University of Colorado Boulder

EXERCISE 5

BY
Amreeta Sengupta

07/26/2019
Question 2 has been performed using Jetson Nano

QUESTION 1

Stanley is an autonomous car which was designed for high-speed desert racing using technologies such as artificial intelligence, machine learning and probabilistic reasoning for the 2005 DARPA Grand Challenge. A good amount of machine vision was employed in the implementation of this system.

The system used five laser range finders with different tilt angles which would scan the terrain for short and medium range obstacle avoidance. Each laser generated a vector containing 181 measurements spaced 0.5 degrees apart which were put together in a 3D point cloud. The collected data was used to learn how the road looked like, its color and texture and this information helped in building a map of the road. This method of surveying the road with the help of lasers is called LIDAR.

The lasers were basically employed to determine which areas of the terrain were flat and classifying the different areas of the terrain as either drivable or non-drivable. It is basically implemented by dividing the space in front of the vehicle into grids. The grid points are there observed and points in the point cloud are selected which are near. The grid classification is based on the vertical difference of the nearest measurements:

$$|Z^i_k - Z^j_m| > \delta$$

- Z denoting the 3D coordinate point
- i, j are the indices for the points
- k, m are the timestamps at which measurements were done
- δ is the critical value which classifies the area as occupied or free or unknown

In order to avoid failure due to error in pose estimation, Stanley employs a probabilistic test for the detection of an obstacle which is:

$$p\,(|Z^i_k - Z^j_m| < \delta) > \alpha$$

- α is a confidence threshold parameter which can be tuned via supervised learning

The labeled training data is accumulated by driving over obstacle free terrain and marking it as obstacle-free and the left and right of the car in a distance based on the road width is labeled as obstacles.

In order to increase the labelling range, Stanley employed video camera to scan the road for obstacles located beyond the LIDAR's range. If a quadrilateral extracted from the terrain area was labelled to be obstacle free by the lasers, it was projected onto the actual camera to scout for obstacles. The projected pixels were utilized as positive labeled training data which were used to build a mixture of Gaussians for classification of drivable pixel colors. Each time a new image arrived, new Gaussian mixture components

were generated out of the projected pixels for determining whether projected pixels represented a drivable area or not. y. Pixels whose RGB value was near one or more of the learned Gaussians were classified as drivable while the rest of the pixels were flagged as non-drivable.

This helped in not only extending Stanley's vision for long range obstacles but also facilitated safe acceleration of the car. The vision computer adapted to different road scenarios making it suitable for various road conditions. It allowed Stanley to go fast if the map showed that at least 40m was drivable and it also slowed down the vehicle if an obstacle was to be encountered. Thus, the vision computer solved several problems including:

- Navigation of the car in bad road conditions (full of obstacles).
- Increasing the speed of the car by increasing the obstacle-labelling range of the car.
- Facilitated safe acceleration of the car.
- Acting as an early warning system by detecting obstacles beyond the detection range of the lasers.

Even though Stanley was a huge success in the domain of self-driving cars, some limitations to its system are:

- The race environment was static, and the system was not built for navigation through traffic.
- The software was not able to handle all types of obstacles. For example, it was unable to differentiate between tall grass and rocks.

Sophisticated machine vision system with multiple cameras can also be used for the same. Active 3D capturing systems can be used to meaningfully represent the scene in the form of grids and determine the depth of the obstacles which would make the car suitable for maneuvering in an urban environment which is not static. Multiple cameras can be employed for a 360-degree view around the car to avoid blind spots. Machine vision can also be used for pedestrian detection, road sign detection, lane detection etc. Thus, the car will be able to navigate through traffic. Hence it can help in visual navigation as well as obstacle detection.

QUESTION 2

Design for Pedestrian Detection

I have used Histogram of gradients (HOG descriptor) for pedestrian detection. HOG can be used to describe the object of interest in an image using the distribution of intensity gradients. Gradients are employed as the magnitude of gradients have a larger value around the edges or regions with change in intensity which can be used to extract information about the object of interest. Basically, the pixels of the image are grouped into cells and each of these cells are utilized to extract the histogram feature based on the gradient orientation. A bin selected based on the gradient direction and the gradient magnitude is utilized for selection of the weight used for voting into the histogram. Using this, the HOG feature vector is calculated which is basically the concatenation of the histograms, and these features help in extraction of the representations of the objects in the image. Support Vector Machine (SVM) is a machine learning technique which is used to classify the detected features. It involves use of positive and negative examples for training and then makes a binary decision which is to classify the objects as human or non-human.

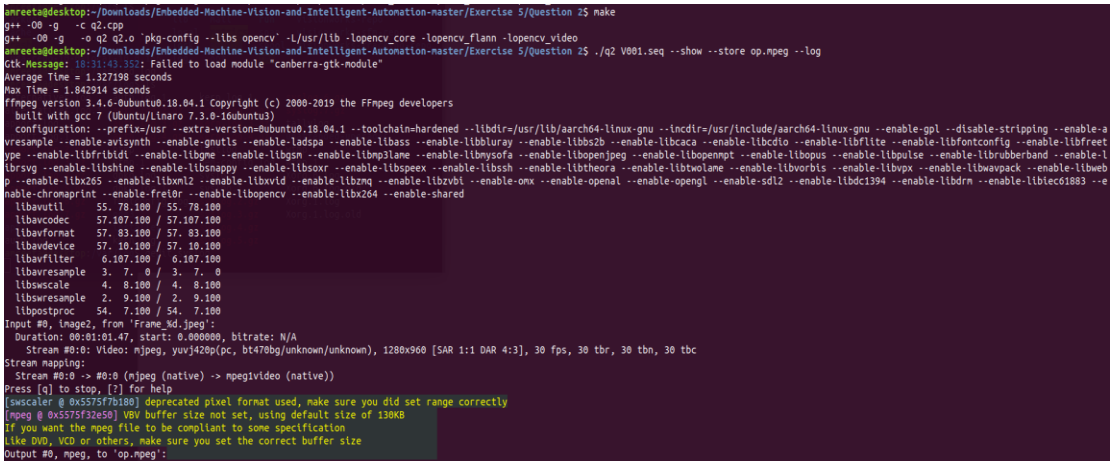
Code

- This code is used to detect Pedestrians using HOG and the detection is done using SVM.
- The code gives the user the option of 4 runtime arguments:
    1) The first one being the video name on which the pedestrian detection algorithm is applied.
    2) The next one can be "--show" option which is used to display the result on each frame.
    3) Another one is the "--store" option followed by the filename which is used to store each transformed frame and re-encode it back to video with the filename as chosen by the user.
    4) The last one being the "--log" option which is used to log the number of pedestrians detected and the coordinates of the detected pedestrians in syslog.

```
Jul 26 19:44:48 desktop LOG_RESULT: Number of Pedestrians detected in Frame 0 = 2
Jul 26 19:44:48 desktop LOG_RESULT: Coordinates of Pedastrian 0: x=1186 y=540
Jul 26 19:44:48 desktop LOG_RESULT: Coordinates of Pedastrian 1: x=1177 y=571
Jul 26 19:44:49 desktop LOG_RESULT: Number of Pedestrians detected in Frame 1 = 2
Jul 26 19:44:49 desktop LOG_RESULT: Coordinates of Pedastrian 0: x=1186 y=540
Jul 26 19:44:49 desktop LOG_RESULT: Coordinates of Pedastrian 1: x=1177 y=571
Jul 26 19:44:51 desktop LOG_RESULT: Number of Pedestrians detected in Frame 2 = 2
Jul 26 19:44:51 desktop LOG_RESULT: Coordinates of Pedastrian 0: x=1186 y=540
Jul 26 19:44:51 desktop LOG_RESULT: Coordinates of Pedastrian 1: x=1177 y=571
Jul 26 19:44:52 desktop LOG_RESULT: Number of Pedestrians detected in Frame 3 = 4
Jul 26 19:44:52 desktop LOG_RESULT: Coordinates of Pedastrian 0: x=725 y=388
Jul 26 19:44:52 desktop LOG_RESULT: Coordinates of Pedastrian 1: x=1192 y=576
Jul 26 19:44:52 desktop LOG_RESULT: Coordinates of Pedastrian 2: x=520 y=471
Jul 26 19:44:52 desktop LOG_RESULT: Coordinates of Pedastrian 3: x=1213 y=312
```

- After each image frame is read, they are resized eliminating a certain number of rows and columns. This is done in order to speed-up the detection process making it less time consuming.
- Next, the HOG descriptor along with SVM detector is applied on each image frame to detect the pedestrians.
- A bounding rectangle is drawn around each detected pedestrian.
- Timestamp calculation has also been performed for analysis of the algorithm.
- ESC can be pressed by the user in order to exit.

Build and test



Analysis of this algorithm using the Caltech Dataset

This code is used to detect Pedestrians using HOG and detection is done using SVM. It has been tested using the Caltech Dataset. The Caltech Dataset is quite a large dataset with several videos which containing approximately 10 hours of 30Hz video taken from a vehicle driving through traffic in an urban environment. After running the code for multiple videos in the dataset, following observations were made:

1. Accuracy of this algorithm does not necessarily match the requirements. In fact, it detects several false positives i.e. non-pedestrians in the video. In some cases, it detects nothing even when pedestrians are present. This failure can occur due to several reasons, some of which include:
    - Failure of detection can sometimes be due to different environmental conditions.
    - Accuracy can sometimes deteriorate due to poor image quality owing to changes in lighting.
    - Since the image is resized to optimize the algorithm, there is a possibility that a pedestrian at the border of the image may not be detected.
    - Different appearances, shapes, sizes and postures of pedestrians can sometimes lead to non-detection.

    The accuracy of the detection can be improved by increasing the scaling of the image. However, this also leads to increase in the processing time which will make it unsuitable for real-time applications.

2. HOG used along with SVM for pedestrian detection tested using the Caltech Dataset, requires quite a bit of processing time. The processing time is reduced to a certain extent by resizing the image and elimination certain number of rows and columns for speeding up while sacrificing the detection accuracy. However, it is not very suitable for real-time applications. The table below shows the execution time for few videos from the Caltech Dataset:

Video Number	Number of Frames in the Video	Average Execution Time	Maximum Execution Time
V001.seq in Set 00	1844	1.327198 seconds	1.842914 seconds
V005.seq in Set 02	1841	1.238746 seconds	1.641090 seconds
V007.seq in Set 05	1842	1.238982 seconds	1.954851 seconds
V002.seq in Set 07	1841	1.233921 seconds	1.700820 seconds
V000.seq in Set 10	1842	1.224591 seconds	1.617663 seconds

The Caltech Dataset has grouped the pedestrians into three categories based on pixel height which are:

- Near scale: 80 or more pixels
- Medium scale: 30-80 pixels
- Far scale: 30 pixels or less

According to the Caltech dataset, for the unoccluded near pedestrians, this algorithm has a miss rate of under 40% at 1 FPPI (False Positives per Image). At medium scale, performance deteriorates with around 72% miss rate at 1 FPPI. At far scale, it is unable to achieve more than 8% recall at 1 FPPI. The reasons

contributing to failure are occlusion and low resolutions. Thus, the algorithm fails in detection at smaller scales and partially occluded pedestrians making it unsuitable for real-time applications.

QUESTION 3

The project will be implemented along with my project partner Vatsal Sheth.

Relevance of selection

We aim to utilize machine vision algorithms and techniques to implement an autonomous vehicle with real-time performance. The reason for the selection of this project is as follows:

- Millions of people throughout the world die due to motor vehicle accidents. A self-driving vehicle could be a step towards safer transportation options and could be instrumental in reducing the deaths caused by inattentive or distracted drivers.
- A self-driving vehicle could be a very good option for people who are unable to drive or people with disabilities or elderly people, making their commute more comfortable.
- It could also help in traffic management due to automatic road sign detection and regulation following vehicles.

Challenges

- Accurate feature detection and elimination of the false positives. Failure of detection can sometimes be due to different environmental conditions or owing to changes in lighting. Different appearances, shapes, sizes and postures of objects can also sometimes lead to non-detection.
- Implementation of multi-threading approach and accurate scheduling of the tasks so that the system can be used in real-time while giving a good performance.
- Integration of the various modules while making sure that the code is optimized can be challenging.

Rationale for selection or proposal in terms of application and learning

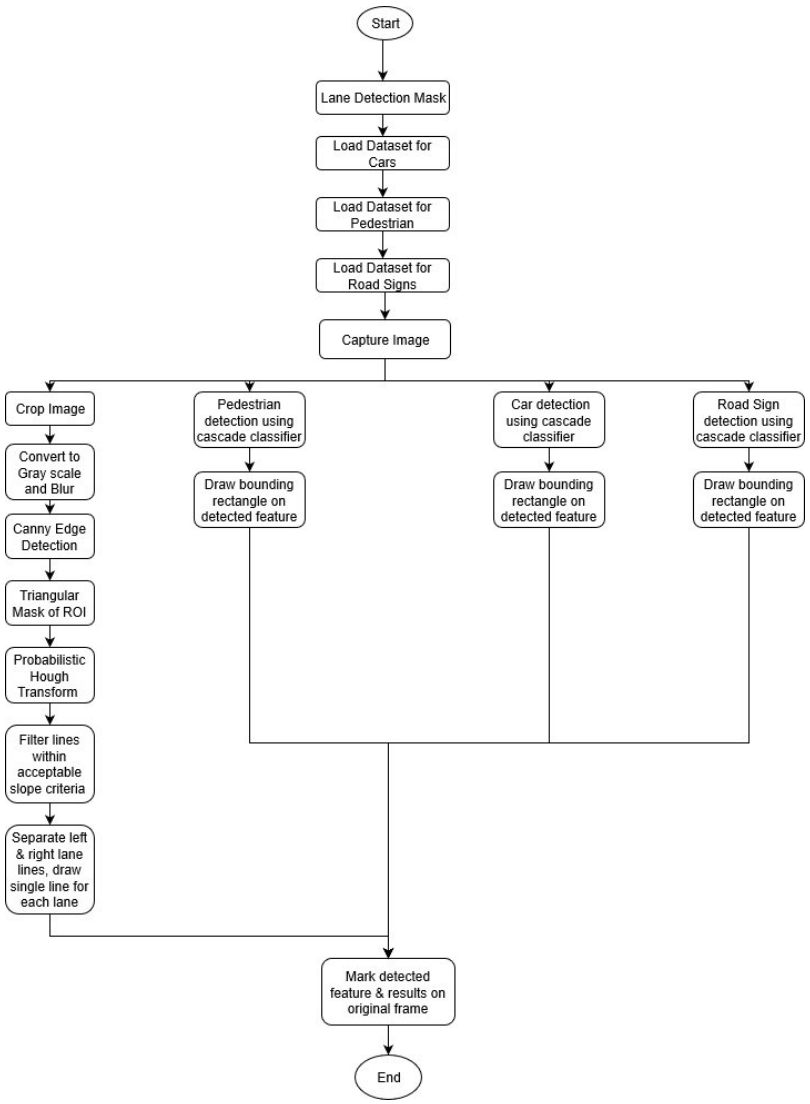
- This project promotes elimination of human intervention by facilitating self-driving cars. In terms of real-time, implementation of this on a large scale is quite challenging. However, we can use the machine vision concepts that we have learned to design few features of this system for a safer and reliable future. We can improve the system performance while also making sure it works in real-time for the modules that we have selected for implementation.

- Implementation of this project will cover quite a few machine vision techniques and algorithms including:
  1) Image capturing and encoding.
  2) Post capture image transformation (computing greyscale of an image, resizing an image, blurring an image).
  3) Image parsing (edge and feature detection)
  4) Image understanding (object detection and tracking).

- From Exercise 5, I have learnt how to perform pedestrian detection. However, for use in real-time applications, the algorithm needs to be optimized to give the best result. This project will give me an opportunity to take a step towards implementing this module of a self-driving car which can be used in real-time with a dependable performance.

QUESTION 4

The project will be implemented along with my project partner Vatsal Sheth.



Minimum

- Lane detection: This includes creation of the lane detection mask first on a single frame and after continuously capturing the image some pre-processing steps are involved. These include cropping the image frame, converting it to grayscale, blurring the image, extracting a triangular mask of region of interest. Post this, Probabilistic Hough Transform will be applied on the image and lines with acceptable slope criteria will be filtered. Next, the left and right lanes will be separated, and lines will be drawn on each lane.

- Pedestrian Detection: This includes capturing each image frame and re-sizing them so that a certain number of rows and columns are eliminated. This is done in order to speed-up the detection process. Post this, HOG descriptor along with SVM detector will be applied in order to detect pedestrians on each image frame. Lastly, a bounding rectangle will be drawn around each detected pedestrian.

- Car Detection: This would include loading the dataset for cars. After capturing each image frame, some pre-processing operations will be performed. Next, the car will be detected using the cascade classifier. Lastly, a bounding rectangle will be drawn around each detected car.

- Road Sign Detection: This would include detection of a particular road sign like 'STOP' sign. It would involve loading the dataset for road signs. After capturing each image frame, some pre-processing operations will be performed. Next, the road sign will be detected using the cascade classifier. Lastly, a bounding rectangle will be drawn around each detected sign.

Target

The target of the project is to utilize machine vision algorithms and techniques to implement an autonomous vehicle with the following features:

- The lane detection algorithm will be used to implement a lane departure warning system which would display a warning message on crossing a certain lane.
- The pedestrian detection algorithm will be implemented to detect the pedestrians within a certain range and display a warning message for the same to avoid accidents.
- The car detection algorithm will be implemented in order to detect the cars within a certain range and display a warning message for the same to navigate safely and avoid accidents.
- The road-sign detection algorithm will be implemented to detect the approaching road sign and display a warning message in order to stop the vehicle.
- Time measurement and frame rate measurement for real-time performance analysis.

Optimal

- Implement a real-time image processing system by using a multi-threaded approach which would execute the image-processing tasks parallelly and will help in improving the system performance.
- Try to use CUDA which will help in providing a better performance.
- To improve the accuracy of feature detection, we will try to train the cascade descriptor using positive and negative samples and simple machine learning algorithm available in OpenCV and this trained descriptor will be used along with the cascade classifier to extract features.
- Feature descriptor extraction and detection is a time-consuming task and if there is no significant change in successive frames, after analysis, some frames can be eliminated to save time in order to meet the real-time requirements. For example, car detection can be performed for every three frames which will speed up the detection process.
- Synchronization between the tasks and application of suitable scheduling policies for optimized and accurate results.

REFERENCES

- http://robots.stanford.edu/papers/thrun.stanley05.pdf
- https://www.ias.informatik.tu-darmstadt.de/uploads/Teaching/RobotLearningSeminar/Glaser_RLS_2012.pdf
- https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf
- https://pdfs.semanticscholar.org/28e1/fa9bc622f035a59bf309837d75c08f630a57.pdf
- https://www.learnopencv.com/histogram-of-oriented-gradients/
- http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/files/CVPR09pedestrians.pdf
- https://docs.opencv.org