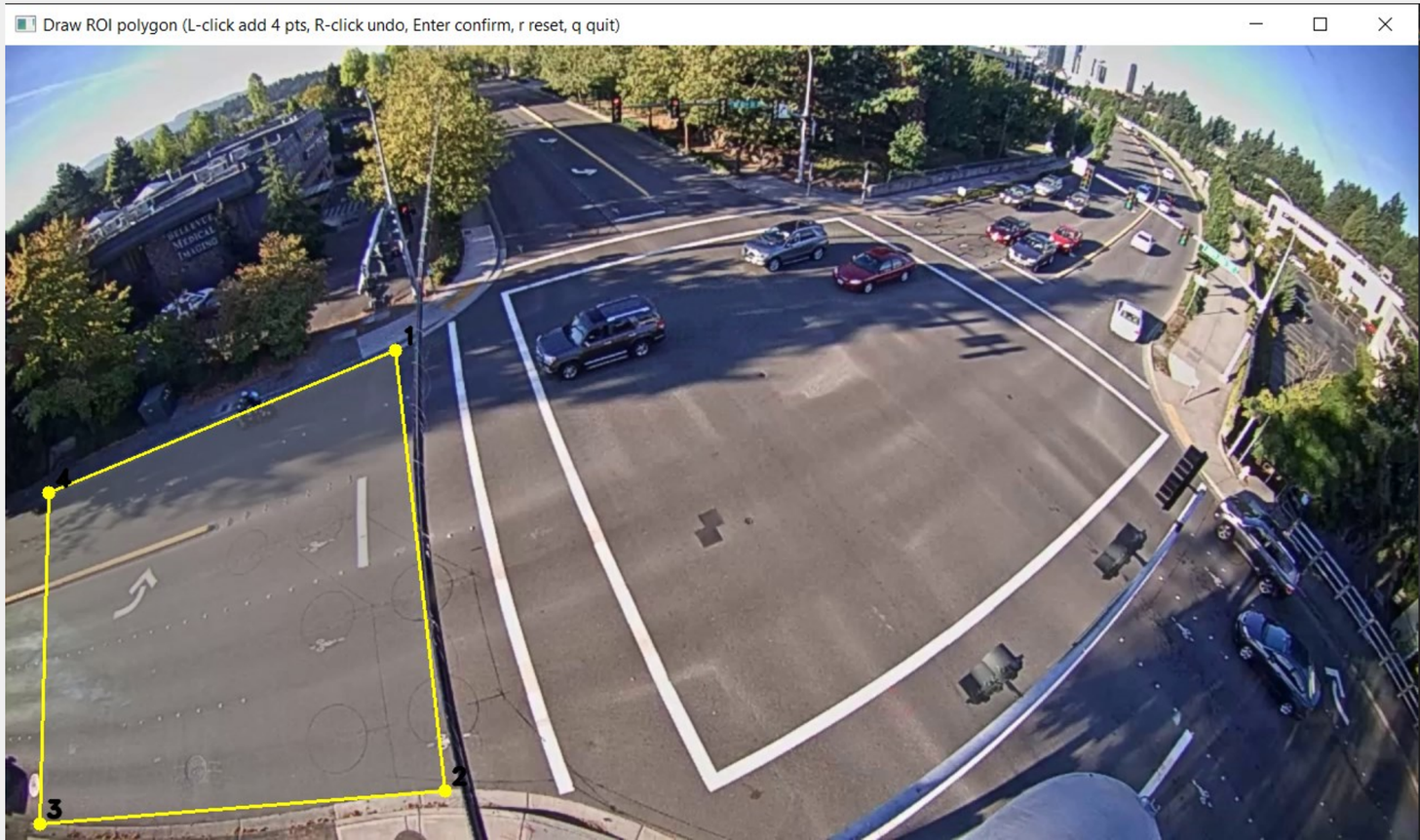
Step 9. Preprocessing

- Select a representative frame by pressing 1 - 4 on the keyboard.
- Prefer a frame with a clear, unobstructed roadway (minimal traffic) for preprocessing/calibration.



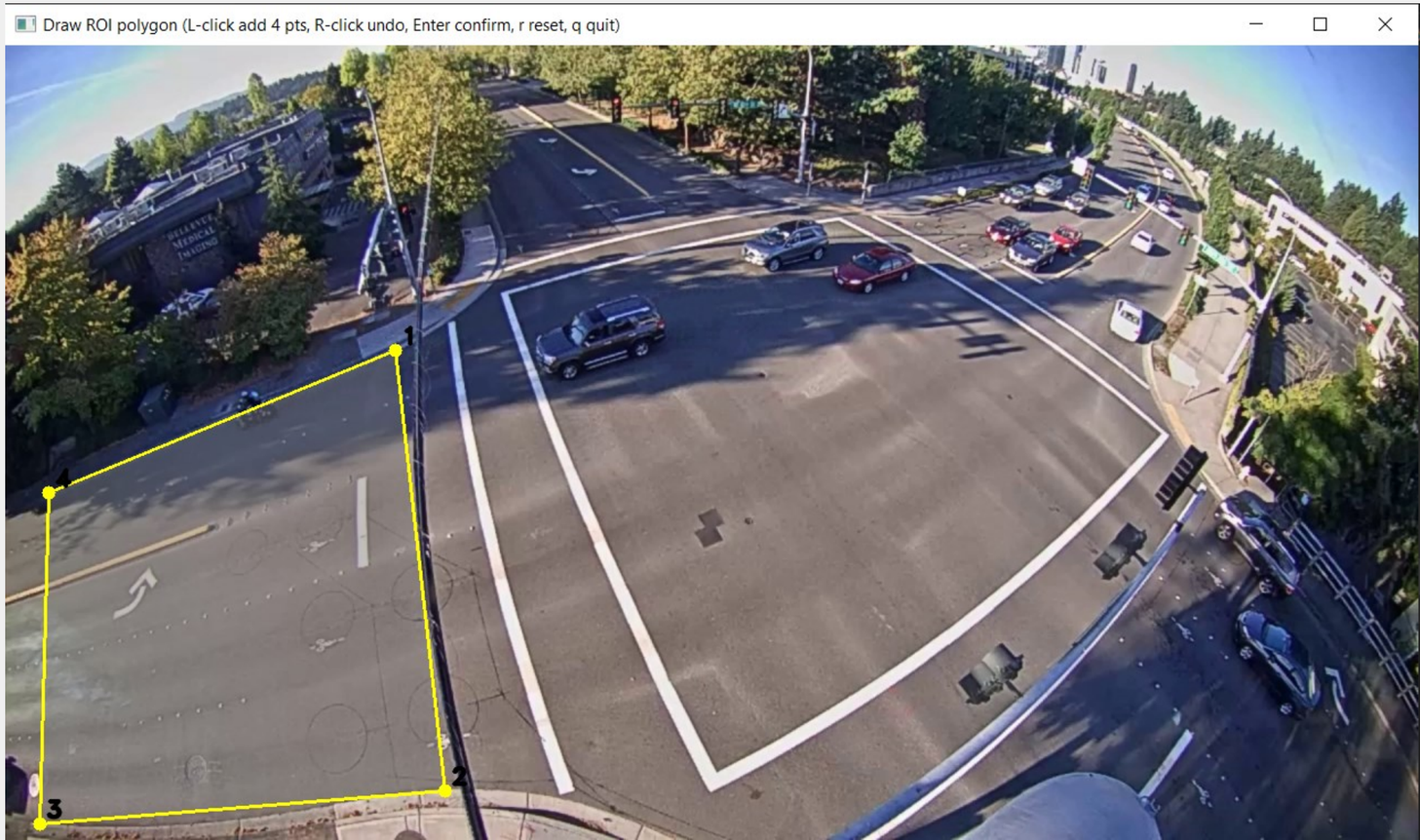
Step 9. Preprocessing (Homography)

- Draw Region of Interest (ROI) polygon
- Left Click → Add 4 points (Clockwise) → See image
- **Click r for reset**
- Once you are done → press “Enter”



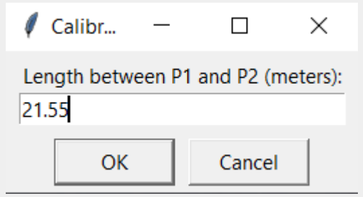
Step 9. Preprocessing

- We need to measure distances between selected points.
- Distance between Point 1 and Point 2
- Distance between Point 2 and Point 3
- See the next slide for the step-by-step process.

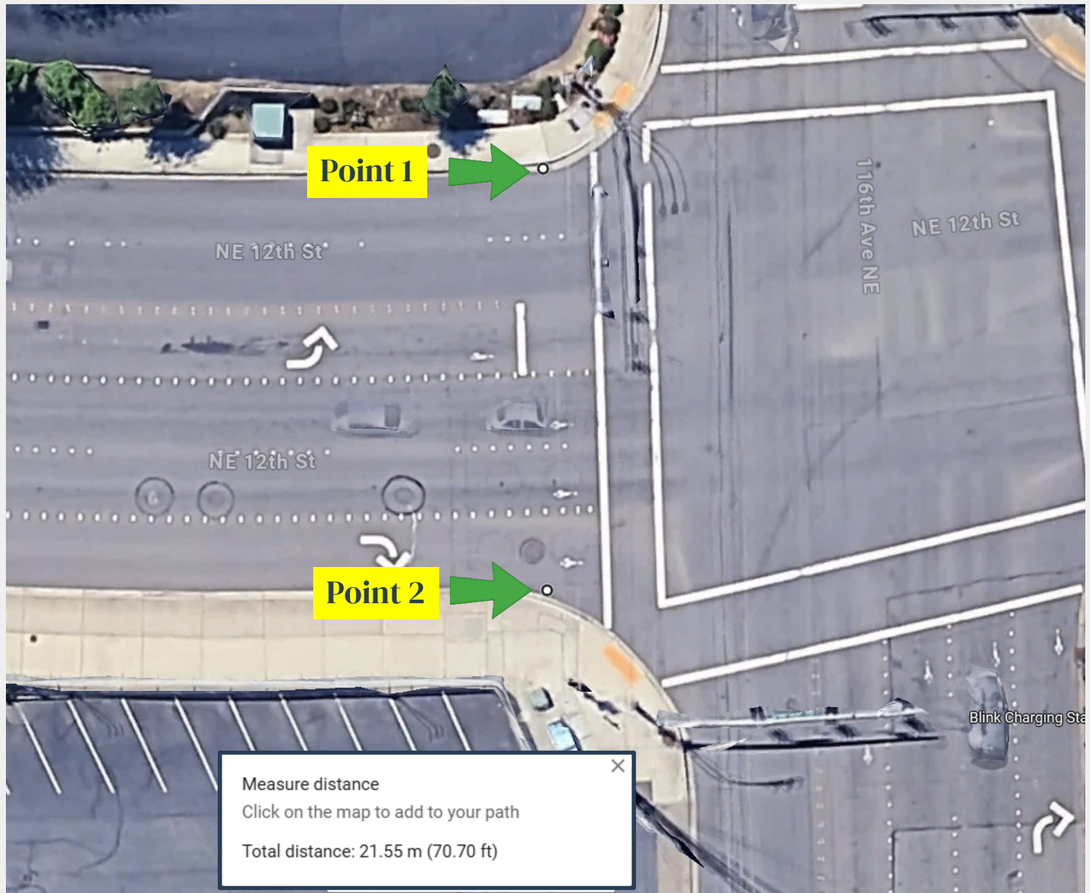


Step 9. Preprocessing

- Open Google Maps and go to: (47.6230411248205, -122.18553244180727)
- Zoom in and rotate the map 180° (to match the image perspective)
- Right-click and select “Measure distance.”
- Place Point 1 and Point 2 in the exact same locations as in the previous slide.
- Enter the measured distance in the box.

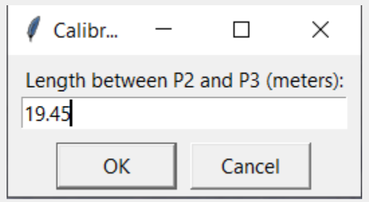


- Click “OK.”
- Right-click and clear the measurement.



Step 9. Preprocessing

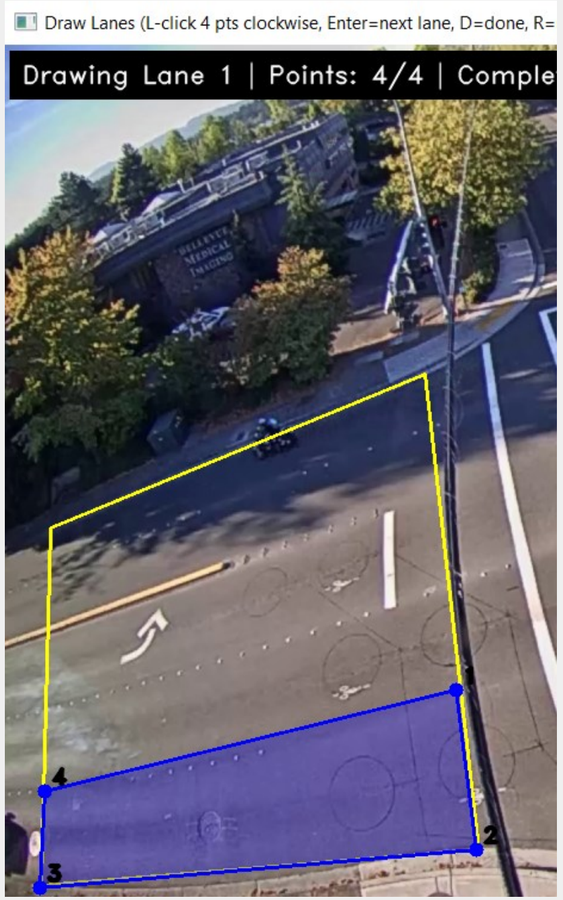
- Right-click → Measure distance
- Add the first point (we can name it Point 2) and the second point (we can name it Point 3) in exactly same location as point 2 and 3 in the two previous slide
- Enter the distance in the box.

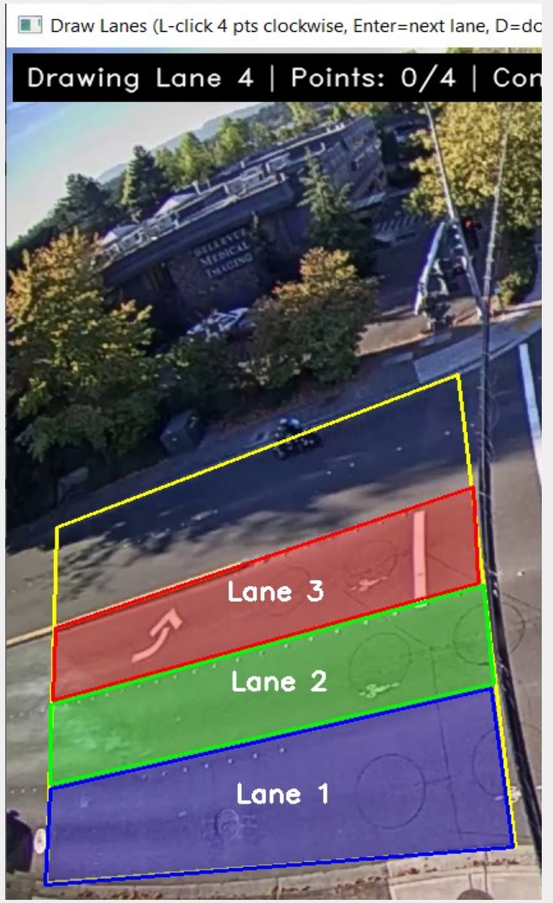


- Click “OK.”
- Right-click and clear the measurement.

Step 9. Preprocessing

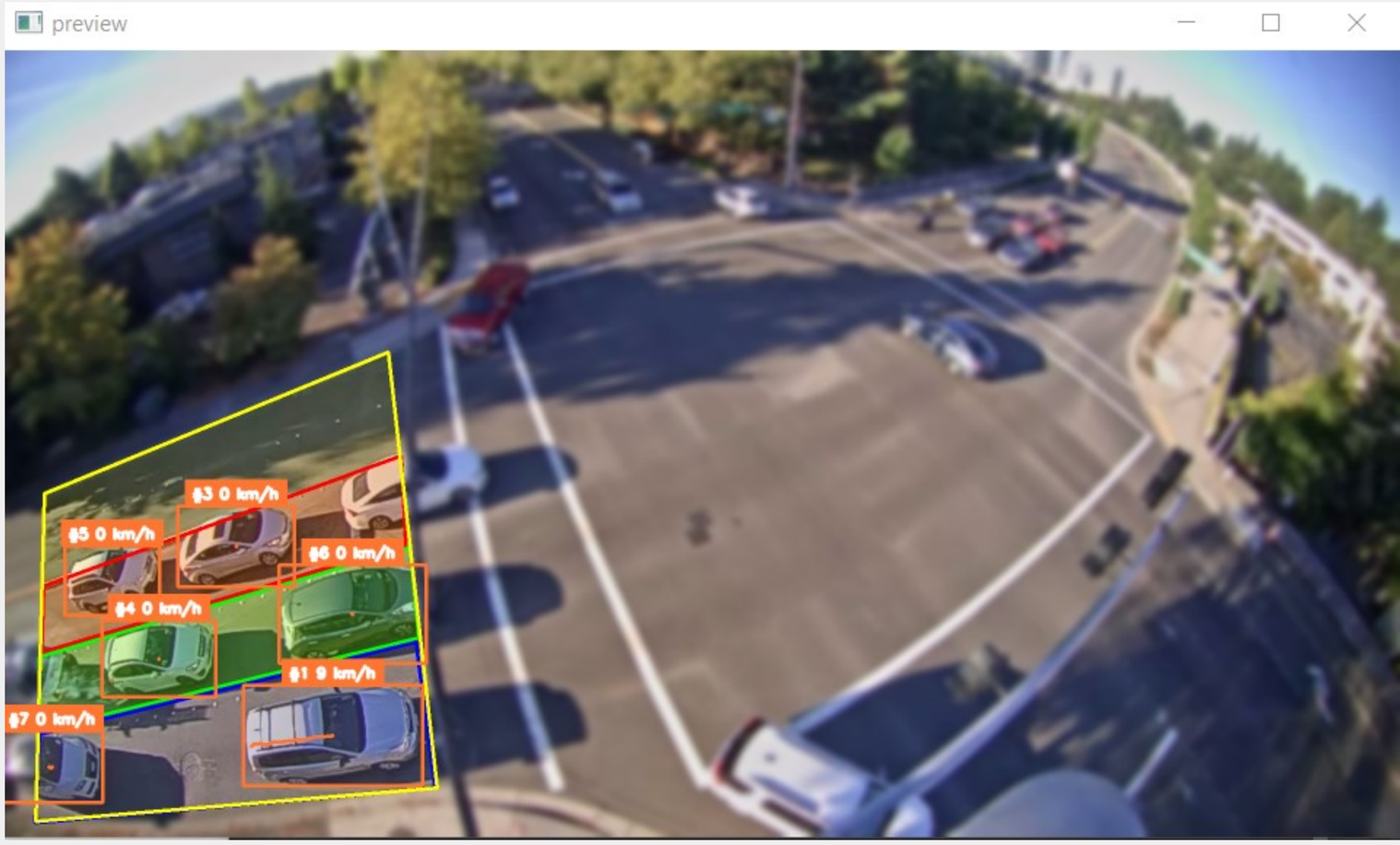
- Draw lane polygons inside the ROI
- For each lane, click 4 corner points (clockwise)
- Press Enter to confirm and move to the next lane
- Make sure lane polygons do not overlap
- Press D when finished





Step 10. Algorithm Running

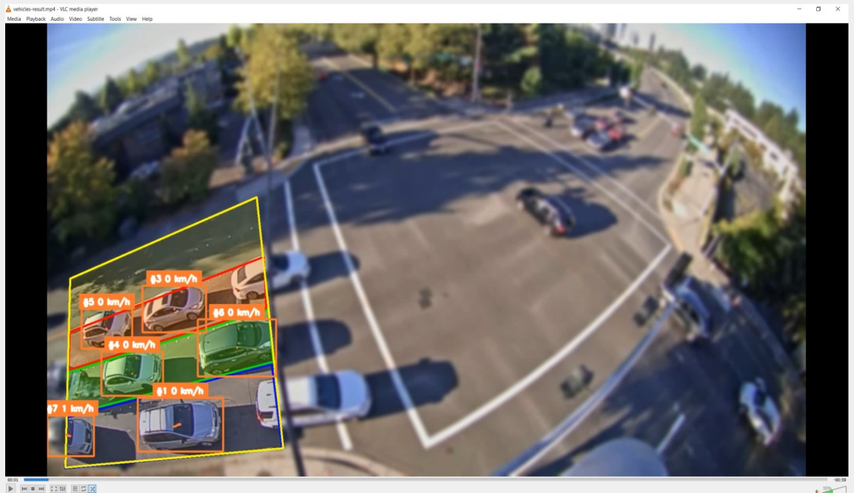
- Algorithm starts running



Step 11. Understanding Output

Open Folder “InputVideo”

1. Open vehicles-result.mp4



2. Open the vehicle_tracks_xy.csv

Frame: Frame number (index of the video frame)

Time: Timestamp of the frame (seconds from start)

Vehicle_id: id of detected vehicle

x_m and y_m: Real-world coordinates in meters

img_x and img_y: Image coordinates in pixels

	A	B	C	D	E	F	G
1	frame	time_s	vehicle_id	x_m	y_m	img_x	img_y
2	0	0	1	2.87	11.431	212.4	640.2
3	0	0	2	8.101	7.827	143.6	556.7
4	0	0	3	13.154	5.098	98.8	485.9
5	0	0	4	12.773	12.147	214.5	453.9
6	0	0	5	8.035	16.914	321	515.5
7	1	0.034	1	2.87	11.431	212.4	640.2
8	1	0.034	2	8.101	7.827	143.6	556.7
9	1	0.034	3	13.154	5.098	98.8	485.9
10	1	0.034	4	12.773	12.147	214.5	453.9
11	1	0.034	5	8.035	16.914	321	515.5
12	2	0.069	1	2.835	11.484	213.5	640.7
13	2	0.069	2	8.102	7.825	143.5	556.7
14	2	0.069	4	12.769	12.147	214.5	453.9
15	2	0.069	5	8.036	16.915	321.1	515.4