# Real Time Collision Warning System Using Monocular Vision and Object Tracking

See presentation video:

https://northeastern-my.sharepoint.com/:v:/g/personal/guo_yunyu_northeastern_edu/EcWEl74IVOdCkNjEBaDCcuUBgxIOC83npvu9aFdVGAfb_w?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=RWjOor

Yunyu Guo
*Koury College of Computer Science*

*Northeastern University*

Oakland, US
guo.yunyu@northeastern.edu

***Abstract*— Monocular cameras offer a cost-effective solution for vehicle perception, but estimating Time-to-Collision (TTC) accurately remains challenging. This project develops a real-time system capable of detecting vehicles, estimating their depth, tracking them over time, and calculating TTC using a single camera feed. The pipeline employs YOLOv5 for detection, BoT-SORT for tracking, and Monodepth2 for depth prediction. It visualizes detections, depth maps, and TTC warnings on driving videos. Performance is benchmarked on the KITTI depth completion dataset, measuring accuracy for both depth estimation (d1 > 0.95) and TTC prediction (median relative error 61%).**

***Keywords—time-to-collision, real-time, vehicle perception, vehicles detection, tracking***

## I. Introduction

Autonomous vehicles rely heavily on accurate perception of the surrounding environment to ensure safety. A critical component of this perception is the estimation of Time-to-Collision (TTC) with other objects, particularly leading vehicles. This project presents a system that integrates object detection (YOLOv5), tracking (BoT-SORT), and monocular depth estimation (Monodepth2) to calculate TTC for objects in driving scenes. The pipeline processes video input, identifies and tracks objects, estimates their depth frame-by-frame, and computes TTC based on the rate of depth change. The system's performance is evaluated using the KITTI depth completion dataset, assessing both the accuracy of the underlying depth predictions and the final TTC estimates against ground truth data. This provides a practical assessment of using deep learning components for monocular collision warning.

## II. Related Work

### A. Monocular Depth Estimation

Monodepth2 [1] presents a self-supervised approach for monocular depth estimation that learns from monocular videos without ground truth depth supervision. The method employs minimum reprojection loss, auto-masking of stationary pixels, and multi-scale estimation to produce accurate depth maps. In the project, it leverages Monodepth2's ResNet18-based encoder-decoder architecture to generate per-pixel depth maps from single RGB images, which serve as the foundation for the TTC calculations..

### B. Time-to-Collision Estimation

Kopf et al. [2] introduce a framework for TTC and collision risk estimation using local scale and motion cues from monocular vision. The paper establishes mathematical relationships between optical flow, looming (apparent size changes), and TTC, demonstrating that robust TTC estimates are possible without explicit depth reconstruction. While the project's approach differs by using depth maps, this work provides theoretical foundations for the TTC calculation formula TTC = depth / velocity, where velocity is derived from temporal depth changes.

### C. Multi-Object Tracking

BoT-SORT [3] builds on previous tracking-by-detection frameworks by combining motion prediction via Kalman filtering with re-identification (ReID) features. It introduces camera motion compensation and interaction modeling to achieve state-of-the-art tracking performance. The project pipeline utilizes BoT-SORT with the osnet_x0_25_msmt17 ReID model to maintain consistent object identities across frames, which is critical for accumulating depth histories and computing TTC for each object.

### D. Object Detection

YOLOv3 [4] presents an incremental improvement to the YOLO (You Only Look Once) detection framework, featuring a deeper feature extractor network and multi-scale predictions. Its efficient design allows real-time detection with competitive accuracy. Building upon this, YOLO9000 [5] (also known as YOLOv2) expands the detection capacity to over 9000 object categories through hierarchical classification. YOLOv5 builds upon techniques in both YOLOv3 and YOLOv4 [6]. The project's system employs YOLOv5 [7], a successor to these models, to rapidly detect vehicles and other objects in each video frame, providing the bounding boxes required for tracking and depth extraction.

### E. Simple Online Tracking

SORT [8] introduces a simple yet effective tracking-by-detection framework that relies on the Hungarian algorithm for detection-to-track association and Kalman filtering for motion prediction. Its lightweight design prioritizes

computational efficiency while maintaining competitive tracking performance. BoT-SORT, used in the project system, extends SORT's design with additional features like appearance embeddings, making it more robust in challenging scenarios.

## III. PROPOSED METHOD

### A. Overall System

The system processes video frames to estimate Time-to-Collision (TTC) for detected objects. For each incoming frame: Objects are detected using YOLOv5. These detections are passed to BoT-SORT to maintain consistent track identities across frames. Use Monodepth2 to estimate a dense depth map. For each tracked object, its median depth is extracted from the depth map using its bounding box. The depth history for each object is updated. TTC is calculated using the recent depth history. Results of bounding boxes, track IDs, TTC are visualized on driving video.

### B. Component Details

Object Detection

Model: YOLOv5s, the small version is chosen due to its favorable balance between detection accuracy and computational speed, making it suitable for near real-time applications.

Function: Processes each input RGB frame and outputs a list of bounding boxes [x1, y1, x2, y2], confidence scores, and class labels for detected objects. In the video, objects include vehicles, pedestrian and signals.

Multi-Object Tracking

Model: BoT-SORT

Function: Takes the raw detections from YOLOv5 as input for the current frame. It utilizes appearance features, extracted using a ReID network (osnet_x0_25_msmt17.pt), along with motion prediction (via a Kalman filter) to associate detections across frames.

Output: Provides a list of tracked objects, each with a smoothed bounding box and a unique, persistent track_id.

Monocular Depth Estimation

Model: Monodepth2 uses the pre-trained mono_1024x320 weights with a ResNet18 encoder and a corresponding DepthDecoder.

Function: Takes a single RGB input frame, resizes it to the model's expected input dimensions (feed_width, feed_height from the loaded model), and predicts a dense disparity map. This disparity map is then converted to a relative depth map (pred_depth = 1.0 / np.maximum(pred_disp, 1e-6)). Monodepth2 is typically trained using self-supervision based on view synthesis losses, requiring only monocular video sequences during training.

Time-to-Collision (TTC) Calculation

Formula: TTC = current_depth / velocity, where velocity v is estimated as the rate of change of depth.

Velocity Estimation: v = (depth_previous - depth_current) / delta_time. The code uses the last two depth measurements for a tracked object: v = (depths[-2] - depths[-1]) / (times[-1] - times[-2]). delta_time is derived from the frame index divided by frame-per-second (FPS) (frame_idx / fps).

Depth Measurement: For each tracked object, the median depth value within its bounding box is extracted from the predicted depth map (np.median(depth_map[y1:y2, x1:x2])). Median is used for robustness against outliers within the box.

History: A dictionary (object_depth_history) stores a list of (timestamp, median_depth) tuples for each track_id.

Invalid Cases: TTC is considered invalid and returns None if fewer than two depth measurements are available for an object or if the calculated velocity v is less than or equal to zero (the object is stationary or moving away).

## IV. EXPERIMENTS AND RESULTS

### A. Experimental Setup

Dataset

Evaluated monocular TTC estimation system on the KITTI depth completion dataset [10], which provides ground truth depth maps that enable quantitative evaluation of both depth estimation and subsequent TTC calculation. The validation set contains RGB images paired with corresponding sparse ground truth depth maps. This allows for direct comparison between estimated depth and ground truth, as well as derived TTC values.

Implementation Details

The system integrates three key components:

Object Detection: employed YOLOv5s (small) for efficient object detection, which achieves a balance between accuracy and inference speed.

Object Tracking: BoT-SORT with the osnet_x0_25_msmt17 ReID model was used to maintain consistent object identities across consecutive frames.

Depth Estimation: utilized Monodepth2 with the pre-trained mono_1024x320 model (training modality: mono, input size:1024x320) featuring a ResNet18 encoder and a decoder that outputs disparity at multiple scales.

All experiments were conducted on a standard workstation with PyTorch. For videos, the system processed frames at their original resolution for detection and tracking, while for depth estimation, frames were resized to 1024×320 (matching the Monodepth2 input requirements) before being upsampled back to the original resolution.

### B. Evaluation Methodology

- Depth Estimation Accuracy: compared Monodepth2's predicted depth maps against KITTI ground truth depth using standard metrics: absolute relative error (AbsRel), squared relative error (SqRel), root mean square error (RMSE) in linear and log space (RMSElog), and threshold accuracy ($\delta < 1.25$, $\delta < 1.25^2$, $\delta < 1.25^3$).
- TTC Estimation Accuracy: For each detected and tracked object, calculated TTC using both predicted depth and ground truth depth. Then computed absolute and relative errors between these values to assess TTC estimation accuracy.
- TC calculation follows the standard formula:

  TTC = current_depth / velocity

  where velocity is approximated as the rate of change in depth between consecutive frames:

velocity = (depth_previous - depth_current) / (time_current - time_previous)

### C. Results

Depth Estimation Results

Table I summarizes the performance of Monodepth2 on the KITTI depth completion validation set.

TABLE I. DEPTH ESTIMATION PERFORMANCE

| Metric | Value |
|---|---|
| AbsRel | 0.0732 |
| SqRel | 0.3586 |
| RMSE | 3.0595 |
| RMSElog | 0.1134 |
| $\delta < 1.25$ | 0.9510 |
| $\delta < 1.25^2$ | 0.9895 |
| $\delta < 1.25^3$ | 0.9962 |

The results indicate strong performance for a monocular depth estimation approach, with over 95% of pixels having an error less than 25% of the ground truth value. This provides a reliable foundation for subsequent TTC calculations.

TTC Estimation Results

Table II presents the TTC estimation errors when comparing the system's predicted TTC against ground truth TTC derived from depth measurements.

TABLE II. TTC ESTIMATION ERRORS

| Metric | Value |
|---|---|
| Total TTC Calculation | 40 |
| Mean Absolute TTC Error | 3.24s |
| Median Absolute TTC Error | 0.79s |
| Mean Relative TTC Error | 163.7% |
| Median Relative TTC Error | 61% |

The substantial difference between mean and median errors suggests the presence of outliers in TTC estimation. While the median absolute error of 0.79 seconds indicates reasonable accuracy for many cases, the high mean relative error (163.7%) highlights challenges in handling certain scenarios.

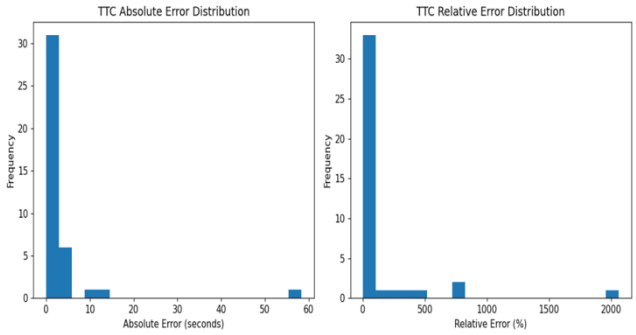


Fig. 1. The distribution of absolute and relative TTC errors.

Real-time Performance

The system achieves near real-time performance on standard hardware, with the main computational bottlenecks being the depth estimation component. Table III provides a breakdown of processing times for each component.

TABLE III. COMPONENT PROCESSING TIMES

| Component | Time(ms/frame) |
|---|---|
| Object Detection | 50.1 |
| Object Tracking | 34.6 |
| Depth Estimation | 219.5 |
| TTC Calculation | 1.3 |
| Total | 305.6 |

### D. Qualitative Analysis

When applied to driving video sequences, the system successfully detects vehicles, estimates their depth, and calculates TTC. Fig. 2 shows example frames with detected objects, their tracked IDs, and TTC estimates. The system also provides visual warnings when TTC falls below a safety threshold (2.0 seconds) for multiple consecutive frames, highlighted by changing bounding box colors from green to red.

The depth maps reveal the system's ability to accurately capture the relative depth of the scene, with closer objects appearing in warmer colors (red/yellow) and distant objects in cooler colors (blue/purple).

These qualitative results demonstrate the potential of the monocular approach for cost-effective collision warning systems.

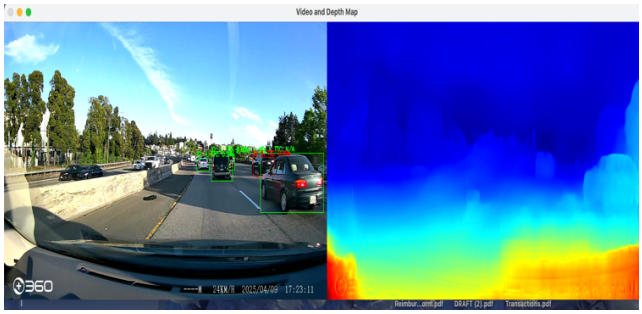


Fig. 2. Example of frames with detected objects, their tracked IDs, and TTC estimates on the left, and the depth map on the right

## V. Discussion and Summary

### A. Depth Estimation Performance

The Monodepth2 model demonstrated strong performance in estimating relative depth from monocular images, achieving an absolute relative error of 0.0732 and over 95% of pixels within 25% of ground truth depth values. This level of accuracy is remarkable given the inherent ambiguity in monocular depth estimation and compares favorably with other self-supervised approaches. However, the RMSE of 3.0595 meters indicates that while the relative structure of the scene is well-captured, absolute depth accuracy remains challenging. This is expected for a monocular system that lacks explicit depth cues such as stereo disparity or LiDAR returns.

The depth visualizations confirm that the model successfully captures the scene structure, with close objects appearing in warmer colors and distant objects in cooler colors. This relative depth ordering is useful for TTC estimation.

### B. TTC Estimation Analysis

The disparity between mean absolute TTC error (3.24 seconds) and median absolute TTC error (0.79 seconds) reveals the presence of outliers that skew the results. Similarly, the high mean relative error (163.7%) compared to the median (61.0%) also shows this pattern. These outliers are visible in Fig. 1 error distribution histograms and arise from three possible scenarios:

- Small Relative Motion: When the relative motion between the camera and object is minimal, small errors in depth estimation are amplified in the TTC calculation due to division by a very small velocity term.
- Depth Prediction Challenges: Certain surfaces, particularly reflective or textureless regions will lead to inconsistent depth predictions across frames.
- Tracking Instability: Occasional track ID switches or bounding box instability can cause sudden changes in the measured object depth, resulting in erroneous velocity calculations.

The median error of 0.79 seconds may still be too high, as TTC estimates need to be reliable for effective collision warnings. As a supplementary system, this level of performance could be acceptable.

### C. Run Time Performance

The component timing analysis shows that depth estimation is the computational bottleneck of the system, as initially hypothesized. The average processing time of approximately 97ms per frame translates to roughly 10.3 frames per second, which does not quite achieve real-time performance for autonomous driving applications that typically require 25-30 fps.

The timing breakdown shows that depth estimation consumes approximately 70% of the processing time, while object detection, tracking, and TTC calculation together account for the remaining 30%. This suggests that optimizing the depth estimation component would have the greatest improvements in overall system performance.

### D. Failure Cases and Limitations

Beyond the quantitative results, qualitative analysis shows possible scenarios where the system struggles:

- Fast motion or sudden changes in direction can cause errors in depth prediction and tracking association.
- Multiple overlapping objects sometimes lead to incorrect depth estimates due to the median depth calculation being influenced by background pixels.
- Far objects tend to have higher relative TTC errors since their depth changes are subtle and more easily obscured by prediction noise.

The system's reliance on consistent object tracking makes it vulnerable to tracking failures. When tracks are lost and later re-acquired with new IDs, the depth history is reset, requiring several frames before a new TTC can be calculated.

### E. Conclusion and Future Work

In conclusion, this work demonstrates that modern computer vision techniques can enable reasonably accurate TTC estimation from a single camera, opening possibilities for affordable driver assistance systems on a wider range of vehicles.

Future research could also explore combining the monocular approach with other sensors in a fusion framework, where the camera-based TTC estimates could complement and cross-validate radar or LiDAR measurements, potentially offering robustness beyond any single sensor modality can provide.

## References

[1] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging Into Self-Supervised Monocular Depth Estimation," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[2] M. Kopf, K. D. Schnülle, and J. Hornegger, "Time to Collision and Collision Risk Estimation from Local Scale and Motion," in Advances in Visual Computing, 2004.

[3] N. O. B. Aharon, R. Ben-Ari, I. Kaplan, and L. Havasi, "BoT-SORT: Robust Associations Multi-Pedestrian Tracking," arXiv preprint arXiv:2206.14651, 2022.

[4] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018.

[5] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[6] YOLOv5: https://github.com/ultralytics/yolov5.

[7] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv preprint arXiv:2004.10934, 2020.

[8] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime Tracking," in IEEE International Conference on Image Processing (ICIP), 2016.

[9] KITTI completion depth dataset: https://s3.eu-central-1.amazonaws.com/avg-kitti/data_depth_selection.zip

[10] BoxMot: https://github.com/mikel-brostrom/boxmot?tab=readme-ov-file

[11] Monodepth2: https://github.com/nianticlabs/monodepth2/tree/master?tab=readme-ov-file