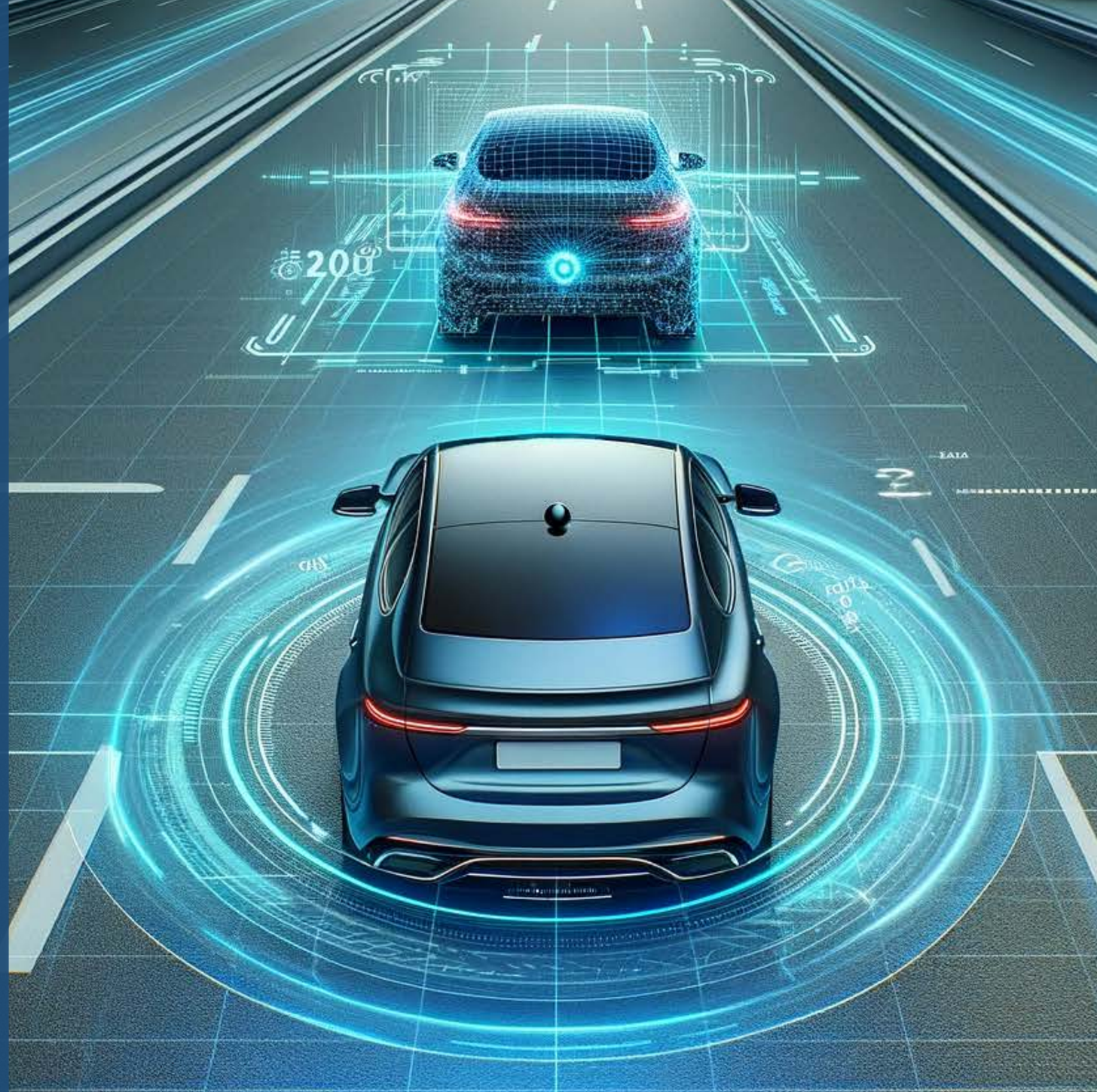
IRACAM

Intelligent Real-time Algorithmic Camera

توسعه الگوریتم تشخیص فاصله
با استفاده از یک دوربین



TEAM

سینا محمدی

دارنده مدال نقره المپیاد هندسه و ریاضیات
کارشناسی مهندسی کامپیوتر
دانشگاه صنعتی شریف

حمیدرضا امیرزاده

کارشناسی ارشد کامپیوتر گرایش هوش مصنوعی
دانشگاه صنعتی شریف

حسن جلالی

رتبه اول کنکور کارشناسی ارشد
کارشناسی ارشد برق گرایش کنترل
دانشگاه تهران

TEAM

جلال نعمت بخش

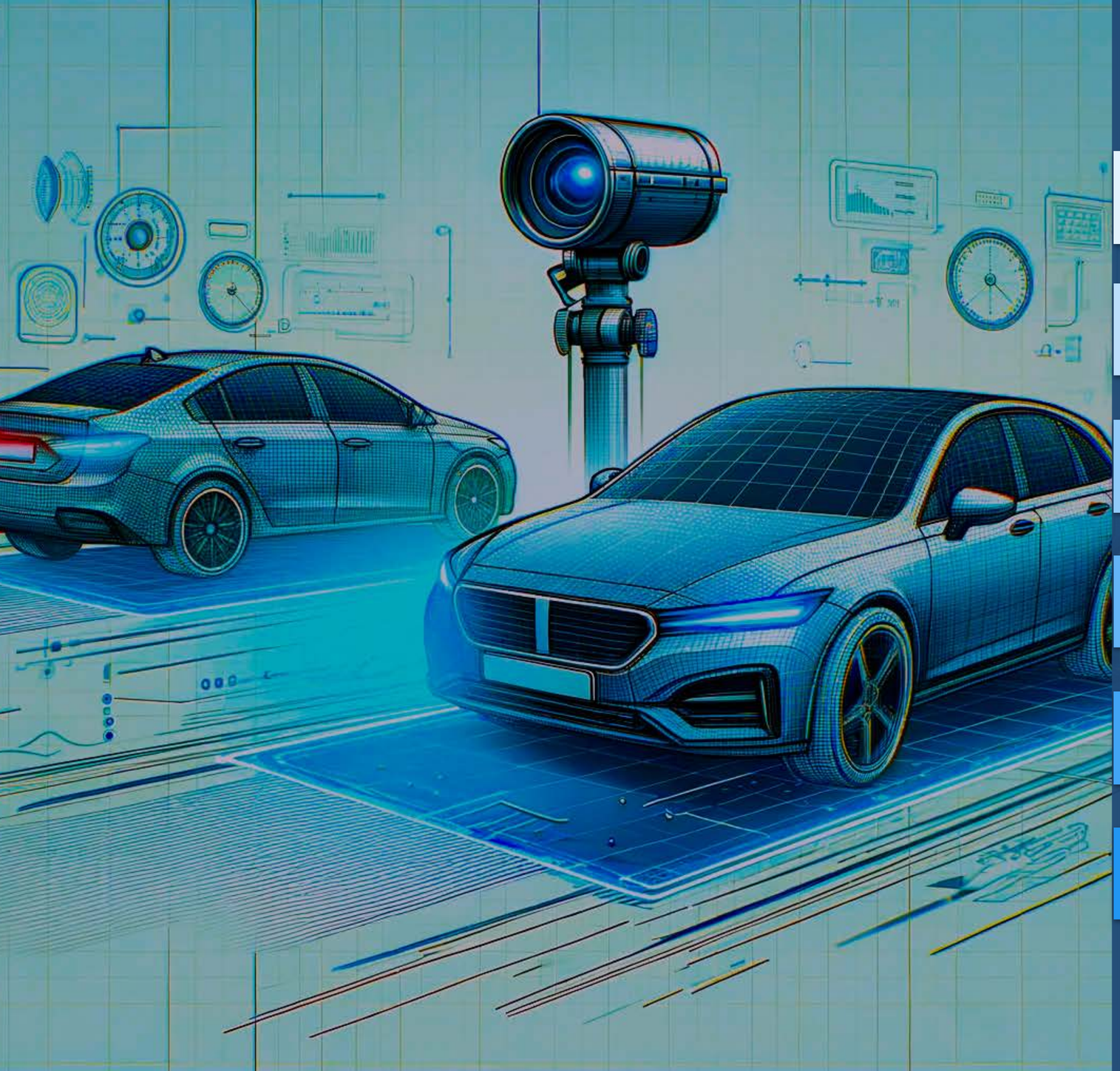
کارشناسی ارشد کامپیوتر گرایش هوش مصنوعی
دانشگاه صنعتی شریف

حسین دماوندی

کارشناسی مهندسی کامپیوتر
دانشگاه صنعتی سجاد

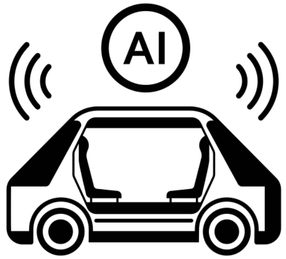
امیر کبیریان

کارشناسی ارشد کامپیوتر گرایش هوش مصنوعی
دانشگاه شهید بهشتی



Opportunities

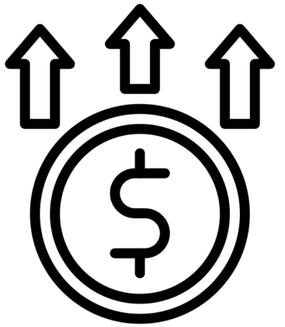
Significant growth in the application of Distance Estimation technologies across various sectors.



Improvements in safety and functionality in ADAS, particularly ACC.

Current Problems

Challenges with processing speed and accuracy, alongside the High computational complexity.



The high cost of distance estimation sensors and environmental sensitivity in existing methods.

Solution

Various methods to tackle this task include Deep Learning, Computer Vision, and mathematical algorithms.

We have implemented 3 approaches using a monocular camera to solve the problem.

Edge AI technologies

Solve some Specific Problems

Innovation

SWOT

S

- Reducing the risk of accidents caused by driver fatigue, distraction, or inattention
- Lowering fuel consumption
- Scalable to higher levels

O

- Reducing traffic on roads
- Enabling driving for individuals with physical disabilities and the elderly
- New job opportunities in the automotive industry.

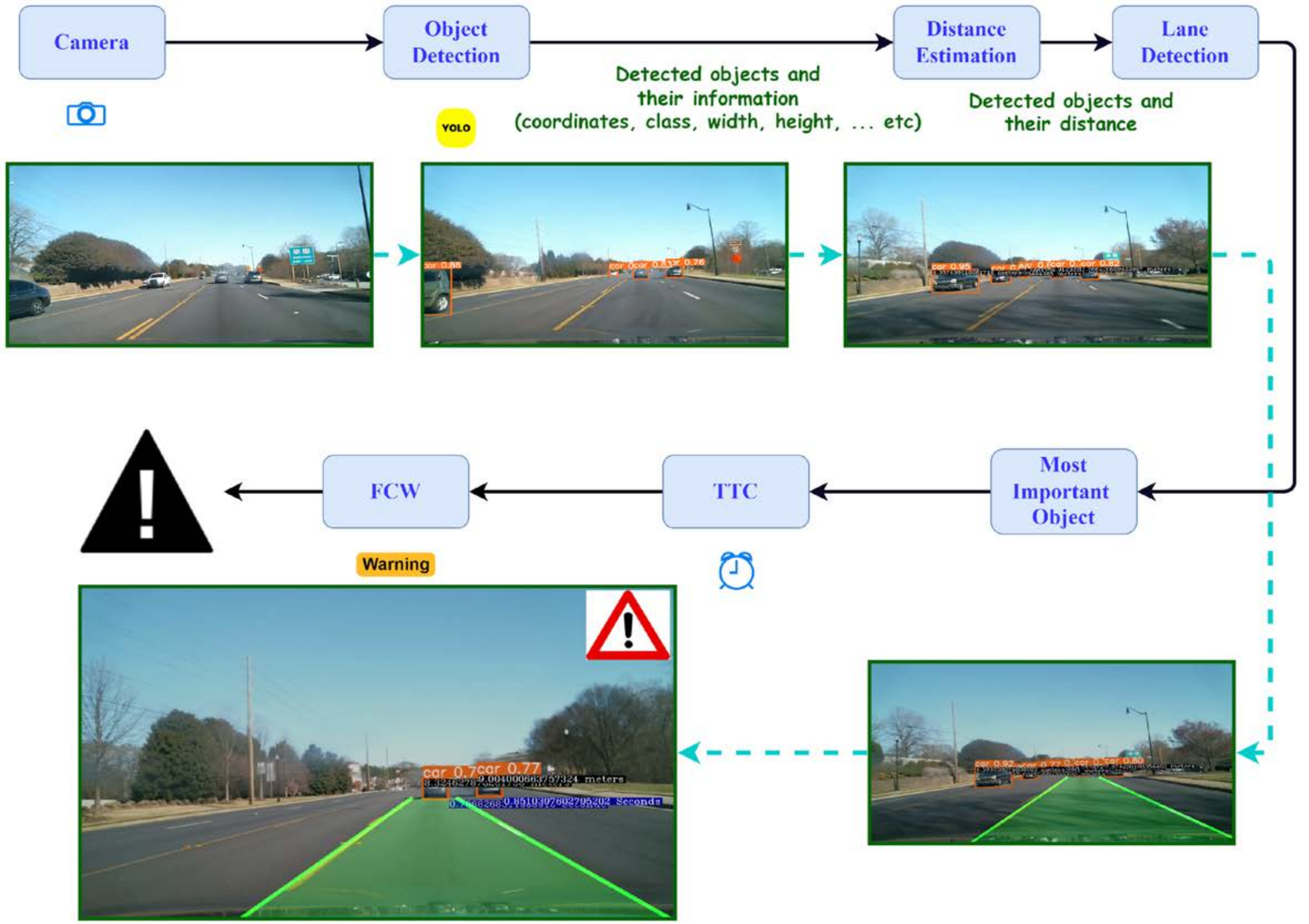
W

The feeling of false security
(requiring constant human supervision)

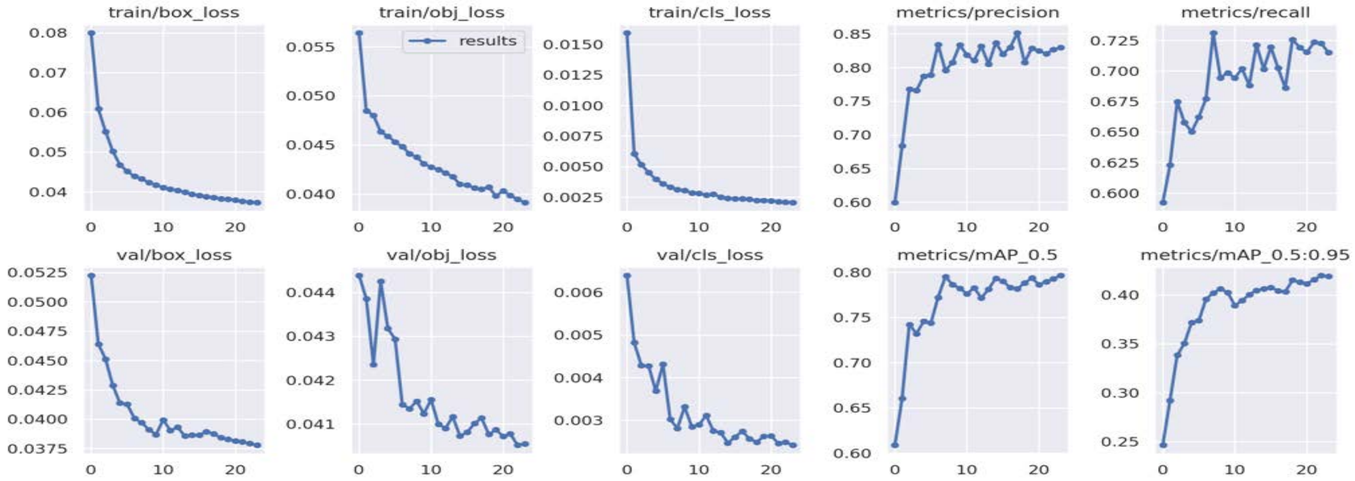
T

- Non-instantaneous processing, especially when data volume increases
- Cybersecurity threats
- Poor performance on roads with weak infrastructure

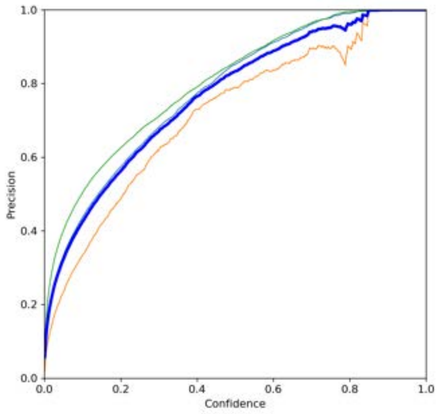
Block Diagram of Approach 1

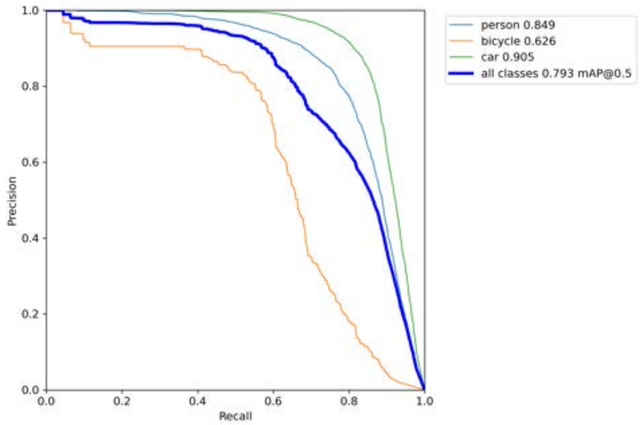


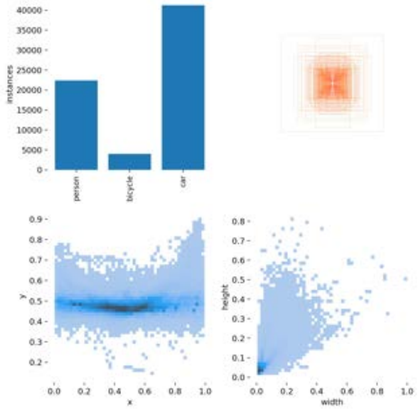
Approach 1

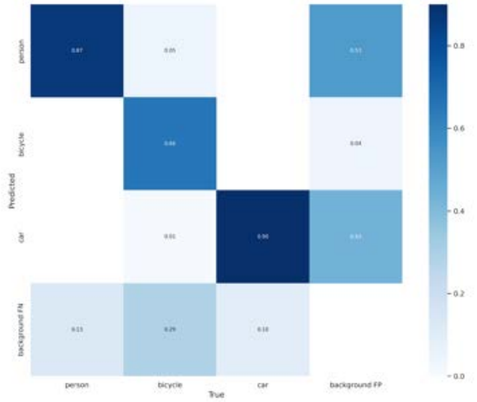


	A	B	C	D	E	F	G	H	I	J	K	L
1	frame	xmin	ymin	xmax	ymax	scaled_xmin	scaled_ymin	scaled_xmax	scaled_ymax	distance		
2	1	500	430	539	465	485.15625	223.9583333	522.9984375	242.1875	10.18347		
3	1	593	419	624	447	575.3953125	218.2291667	605.475	232.8125	17.60549		
4	1	619	414	659	446	600.6234375	215.625	639.4359375	232.2916667	14.91623		
5	1	721	411	769	455	699.5953125	214.0625	746.1703125	236.9791667	9.757074		
6	1	537	423	595	468	521.0578125	220.3125	577.3359375	243.75	7.622859		
7	1	827	405	893	458	802.4484375	210.9375	866.4890625	238.5416667	7.874296		
8	1	372	417	502	499	360.95625	217.1875	487.096875	259.8958333	4.59325		
9	2	492	428	533	467	477.39375	222.9166667	517.1765625	243.2291667	9.090521		
10	2	719	410	769	455	697.6546875	213.5416667	746.1703125	236.9791667	9.670923		
11	2	613	415	656	449	594.8015625	216.1458333	636.525	233.8541667	12.68235		
12	2	530	426	590	469	514.265625	221.875	572.484375	244.2708333	7.842927		
13	2	827	406	892	459	802.4484375	211.4583333	865.51875	239.0625	7.671318		
14	2	352	419	490	501	341.55	218.2291667	475.453125	260.9375	4.539041		
15	3	588	420	619	449	570.54375	218.75	600.6234375	233.8541667	15.80821		
16	3	612	413	654	448	593.83125	215.1041667	634.584375	233.3333333	12.65343		
17	3	721	410	770	455	699.5953125	213.5416667	747.140625	236.9791667	9.678482		
18	3	524	425	586	470	508.44375	221.3541667	568.603125	244.7916667	7.29614		
19	3	828	406	891	459	803.41875	211.4583333	864.5484375	239.0625	7.691926		
20	3	330	417	478	505	320.203125	217.1875	463.809375	263.0208333	4.268023		
21	4	611	414	652	450	592.8609375	215.625	632.64375	234.375	12.16618		
22	4	723	409	771	454	701.5359375	213.0208333	748.1109375	236.4583333	9.912322		
23	4	514	426	584	473	498.740625	221.875	566.6625	246.3541667	6.395777		
24	4	829	407	893	459	804.3890625	211.9791667	866.4890625	239.0625	7.766239		
25	4	305	420	470	510	295.9453125	218.75	456.046875	265.625	4.067436		
26	5	579	422	612	452	561.8109375	219.7916667	593.83125	235.4166667	13.40512		
27	5	724	411	769	455	702.50625	214.0625	746.1703125	236.9791667	9.788103		









Output of Object Detection

•detect: weights=['training-results/weights/best.pt'], source=videos/vidd.mp4, data=data\coco128.yaml, imgsz=[640, 640], conf_thres=0.4, iou_thres=0.45, max_det=1000, device=, view_img=

•False, save_txt=True, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs\detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False

•YOLOv5 2023-11-11 Python-3.10.5 torch-1.12.0 CUDA:0 (NVIDIA GeForce RTX 3060 Laptop GPU, 6144MiB)

•

•Fusing layers...

•YOLOv5s summary: 213 layers, 7018216 parameters, 0 gradients

•video 1/1 (1/547) D:\Irascience\rahisho\object_detection-PLUS-distance_estimation-v1\object-detector\videos\vidd.mp4: 384x640 7 cars, Done. (0.012s)

•video 1/1 (2/547) D:\Irascience\rahisho\object_detection-PLUS-distance_estimation-v1\object-detector\videos\vidd.mp4: 384x640 6 cars, Done. (0.004s)

•video 1/1 (3/547) D:\Irascience\rahisho\object_detection-PLUS-distance_estimation-v1\object-detector\videos\vidd.mp4: 384x640 6 cars, Done. (0.008s)

•video 1/1 (4/547) D:\Irascience\rahisho\object_detection-PLUS-distance_estimation-v1\object-detector\videos\vidd.mp4: 384x640 5 cars, Done. (0.010s)

•video 1/1 (5/547) D:\Irascience\rahisho\object_detection-PLUS-distance_estimation-v1\object-detector\videos\vidd.mp4: 384x640 6 cars, Done. (0.005s)

•video 1/1 (6/547) D:\Irascience\rahisho\object_detection-PLUS-distance_estimation-v1\object-detector\videos\vidd.mp4: 384x640 6 cars, Done. (0.010s)

•video 1/1 (7/547) D:\Irascience\rahisho\object_detection-PLUS-distance_estimation-v1\object-detector\videos\vidd.mp4: 384x640 6 cars, Done. (0.010s)

Speed: 1.3ms pre-process, 31.0ms inference, 1.4ms NMS per image at shape (1, 3, 640, 640)

pre-process	inference	shape	Mean-cars
1.3ms	31ms	1,3,640,640	5

Output of Distance Estimator

```
Loaded model from disk
77/77 [==============================] - 0s 647us/step
process time 0.37499213218688965
```

Total time for distance estimator	frame	Time per frame
0.375	547	0.00006s

Demo


car 0.92
car 0.77
0.car 0.
car 0.80
4.593250274658203 meters
7.874296188354492 meters
1.61.
0.720497024983224 Seconds
0.744262396554341 Seconds
0.434144638436503 14 Seconds

Depth Estimation DL Approach

We need a NN that:

- Be fast at inference time (real-time)
- Accurate enough
- Low resource
- Trained on Monocular camera

Suggested Models:

- Monodepth2
- Fast-depth -> even faster (**37ms** on cpu and **5.6ms** on GPU)

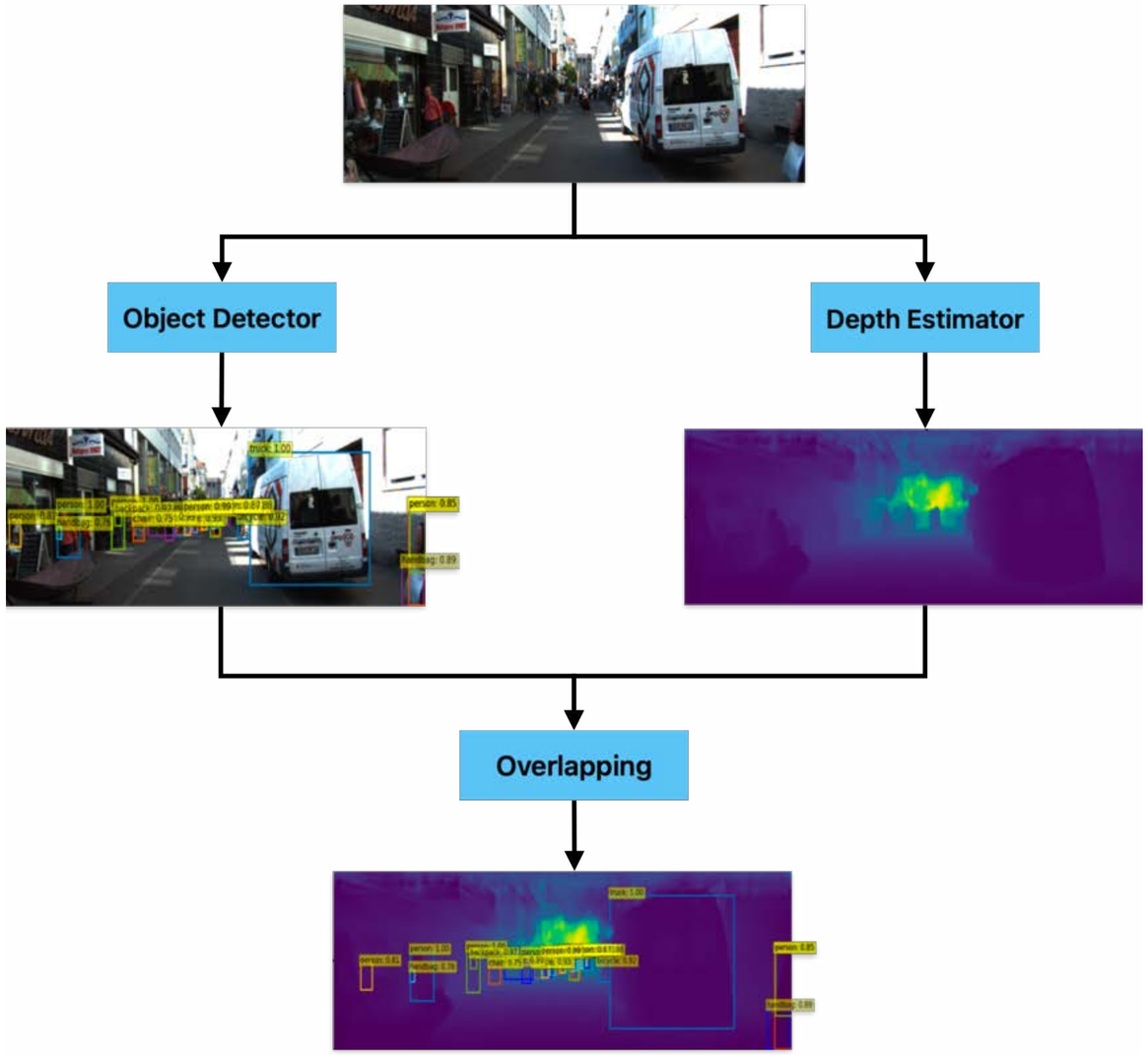
Demo

Real-time test of Monodepth2:

Rethinking the common methods

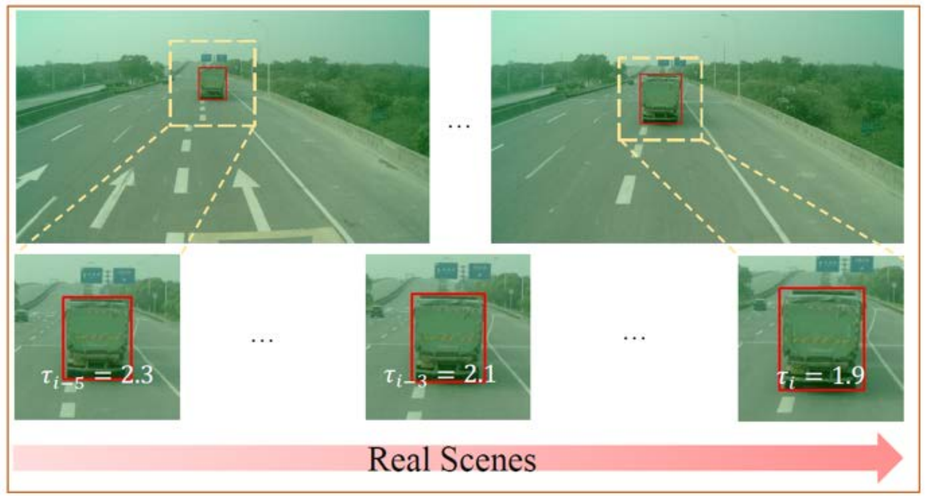
- Is it the only solution?
- Can we make this whole process end2end?
- We need a TTC based dataset

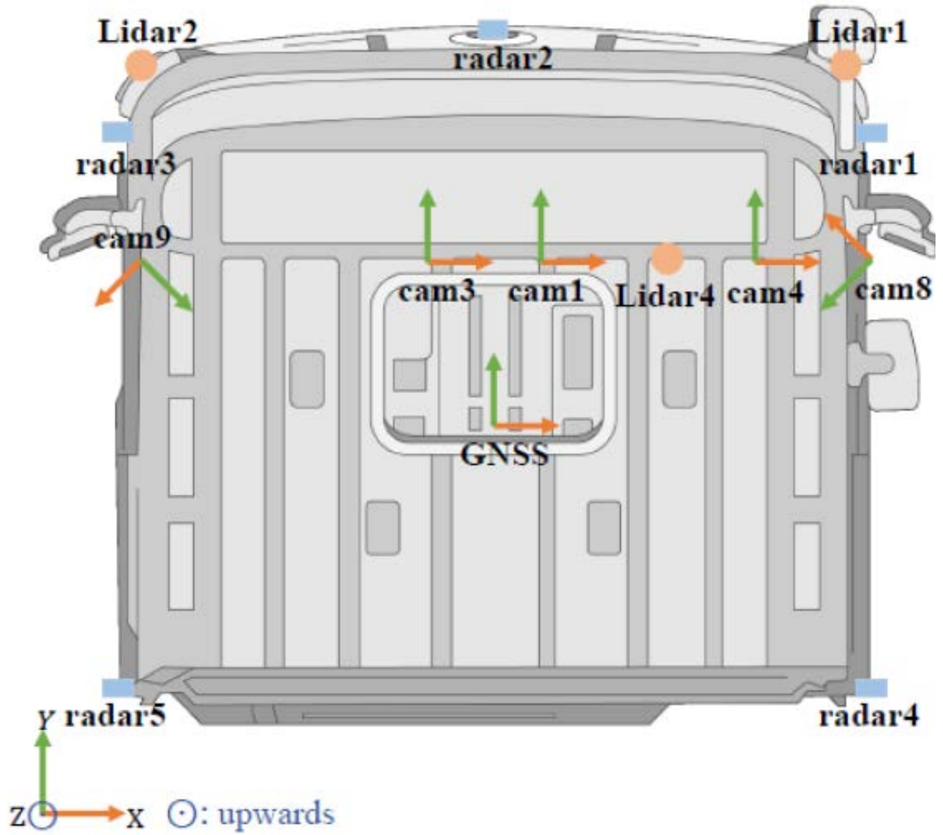
Approach 3



Introducing TSTTC dataset

- 200000 sequences of highway drivings
- 10 Hz of sampling
- Labeled by the TTC of a specific vehicle





Cameras horizontal field of view

Camera	1	3	4	8	9
HFOV	±63.2°	±40.4°	±18.4°	±63.2°	±63.2°

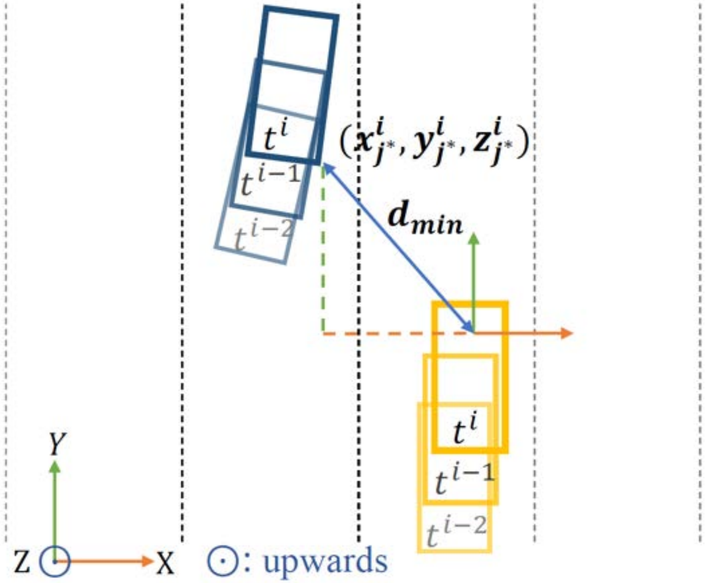
Data annotation

- Depth estimation(y^i):
  Use 3D object detection using LiDAR data to obtain depth.

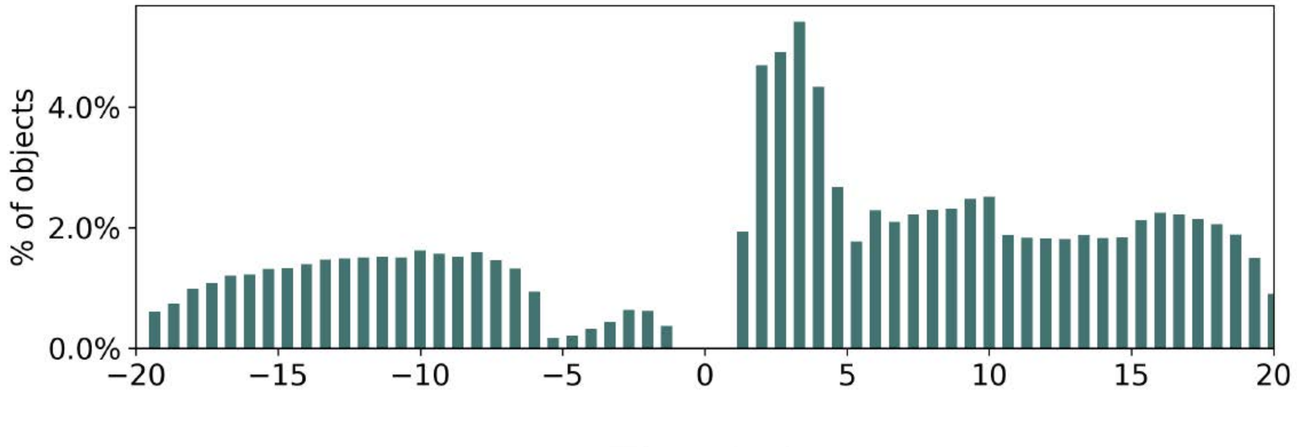
$$j^* = \underset{j \in \{1,2,\ldots,8\}}{\arg\min} \left(\sqrt{(x_j^i - 0)^2 + (y_j^i - 0)^2 + (z_j^i - 0)^2}\right), y^i = y_{j^*}^i,$$

- Velocity estimation(v^i):
  Fit the depth of the vehicle in past q frames to obtain the relative velocity by RANSAC algorithm.

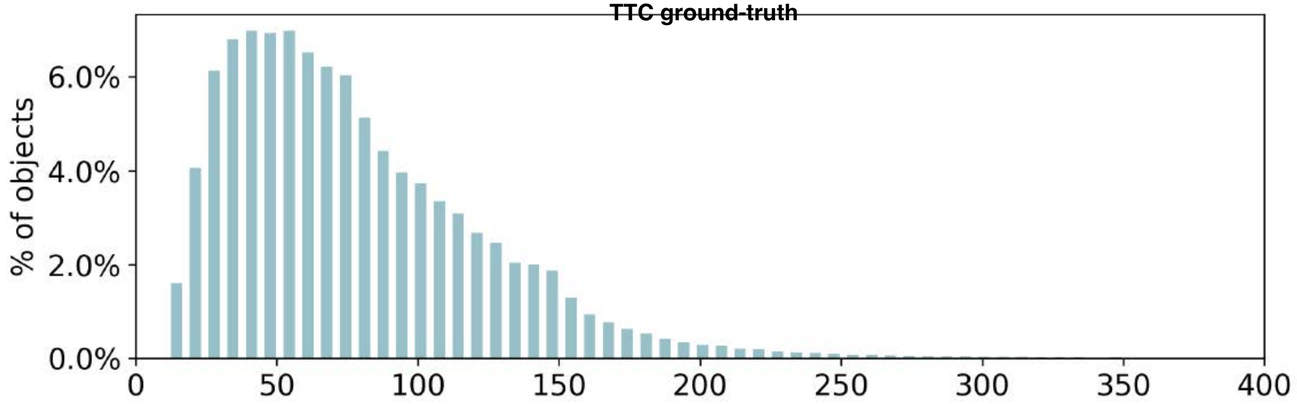
- Ground-truth TTC(τ^i): $\tau^i = y^i / v^i$



Dataset statistics



TTC ground-truth



Relative depth

Okay, TTC labeled dataset is ready, Now let's tackle our own problem!

- **Idea:** Can we use scale ratio of the detection boxes as a lightweight heuristic for TTC estimation?!

- Based on [2] we can obtain the image size using:

$$s = fS/y$$

f : focal length of the camera
S : object size
y: distance to the object

- For an object without rotation:

$$\tau = \frac{t_1 - t_0}{1 - \frac{s(t_0)}{s(t_1)}} = \frac{t_1 - t_0}{1 - \alpha}$$

t_0, t_1 : two consecutive frame
α : scale ratio

- Interesting takeaway:

scale ratio estimation → TTC estimation

Evaluation metrics

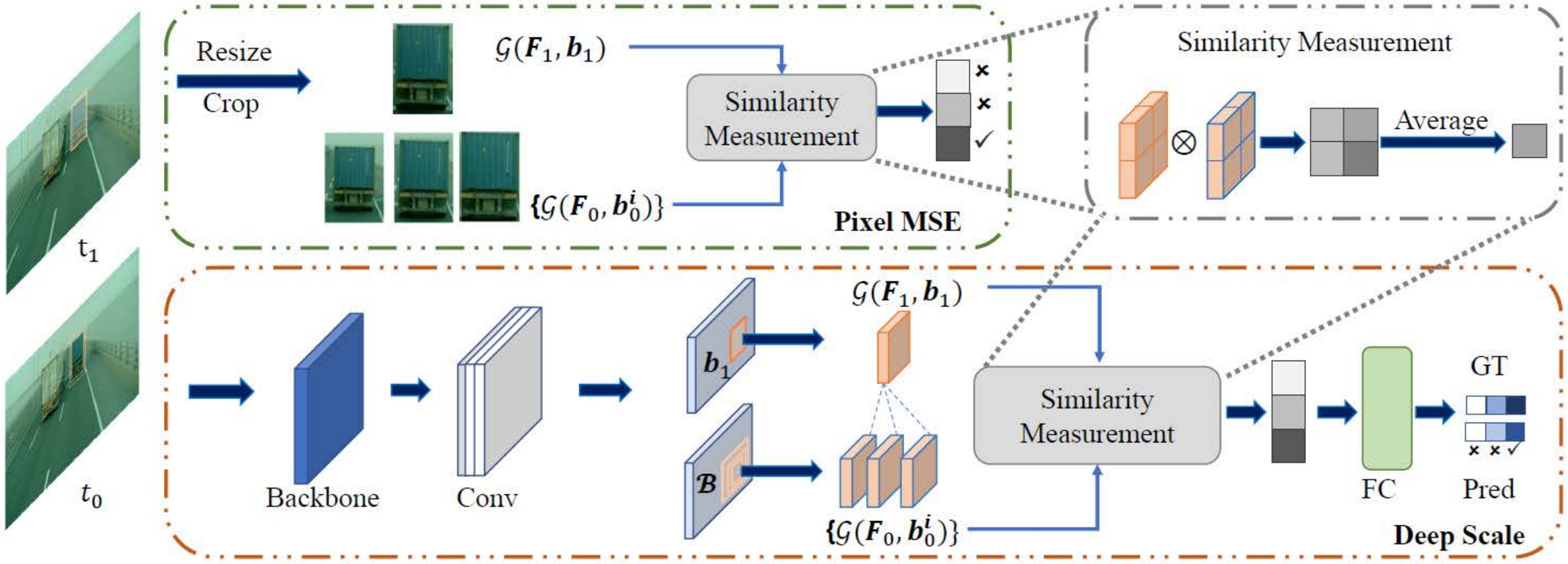
- **Motion-in-Depth (MiD) error**

$$\mathrm{MiD} = ||\log(\alpha) - \log(\hat{\alpha})||_1 \times 10^4$$

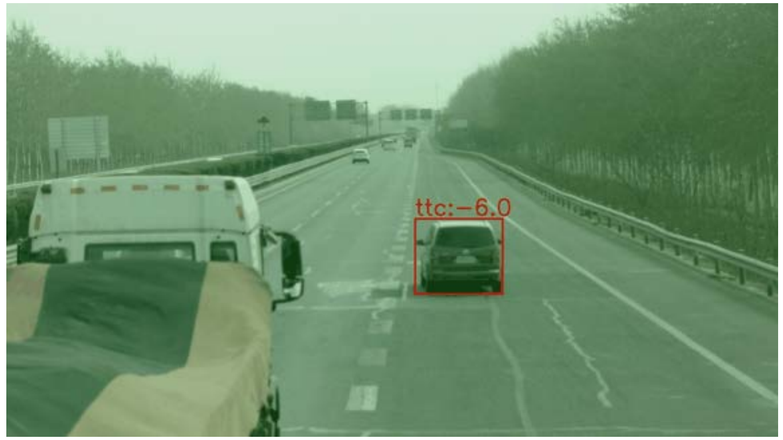
- **Relative TTC Error (RTE)**

$$\mathrm{RTE} = ||\frac{\tau - \hat{\tau}}{\tau}||_1 \times 100\%$$

Proposed approaches



Demo



Experimental evaluation

- Scale ratio estimation in feature space is more robust than image space.

روش	MID	MiD_c	MiD_s	MiD_l	MiD_n	RTE	RTE_c	RTE_s	RTE_l	RTE_n
Pixel MSE	۴۱.۰	۵۷.۴	۳۶.۵	۳۲.۵	۴۸.۵	۲۹.۹	۱۱.۳	۱۳.۰	۳۱.۰	۴۴.۳
Deep Scale	۱۴.۴	۲۷.۱	۱۶.۴	۱۰.۹	۱۳.۵	۱۲.۱	۶.۳	۸.۷	۱۳.۰	۱۴.۸

Limitations

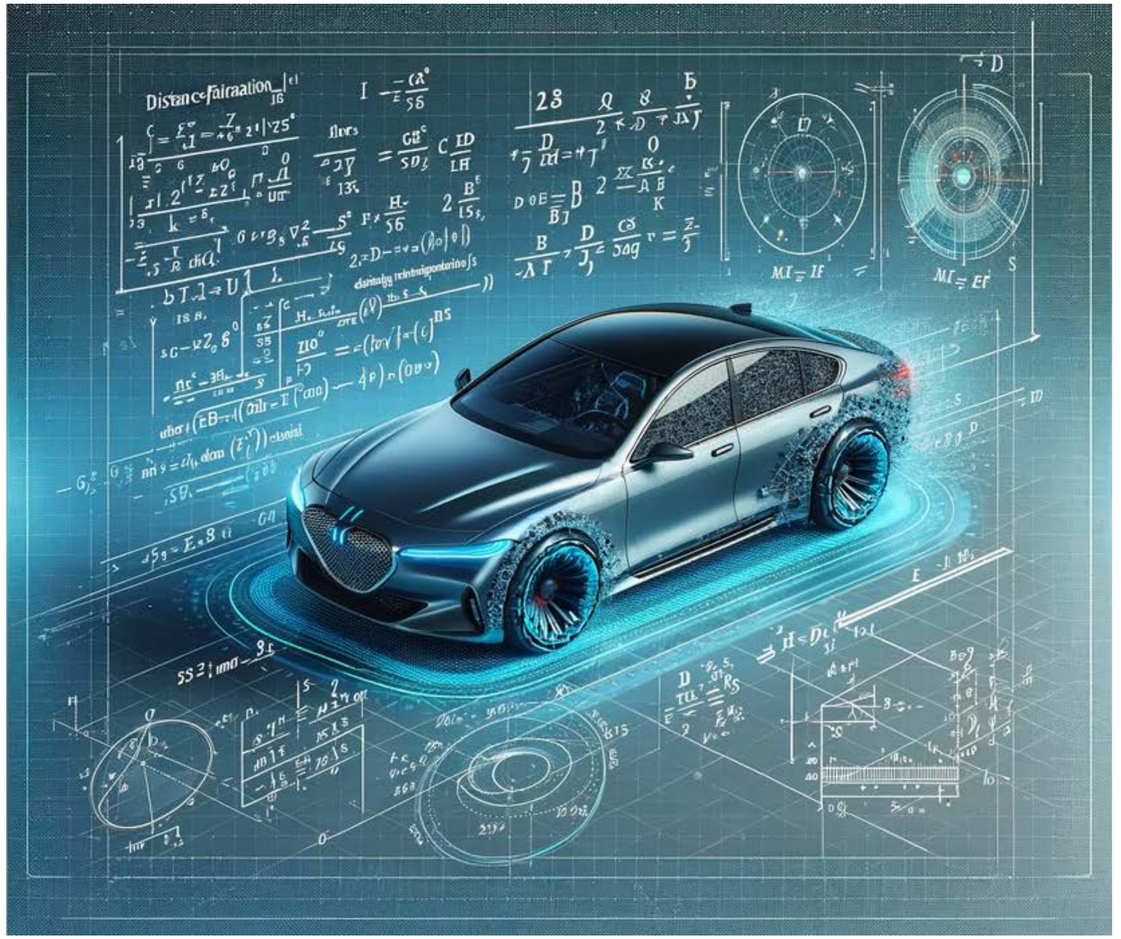
- Used simplified assumptions like objects exhibit frontal-parallel characteristics.
- There is no lane detection, but it can be simply added.

Mathematical Innovation

In the mathematical approach, two subjects are very important:

1. The approach focuses on using image data without considering real-world distances.

2. The method addresses more than just constant speed scenarios, assuming variable speeds between vehicles.



So we define T_m:

$$T_m \rightarrow \left(\text{momentary } TT_{\mathbb{C}}\right) T_m = \frac{-z}{V}$$

- a lot of information from just point z_0:
- Advancing the topic to a more advanced state (constant acceleration)

$$z = z_0 + v_0\Delta t + \frac{1}{2}a(\Delta t)^2 \Rightarrow T = \frac{-V_0 + \sqrt{V_0^2 - 2z_0a}}{a}$$

(T is Time to collision)

- We will calculate the value of V_0:

$$z(t) = \frac{fW}{x_r(t) - x_l(t)} = \frac{\big(x_r(0) - x_l(0)\big)z(0)}{x_r(t) - x_l(t)}$$

- Calculate z_0 for the recent equation:

$$\frac{fW}{\big(x_r(0) - x_l(0)\big)}$$

Challenge for Calculate W

- Calculating the left and right coordinates of an image and its benefits in line detection

$$\begin{cases} X_l(t) = \dfrac{z(t)x_l(t)}{f} = \dfrac{x_l(t)z(0)}{f}\dfrac{x_r(0)-x_l(0)}{x_r(t)-x_l(t)} \\ X_r(t) = \dfrac{z(t)x_r(t)}{f} = \dfrac{x_l(t)z(0)}{f}\dfrac{x_r(0)-x_l(0)}{x_r(t)-x_l(t)} \end{cases}$$

Innovation

- instantaneous speed calculation

$$V = \frac{D_2 - D_1}{\Delta t} = \frac{f_W \left[\frac{1}{W_1} - \frac{1}{W_2}\right]}{\Delta t}$$

- so:

$$V = \frac{\left(x_r(0) - x_l(0)\right) z(0) \left[\frac{1}{W_1} - \frac{1}{W_2}\right]}{\Delta t}$$

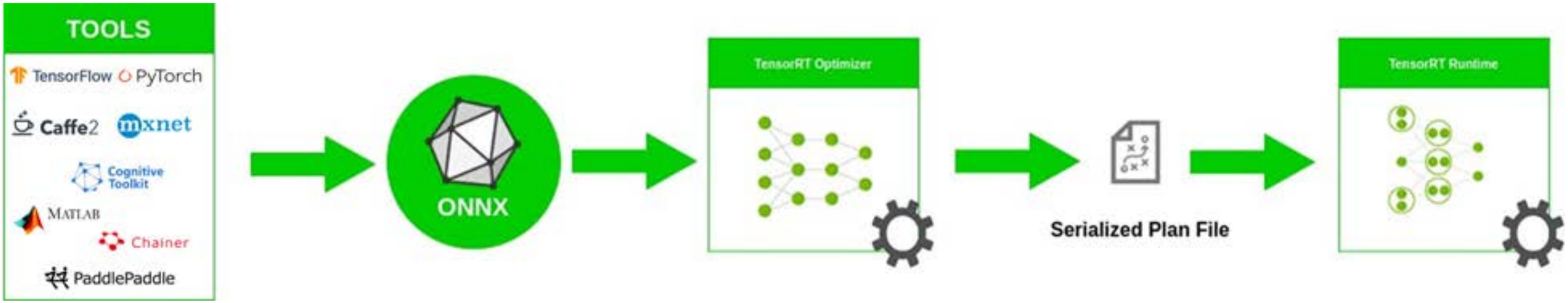
- Calculating instantaneous acceleration in the generalized state
- Calculating instantaneous acceleration in the constant acceleration.

$$a = \frac{V_2^2 - V_1^2}{2\Delta x}$$

$$a = \frac{V_{2,\Delta t}^2 - V_{1,\Delta t}^2}{2\Delta x} = \frac{V_{2,\Delta t}^2 - V_{1,\Delta t}^2}{2(D_2 - D_1)}$$

$$T = \frac{-V_0 + \sqrt{V_0^2 - 2z_0 a}}{a} = \frac{-V_0 + \sqrt{V_0^2 - 2z_0 \frac{V_{2,\Delta t}^2 - V_{1,\Delta t}^2}{2(D_2 - D_1)}}}{\frac{V_{2,\Delta t}^2 - V_{1,\Delta t}^2}{2(D_2 - D_1)}}$$

TensorRT and ONNX



TensorRT and Pytorch Comparison

	Framework	mAP@0.5:0.95	Inference time (ms)
0	PyTorch	0.462296	9.159939
1	TorchScript	0.462296	6.607546
2	ONNX	0.462296	12.698026
3	OpenVINO	NaN	NaN
4	TensorRT	0.462280	1.725197
5	CoreML	NaN	NaN
6	TensorFlow SavedModel	0.462296	20.273019
7	TensorFlow GraphDef	0.462296	20.212173

Selecting Idea & Competitive Analysis - How?

After exploring **what** we have done about selecting idea & and analysis, let's see **how.**

- Problem & Challenge - Summary
  - **Problem:** drivers face vision problems while driving in **unclear vision** (e.g. foggy weather and darkness), which leads to many accidents.
  - **Challenge:** How to reduce car accidents due to unclear vision during driving? **Thermal Camera**

Demo


AG34
car 0.50
42.0 meters
car 0.87
11.0 meters
32.0 meters
car 0.43
37.0 meters
12.0 meters
17.0 meters
6.0 meters
6.0 meters
7.0 m

Conclusion

In conclusion, our team has successfully implemented three distinct approaches utilizing a monocular camera, yielding promising results in distance estimation. Our innovative strides have not only embraced mathematical advancements but have also integrated efficient methodologies such as ONNX and TensorRT, significantly enhancing operational speed. Upon reviewing commonly employed techniques, we have proposed a method that stands out for its improved accuracy, setting a new benchmark for future development in this field.

Future Works

- Apply Incremental Learning for better accuracy and processing speed.
- Explore a range of diverse approaches, possibly integrating hybrid models combining deep learning, computer vision, and mathematical algorithms.
- Focus on real-world implementation
- …