COMP 9517 Computer Vision
T1, 2021
Project Specification

Maximum Marks achievable: 40 marks

The project is worth 40% of the total course marks. Refer to the marking criteria for detailed information about marking. Submission instructions and demo schedule will be released later.

This project consists of two components:
A. Individual Component: This component must be completed individually by each student. It is worth **15% of the total course marks**.
B. Group Component: This component must be completed by a team of up to 5 students, as confirmed by Course Admin. It is worth **25% of the total course marks**.

PLAGIARISM NOTICE

Group submissions will not be allowed for the Individual Component (A above). Your program must be entirely your own work for the Individual Component . Plagiarism detection software will be used to compare all submissions pairwise (including submissions for similar assignments in previous years, if applicable) and serious penalties will be applied, particularly in the case of repeat offences.

For both components:

- Do not copy ideas or code from others.
- If you use a publicly accessible repository (where allowed), it MUST be attributed / referenced appropriately.
- Do not allow anyone outside your group to see your code.

Please refer to the on-line resources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Academic Integrity and Plagiarism
- UNSW Plagiarism Policy Statement
- UNSW Plagiarism Management Procedure

Project Description: Over the past decade, there have been significant advances in autonomous or self-driving technology. New car models combine a variety of sensors such as radar, lidar, and GPS to better perceive their surroundings. Advanced control systems, artificial intelligence and computer vision technologies interpret sensory information to identify obstacles and enable vehicles to steer themselves on existing roads and navigate many types of environmental conditions with almost no direct human assistance. In this project, you will develop a simple visual perception module for a self-driving car.

Objectives: The project is aimed at the application of computer vision techniques learned in this course. An image-based approach, using the output of an on-board camera, will be developed to recognise vehicles, estimate distance and velocities and finally, detect driving lanes.
Learning Outcomes: After completing this project, you would have built a practical computer vision application that draws upon all the topics taught in this course. You will also learn how to work individually and with a group.

1. Dataset

For this project, the TUSimple dataset will be used. It was part of a competition released at the CVPR 2017 Workshop on Autonomous Driving Challenge. The competition had two challenges:

- Lane Detection – contains about 7,000 one-second-long video clips of 20 frames each.
- Velocity Estimation – a set of over 1,000 2-second-long video clips, with velocity and positions generated by range sensors on the last frame.

For each task below, there are instructions on which dataset to use for that task, please read that carefully.

2. Individual Component (15 marks)

For this component, you are required to detect vehicles in a given image. Implement a Python solution to detect vehicles in a single (image frame). You may implement any technique, including supervised, unsupervised, heuristic or model-based, so long as you justify your choice. If you are implementing a supervised technique, you may use all the training data provided or a portion of it, depending on the algorithm of your choice and computational resource limitations. You should display bounding boxes around the recognised vehicles.
Note: **You are required to use traditional feature extraction techniques from computer vision (hand-crafted or engineered features, and not deep learning features) to implement this task**.

INPUT DATA: For this task, you will use the Velocity Estimation Dataset and the associated human-annotated (labelled) bounding boxes. Click the links to read more and download the data. The input should be the 40th frame file path in a clip, and Figure 1 shows a sample frame in the labelled dataset.

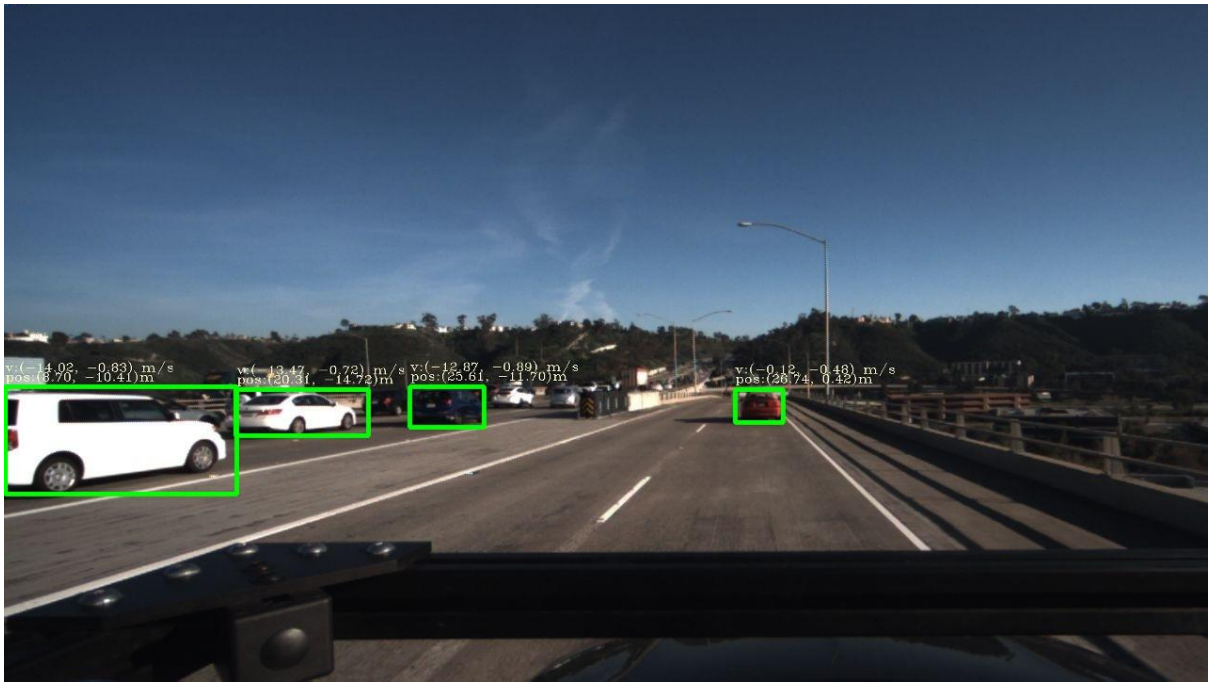

Figure 1 Vehicle Detection Ground Truth

Evaluation: You can use the demo code to format the output file and perform evaluation. Use the associated .json file to extract the ground truth bounding box labels. (**NOTE:** As the ground truth files may only have labels for those vehicles that are near the camera, you need to have a criterion on how to handle this and achieve good results.)

DELIVERABLES FOR INDIVIDUAL COMPONENT: Each student will submit an individual report of maximum 3 pages (2-column IEEE format), along with your source code(s) by Thursday week 7, April 1st, 16:59:59 AEDT. The report should include the following parts:

1. **Introduction and Background:** briefly discuss your understanding of the task specification, data and a brief literature review of relevant techniques.
2. **Method (implementation):** justify and explain the selection of the techniques you implemented, using relevant references when necessary.
3. **Experiment:** explain the experimental setup and the evaluation methods and metrics used.
4. **Results and Discussion:** provide some visual results in addition to statistical evaluation, along with a discussion on performance and outcomes.
5. **References:** list all sources including papers and code used for this task, with details of what source was used for which part of the task.

3. Group Component (25 marks)

The group component consists of THREE tasks, each of which needs to be completed as a group and will be evaluated ONCE for the whole group.

TASK 1

Visual perception plays a critical role in achieving autonomous driving. Different types of information, such as the position and motion of the agents in the environment, play an

important role in motion planning. For this task, implement a Python solution to estimate the distance of each vehicle relative to the camera in each image. You may use your implementation from the individual component to detect the vehicles prior to distance estimation. You also have the option to develop a new vehicle detection module (for example using deep learning). You should display the estimated distance above the bounding box of each detected vehicle. You may assume that vehicles are of the same width, a constant that needs to be carefully chosen. Use the following camera parameters for all tasks: fx = 414.1526px, fy = 710.3725px, cx = 713.85 px, cy = 327px. Camera height is 1.80m. (**HINT**: use triangle similarity for this task.) Refer to: http://ksimek.github.io/2013/08/13/intrinsic/ or http://emaraic.com/blog/distance-measurement for more information.

INPUT DATA: You will use the same dataset used in the individual component. The input should be the 40th frame file path in a clip.

EVALUATION: You can use the demo code to format the output file and perform evaluation. Use the associated .json file to extract the ground truth position labels. (**NOTE:** As the ground truth files may only have labels for those vehicles that are near the camera, you need to have a criterion on how to handle this and achieve good results.)

TASK 2

For this task, implement a Python solution to estimate the velocity of each recognised vehicle. You may use your implementation from the individual component to detect the vehicles prior to velocity estimation. You also have the option to develop a new vehicle detection module using any method you choose (including deep learning). You should display the estimated velocity above the bounding box of each detected vehicle.

INPUT DATA: You will use the same dataset used in the individual component. You need to estimate the velocity of each recognised vehicle in the 40th frame. You may use as many previous frames as you want for this task.

EVALUATION: You can use the demo code to format the output file and perform evaluation. Use the associated .json file to extract the ground truth velocity labels. (**NOTE:** As the ground truth files may only have labels for those vehicles that are near the camera, you need to have a criterion on how to handle this and achieve good results.)

TASK 3

In autonomous driving, lane detection is crucial as it provides localisation information to the car's system controls. In this task, implement a Python solution to detect all driving lanes in each image. Draw the detected lanes and overlay them on the input image. You can implement any method of your choice. If you are implementing a supervised technique, you may use all the training data provided or a portion of it, depending on the algorithm of your choice/computation resource limitations.

INPUT DATA: For this task, you will use the Lane Detection dataset and the associated annotated frames. Click the links to read more and download the data (lane detection). For this task, the input frame should be the 20th frame file path in a clip, and Figure 2 shows samples of the ground truth data.

Figure 2 Lane Detection Ground Truth

EVALUATION: You can use the demo code to format the output file and perform evaluation. Use the associated .json file to extract the ground truth lane labels.

DELIVERABLES FOR GROUP COMPONENT

The deliverables for the group project are 1) a group demo and 2) a group report. Both are due in Week 10. More detailed information on the two deliverables are below.

3.1 Demo

Project group demos will be scheduled in week 10. Each group will make a 12 minute online presentation to your own tutor and one assessor, and students from other groups may tune in as well. The demo should include a short slide-show presentation (5 slides maximum) explaining your methods and evaluation, followed by a demonstration of your methods, and a brief discussion of how they perform on the given data. **Recorded demos are permitted**. Afterwards, you will answer questions from the tutor/assessor/audience. All group members **must** be present for this demo. The demo roster will be released closer to the deadline.

3.2 Report

Each group will also submit a report (maximum 10 pages, 2-column IEEE format) along with the source code(s), by week 10 Friday April 23rd, 16:59:59 AEST. The report should include:

1. **Introduction:** Discuss your understanding of the task specification and data sets.
2. **Literature Review:** Review relevant techniques in literature, along with any necessary background to understand the techniques you selected.
3. **Methods:** Justify and explain the selection of the techniques you implemented, using relevant references and theories where necessary.
4. **Experimental Setup:** Explain the experimental setup and evaluation methods.

5. **Results and Discussion:** Provide statistical and visual results, along with a discussion of method performance and outcomes of the experiments.
6. **Conclusion:** Summarise what worked / did not work and recommend future work.
7. **Contribution of Group Members:** State each group member's contribution in brief. In utmost 3 lines per member, describe the component(s) that each group member contributed to.
8. **References:** List the references to papers and code used in your work, including sources used in the code with details of what is used.

3.3 Group Project Logistics

- Each member of a team generally receives the same mark for the project, however, where individual contributions to software development and report are highly unequal, this mark will be adjusted to reflect the level of contribution using peer assessments entered on the Moodle Team Evaluation tool. Peer review is mandatory, and any student who does not enter their review will get 0 for the **Contribution of Group Members** section of the report. Instructions on how to complete the peer review will be posted later on.
- It is recommended that all communications for the group project be maintained on an online system, for example the Microsoft Teams platform. Your assigned tutor will create a Team in Microsoft Teams for each project group, then invite group members to it. Your group may use this Team for communication with your tutor as well as for the consultation sessions. In addition, you may optionally maintain all the communication, code sharing and task planning within your group on Teams. Please keep the code sharing private within the group to avoid the possibility of plagiarism. If you prefer another platform for the group communication, we still recommend that you maintain it systematically. Some useful apps you can install in your Microsoft Teams include:
    - Github / Bitbucket for code sharing
    - Asana / Trello for task planning

4. References

1. Chu, H. C., & Yang, H. (2014, April). A simple image-based object velocity estimation approach. In *Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control* (pp. 102-107). IEEE.
2. Ali, A., Hassan, A., Ali, A. R., Khan, H. U., Kazmi, W., & Zaheer, A. (2020). Real-time vehicle distance estimation using single view geometry. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 1111-1120).
3. Khan, H. U., Ali, A. R., Hassan, A., Ali, A., Kazmi, W., & Zaheer, A. (2020). Lane detection using lane boundary marker network with road geometry constraints. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 1834-1843).
4. Song, Z., Lu, J., Zhang, T., & Li, H. (2020, May). End-to-end Learning for Inter-Vehicle Distance and Relative Velocity Estimation in ADAS with a Monocular Camera. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 11081-11087). IEEE.

5. Chiu, K. Y., & Lin, S. F. (2005, June). Lane detection using color-based segmentation. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* (pp. 706-711). IEEE.
6. Assidiq, A. A., Khalifa, O. O., Islam, M. R., & Khan, S. (2008, May). Real time lane detection for autonomous vehicles. In *2008 International Conference on Computer and Communication Engineering* (pp. 82-88). IEEE.
7. Low, C. Y., Zamzuri, H., & Mazlan, S. A. (2014, June). Simple robust road lane detection algorithm. In *2014 5th International Conference on Intelligent and Advanced Systems (ICIAS)* (pp. 1-4). IEEE.