Simulation of Path Planning Algorithms Using Commercially Available Road Datasets with Multi-modal Sensory Data

R. Senthilnathan, Arjun Venugopal, K. S. Vishnu

Simulation of Path Planning Algorithms using Commercially Available Road Datasets with Multi-modal Sensory Data

R. Senthilnathan[1], Arjun Venugopal[1], K. S.Vishnu[1]

[1]Department of Mechatronics Engineering, SRM Institute of Science and Technology, Kattankulathur, India

Abstract. Road datasets for computer vision tasks involved in advanced driver assist systems and autonomous driving are publicly available for the technical community for the development of machine learning aided scene understanding using computer vision systems. All the perceived data from multiple sensors mounted on the vehicle must be fused to generate an accurate state of the vehicle and its surroundings. The paper presents details of the simulation implementation of local path planning for an autonomous vehicle based on multi-sensory information. The simulation is carried out with sensory inputs from RGB camera, LIDAR and GPS. The data is obtained from the KITTI dataset. A variant of the D-star algorithm is utilized to demonstrate global and local path-planning capabilities in the simulation environment.

Keywords: Multi-sensory data fusion,Object Detection, Range Detection, RGB-D Data, Path Planning.

1 Introduction

Advanced Driver Assist Systems (ADAS) and autonomous driving technology have greatly contributed to the explosive growth of the field of deep learning which was fueled by the massive collection and availability of road datasets for public usage. Such tasks use multiple sensors such as cameras, inertial measurement units (IMU), global positioning systems (GPS), range finders such as LIDAR sensor and RADAR sensor. An autonomous driving vehicle utilizes all these sensors for tasks such as localization of the vehicle in a metric map, mapping of surroundings with reference to the vehicle, etc. The sensors fundamentally aid in the perception stage of the autonomous vehicle's control loop. The role of perception is to aid the autonomous vehicle to plan trajectories and execute graceful navigation to the destination. Hence it may be appreciated that all the perceived information must be ending up utilized in path planning stages especially for the purpose of local path planning which often is an obstacle avoidance problem. The paper presents the details of the simulation of local path planning algorithms based on the sensory dataset of road scenes available as part of the KITTI benchmarking suite [1, 2]. The task of path planning generally manifests in at least two ways namely global and local. Global path planning is car-

ried out from source to destination of an autonomous driving mission based on GPS data with reference to a map of the environment. Local path planning requires an accurate estimate of the objects in the road such as vehicles, pedestrians, civil structures, etc. In autonomous vehicles, utilizing multiple sensors is generally the norm. This is required to address the inherent sensor noise and aliasing present in the data obtained from a completely unstructured environment. The current work utilizes the sensory data of the sample driving scenario from the KITTI dataset for simulation studies. Simulation studies refer to the offline processing of real-world data acquired from sensors such as RGB-camera, LIDAR, GPS, etc. This multi-modal sensory information is acquired from the real world and can be further used to implement intelligent behavior in many systems. The intelligence of a system depends on the interpretation of sensory data for the perception of the environment. Also, the accuracy of a system corresponds to the amount of data the system is able to collect or process within the given span of time. Simulation facilitates analysis and visualization of the performance of the system in a virtual environment in which the errors can be corrected then and there. When the simulation results are up to the mark, the model can be used for deployment in a real-time environment. Many simulation-based studies related to ADAS and autonomous driving had been proposed.

1.1 Elements of Simulation Studies

The simulation studies are based on images, range data (LIDAR), and GPS information. The various tasks using such information are addressed below.

Object Classification. It is used to classify the set of similar objects in a scenario as a group. Finally, the scene will consist of several such groups.

Object Detection. To detect objects, if any, which are of interest present in a scenario at any point in time and to generate a bounding box for such objects. The object is segmented by a bounding box generated around the detected objects.

Range Detection. The distance between the ego vehicle with respect to the objects is found out from the given scenario and to generate a bounding box for such objects.

Registration of Range Data on RGB Image. The object is segmented by a boundary which is with respect to the distance from the ego vehicle.

Collision Avoidance. Similar to range detection, but the focus is not just to detect the object, rather use such information to avoid the collision of the object with the ego vehicle.

Path Planning. To generate a path, the ego vehicle must take for the given start and goal point which ensures collision-free traverse.

1.2 Limitations of Simulation Studies

The simulation studies are performed in an environment that assumes the system or its parameters be static in nature which is not the case, and any changes in the system affect the overall performance. Some constraints of such systems are presented as follows:

Environmental constraints. Any change in the behavior of the environment with respect to the environment it was trained result in an unpredictable solution or failure.

Real-time constraints. The simulation studies do not work as efficiently as it does in a simulated environment when compared to the real-world environment.

Data availability constraints. In a simulated environment, the data is already available for the computational purpose, which is not the case in a real-time environment, since the collection of data from a real-world environment in a real-time manner is a crucial task. The real-world data acquisition may not exactly meet the fidelity.

2 Details of Dataset

To find the suitable dataset for simulation studies, the datasets available for road data were referred, and among them, the three most suited datasets which contain RGB, range and GPS information are filtered for final selection. Some of the datasets considered for the purpose include the Ford Campus Vision and LIDAR dataset [3], The KITTI Vision dataset, and Oxford RobotCar Dataset [4]. Among these, the KITTI dataset was chosen based on the merit of the abundance of supporting materials for development and popularity in benchmarking for various computer vision tasks. The other advantage of the KITTI dataset is the content-based grouping. The various groupings include stereo, flow, depth, odometry, tracking and road semantics. For all the work reported in this paper, the KITTI tracking dataset acquired during the year 2012 is utilized [2]. The tracking section consists of different categories of files which include left and right color images, LIDAR point clouds, GPS/IMU data, camera calibration matrices [5], training labels, L-SVM reference detections, Regionlet reference detections, tracking development kit many more. Among these, the simulation specific files are utilized from the tracking dataset which is explained in the following sections.

Since in the current work the RGB images are used only for semantic understanding of the scene the left camera images are utilized for the same. This file consists of RGB information taken from the left camera mounted on top of the ego vehicle. The

RGB information is packed as Portable Network Graphics (PNG) file format and has a resolution of 1242 × 375 pixels.

The second sensory data used in the work is the LIDAR point cloud data. This file consists of laser information taken from the Velodyne laser scanner mounted on top of the ego vehicle and the specifications are listed in Table 1.

Table 1. Specifications of LIDAR Point Cloud Data.

Parameter	Specification
File format	Text(TXT)
Point information utilized in the current work	'*x*' – coordinate in m '*y*' – coordinate in m '*z*' – coordinate in m

The third sensory data utilized is the GPS/IMU combo. This file consists of GPS/IMU information taken at each instance the frame from the camera was captured, the GPS/IMU sensor is also mounted on top of the ego vehicle. The various information contained in the data is listed in Table 2.

Table 2. Specifications of GPS/IMU Data.

Parameter	Specification
File format	Text (TXT)
GPS information utilized in the current work	latitude in deg longitude in deg

One of the important tasks involved in utilizing multiple sensors is the requirement for registration of correspondence of information recorded by the sensors with respect to the one frame of reference. KITTI dataset provides accurate measurements of the physical locations of the sensors with reference to each other in terms of positions and orientations. This information is generally part of system calibration which enables tasks such as sensor fusion. In KITTI dataset a set of calibration matrices that defines the transformation between various sensors located on the vehicle, which are listed in Table 3.

Table 3.KITTI Calibration Matrices.

Parameter	Specification
Matrix information utilized in the current work	projection matrix of left color image rotation matrix of the camera transformation matrix from LIDAR to camera

3 Deep Learning Inferences on RGB Images

When multiple objects are present in the image, the task is to localize the objects in the image, also known as object detection. From the term object detection, it is evident that it predicts an output or detects objects which are of interest based on application. When an input is fed to the system (i.e., images), the computer runs a suitable algorithm and bounding boxes are generated for the objects present in the input image. Since images are large data and consist of several features, it is impossible for shallow networks to predict the output with high accuracy. Therefore, a deep learning network which consists of a greater number of layers is chosen. For the purpose of object detection, a pre-trained model of Faster RCNN with Resnet 101 [6] is utilized. The training specification of the pre-trained model is listed in Table 4.

Table 4. Training Specifications of Faster R-CNN Resnet 101.

First Stage	Second Stage
Anchor generator:	Bounding box predictor:
Height stride: 16	Regularizer:
Width stride: 16	Regularizer: 12
Scales: 0.25, 0.5, 1.0 and 2.0	Weight: 0.0
Aspect ratios: 0.5, 1.0 and 2.0	Initializer:
Bounding box predictor:	Variant scaling
Regularizer:	Factor: 1
Regularizer: 12	Uniform: True
Weight: 0.0	Dropout:
Initializer:	Usage: False
Truncated normal	Keep probability: 1.0
Stddev: 0.00999999977648	Post processing
Non max suppression:	Batch non max suppression:
score threshold: 0.0	score threshold: 0.300000011921
iou threshold: 0.699999988079	iou threshold: 0.600000023842
max proposals: 100	max detections per class: 100
Localization loss weight: 2.0	max total detections: 100
Objectness loss weight: 1.0	Localization loss weight: 2.0
Initial crop size: 14	Objectness loss weight: 1.0
Maxpool kernel size: 2	Score converter: SOFTMAX
Maxpool stride: 2	
Training Configuration	**Evaluation Configuration**
Batch size: 1	Number of examples: 8000
Learning rate	Maximum evaluations: 10
Step: 0	Moving average: False
Learning rate: 0.000300000014249	Shuffle: False
Step: 900000	Number of readers: 1
Learning rate: 2.99999992421e-05	
Step: 120000	

Learning rate: 3.00000010611e-06
Momentum: 0.899999976158
Moving average: False
Gradient clipping: True

The model uses Faster R-CNN algorithm for object detection when an input image is fed to the model and classifier used is a type of Residual network with 101 layers called ResNet-101, The dataset was trained on MS Common Objects in Context (COCO) [7] which is a dataset with 330K images and 80 object categories of which more than 200K of them are labeled. An illustration of the Faster R-CNN network architecture is presented in Fig. 1.

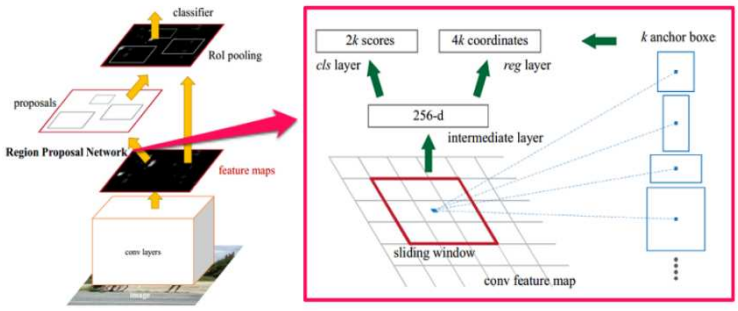

Fig. 1. Faster RCNN Architecture[6].

For the pre-trained network implementation, TensorFlow API is used. The files of a pre-trained model include those listed in Table 5.

Table 5. Files in a pre-trained model [8].

File	Description
checkpoint:	This file contains all the checkpoint name and its path.
model.ckpt.data-00000-of-00001	This file contains the values of variables i.e., weights, biases, placeholders, gradients, hyper-parameters, etc.
model.ckpt.index	This file contains a table of each tensor and its values
model.ckpt.meta	This file contains the complete graph, separately from the variables
frozen_inference_graph.pb	This file contains the model architecture and weights in a single file
pipeline.config	This file contains the training specification of the network, training configuration, evaluation configuration, etc.

Inferencing on a pre-trained model is advantageous since it does not require to train the model which is again a time-consuming process and moreover for a simulation task based on popular datasets pre-trained models are more than adequate. Fig. 2

shows a sample result of network inference. The inferencing was done for 154 frames which are continuously captured image sequence and in all cases the model was able to detect the objects which are of interest. The pre-trained model was set to detect 4 classes of labels namely *"person"*, *"two-wheeler"*, *"light vehicle"*, and *"heavy vehicle"*, but the actual model was trained to detect 90 classes of labels. Therefor the classes which are not required for the scope of this task are omitted and are represented as *"N/A"*. All the detectable classes may be noted in the figure.

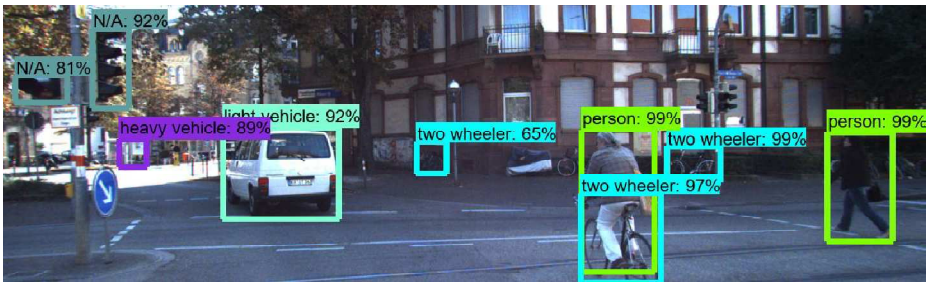

Fig. 2. Object Detection Results

4 Range Data Processing

The LIDAR point clouds are also referred to as range data. This data consists of the distance of the surrounding objects scanned with reference to a home position which in this case corresponds to ego vehicle. This data gives a description of the position of each object as a collection of points. The detailed description of the data structure and its plot visualization are explained in the following sub-sections.

4.1 Description of Data Container

The primary data structure used for data storage is arrays. Arrays give the unique feature of storing the same type of data under the same name and the data can be an n-dimensional vector. The programming language used is *Python 3* and since python does not contain an inbuilt array data structure, the *NumPy* library is used for support. The LIDAR point cloud is stored as a binary file and contains a minimum of 1 lakh points per file where each point has its respective *x*, *y*, *z* and *reflectance*in meters. This binary file is unpacked using the *fromfile* functionality in *NumPy*.

4.2 LIDAR Data Visualization

Visualization is the only way to observe, the correctness of obtained data or the data on which different processing techniques are used, with reference to the desired behavior. Visualization involves plotting of points in 2-D and 3-D space to understand the output image or any other data. Here, *Matplotlib* is used to handle all kinds of visualization tasks. The LIDAR data is a collection of points in 3-D space, in which each

point is represented by the three-coordinate system. This information is unpacked and stored as an array of ($n \times m$) where, n corresponds to rows and m corresponds to columns, in this case, m is taken as 3. Fig. 3 shows the plot visualization of LIDAR data.

4.3 Correspondence Between RGB and Range Data

In an RGB image, the three layers are stacked above one another to visually produce real-world information based on its actual color. The RGB image is the scaled two-dimensional version of the real-world information which does not provide any information with regard to the actual distance of the object, actual height of the object, or about the actual width of the object. Whereas, the data obtained from LIDAR gives the distance of the object with respect to the LIDAR and this information is on the metric scale. To establish the correspondence between camera-RGB data and range data, both the data has to be in pixels. This is done by performing matrix multiplication with four different matrices in which the LIDAR coordinated from metric scale are converted to pixel information, similar to how the camera captures the image. The first two matrices are the function of internal and external parameters of the camera which obtained from camera modeling, the third matrix is used to transform the LIDAR frame with respect to the camera frame and the final matrix corresponds to the LIDAR points.

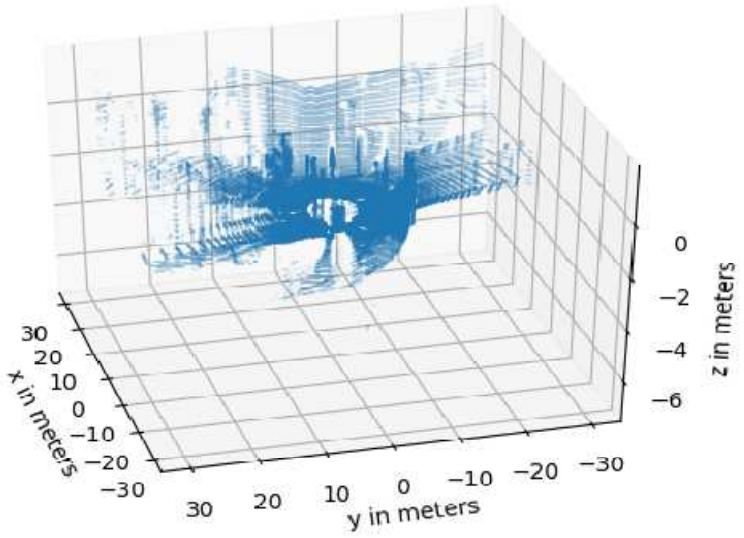

Fig. 3.3-D Plot of LIDAR Data

In order to establish a correspondence between LIDAR data and RGB image, the necessary transformation equation [9] must be formulated which are provided as part of the dataset. The image points i.e., x and y in pixels are obtained by performing the

matrix multiplication in the order presented in equation 1. The projection matrix is also called as the intrinsic parameters of the camera and the rotation matrix is also called as the extrinsic parameters of the camera. The transformation matrix is to relate the LIDAR frame with respect to the camera frame.

$$s\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} fx & 0 & cx & dx \\ 0 & -fy & cy & dy \\ 0 & 0 & 1 & dz \end{bmatrix} \times \begin{bmatrix} r11 & r12 & r13 & 0 \\ r21 & r22 & r23 & 0 \\ r31 & r32 & r33 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} t11 & t12 & t13 & tx \\ t21 & t22 & t23 & ty \\ t31 & t32 & t33 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} lx \\ ly \\ lz \\ 1 \end{bmatrix} \quad (1)$$

Where,

- '*x*' and '*y*', the '*x*' and '*y*' of the LIDAR point in pixels respectively.
- '*fx*' and '*fy*', the focal length in '*x*' and '*y*' respectively.
- '*cx*' and '*cy*', the principal point offset in '*x*' and '*y*' respectively.
- '*dx*', '*dy*', and '*dz*', the distortion in '*x*', '*y*', and '*z*' respectively.
- '*r11*'- '*r33*', the rotation with respect to the camera frame.
- '*tr11*'- '*tr33*', the rotation of LIDAR frame with respect to camera frame.
- '*tx*', '*ty*', and '*tz*', the translation of LIDAR frame with respect to camera frame.
- '*lx*', '*ly*', and '*lz*', the '*x*', '*y*', and '*z*' of the LIDAR points in metric scale respectively.

By performing the matrix multiplication, the correspondence between RGB and range can be obtained. Hence, by plotting the points of LIDAR which lies in the image resolution, the RGB-D data is obtained. Fig. 4 shows an RGB-D image in which the LIDAR range information are indicated by green, blue, red and yellow colored points in the increasing order of range.

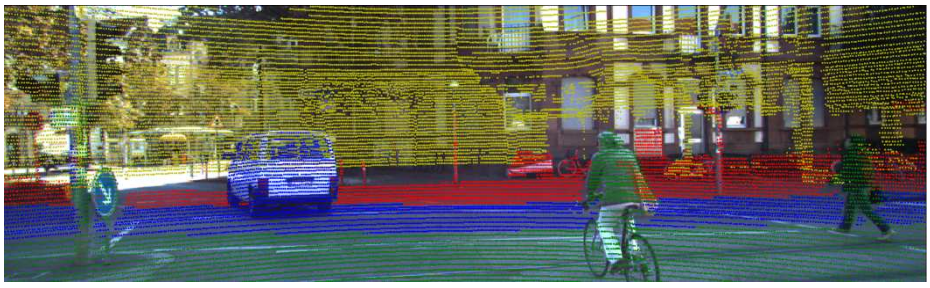

Fig. 4.RGB-D Data

The RGB-D data obtained is segmented based on the requirement i.e., the objects of interest. Here, the segmentation is based on the number of range points received back from the LIDAR sensor for all objects of interest. Fig. 5 shows the range seg-

mented RGB-D image in which the LIDAR range information are indicated by green, blue, and red colored points in the increasing order of range.

Fig. 5. Segmented RGB-D Data

5 Path Planning

The work reported in this paper demonstrates the simulation of path planning based on multi-sensory data acquired from the real-world. In order to demonstrate the local path planning capabilities based on the proposed method, the vehicle must first have an intended path to connect any given source point to the destination. This task in mobile robotics literature is referred to as global path planning. Since it is a simulation of the path planning algorithm that is intended, the actual path traveled by the vehicle used for acquiring the dataset is considered as the intended path to travel in the simulation environment. It must be noted that the path is the actual path traveled by the vehicle. This information may be obtained from the GPS data in the dataset. The GPS coordinates from the KITTI dataset are in the form of latitude and longitude measured degrees. In order to have a path planning performed on a map, it is convenient to have the map coordinates recorded in linear dimensions in meters. The python library '*utm*' is used to convert the coordinates to meters and the plot obtained is in the easting-northing convention. To convert this convention into the *x-y* right-handed orthogonal convention, the subtraction of every point with the first point is carried out. Hence, the first point will be (0,0) and the rest of the points take this as reference. Table 6. shows thetest case results obtained on different sample video scenesfrom the same dataset. The figures illustrated in this paper are results obtained from Test Case – 1.

Table 6. Specification of Test Cases

Test Case - 1	Specification
Number of frames from video scene	154
Number of GPS points from scene	154
Objects detected using pre-trained network	9
Objects under 20m LIDAR range	7
Detected classes of labels:	

Person	2
Two-wheeler	4
Light vehicle	1
Heavy vehicle	0
Test Case - 2	Specification
Number of frames from video scene	374
Number of GPS points from scene	374
Objects detected using pre-trained network	12
Objects under 20m LIDAR range	8
Detected classes of labels:	
Person	0
Two-wheeler	0
Light vehicle	8
Heavy vehicle	0

5.1 GPS-based Global Map

The GPS information of the original acquired dataset contained 15 parameters and out of which the 2 parameters utilized were latitude and longitude. By using these parameters and performing appropriate mathematical conversions, the path travelled by the ego vehicle was obtained which in this case is considered as the global map. The objects detected from the deep learning network's inference on the RGB image and the corresponding point cloud data were also plotted in this graph as shown in Fig. 6. Objects are represented as blue colored blobs and red-colored points represent the GPS path taken by the ego vehicle.

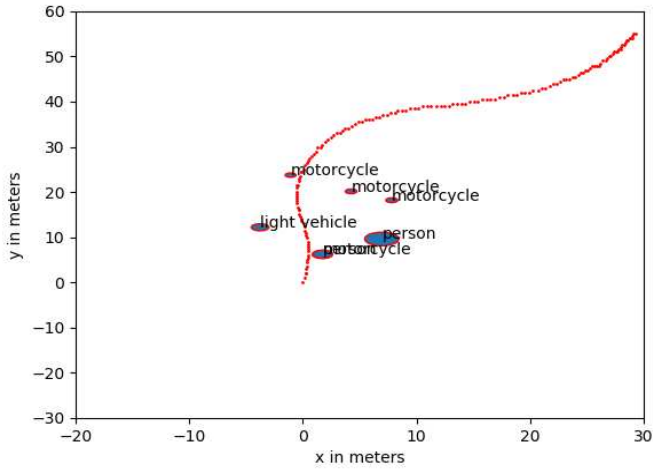

Fig. 6.GPS Data-based Global Intended Path

5.2 Global Map-based Local Path Planning

Path planning algorithms are basically graph traversal algorithms that estimates the best and least possible distance from a start position to a goal position using numerous theorems. The nature of the required task involved direct traversal from start to goal. The possible path planning algorithms referred were algorithms which had the capability to change the cost functionand re-plan frequently. The selection list came down to Lifelong Planning A Star (LPA*) path planning algorithm [10] and D Star (D*) Lite path planning algorithm [11]. LPA* is the modified form of A Star (A*) path planning algorithm [12] and is used to find the shortest path under changing costs. The start position and goal position are static and the direction of search is from start to goal. D* Lite implements the same behavior as that of D Star (D*) path planning algorithm [13] and is based on LPA*. Similar to LPA*, the D* Lite algorithm is used to find the shortest path under changing costs except that when there is a change in cost, the algorithm re-plans the costs from the goal position to the current position. The direction of search is from goal to start and the start position is not static unlike LPA*. D* Lite was preferred over LPA* because of its ability to take decisions under changing environmental conditions with respect to the current position.

The local path planning is done with respect to this GPS data-based global intended path obtained using the number of framesfrom the video scene and by performing D* Lite path planning algorithm as shown in Fig. 7. The position of the ego vehicle in the first frame is taken an origin, which is also taken initially as the start position in the path planning algorithm. The goal position is taken to be 20 m away from the start position along the global intended path. This is just to ensure that any local planning should not be carried out without considering the global outcomes. A slight variation is applied to the way in which the D* Lite algorithm is applied. In original D* Lite algorithm, the goal position is considered to be static while the start position may vary. In this paper, the D* Lite algorithm is performed from frame to frame.Therefore, both the start position and goal positions vary in every frame.After a path is generated in a frame, the goal position becomes the start position and a new goal position is set using the same criteria for rest of the frames. This criterion is applied to ensure the dynamic nature of the environment is considered while performing the path planning. In Fig. 7. the green point represents the start and red represents a goal, the black blobs are objects present in the current frame as detected by the deep learning network and registered against the LIDAR data for range from, the ego-vehicle. The black line represents the path generated from start to goal on the particular frame.

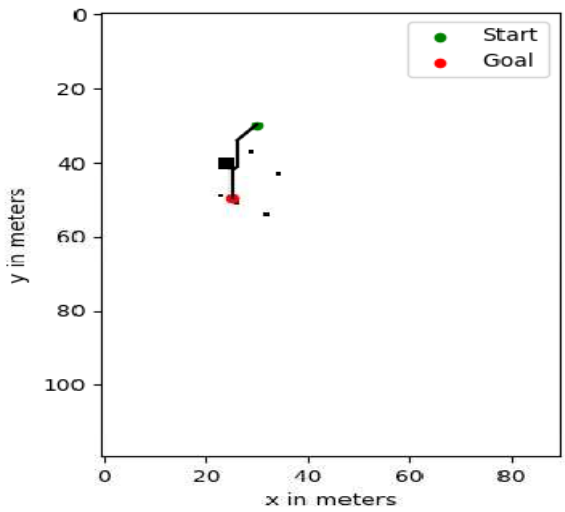

Fig. 7. Local Path Planning on a Single Frame

6 Conclusion

The evaluation of the entire studies was based on software simulation and process flow can be understood from Fig. 8. The simulation studies were performed mainly using the acquired LIDAR data, camera data and GPS data.The task involved the usage of different sensory data which contained different information for understanding the scene as captured by the sensors. Therefore,all the sensory information as a package is referred to as multi-modal sensory data. This data was used in order to achieve certain tasks within the scope of this paper which includes object detection, range detection and path planning. The objects are detected from the image obtained using the camera, the range of the objects are taken from the LIDAR sensor. Correspondence is established between the objects detected from different sensors and are projected to a 2-D global map. This 2-D global map is further used as reference for local path planning by using D* Lite path planning algorithm to generate a path from a given start position to a given goal position. The proposed system was able to generate a local path for every frame in the sample video scene for different test cases.

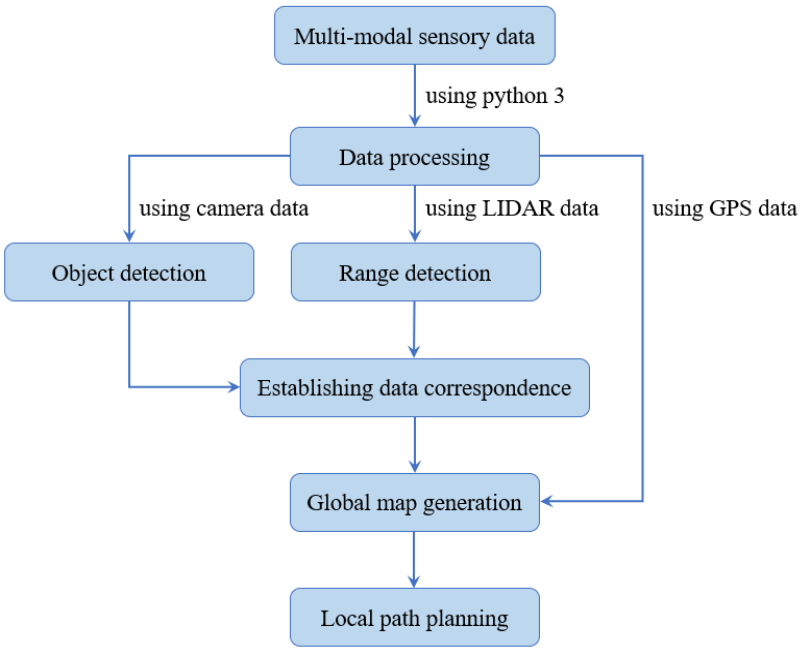

Fig. 8. Block diagram of the Proposed System

The studies presented in this paper was able to meet the expected outcome and performance metrics of the proposed system. Such metrics include reproducibility of the results under similar sensory data which proved to be valid from differenttest cases both obtained from KITTI dataset. The metrics of the system can also be universally repeated for any dataset other than KITTI dataset provided the sensory data obtained from those consist of information from camera, LIDAR and GPS taken simultaneously which was the case in KITTI dataset. The time and space complexity of the system is linear as when the number of inputs grows, the proposed system takes longer time to complete and longer memory space.

The test case usage of the multi-modal sensory data proved to be true in simulation. The same can be implemented on a real time situation, collecting real time data provided the global map is known, also considering factors such as the dynamics of the system and environment to providebetter assistive capabilities to the user at any given point of time. By the usage of dynamic path planning algorithms, the system can continuously monitor the change in position of the obstacles, which in turn helps collision avoidance.

References

1. The KITTI Vision Benchmark Suite Homepage,http://www.cvlibs.net/datasets/kitti/, last accessed 2020/04/22.
2. Andreas Geiger, Philip Lenz and Raquel Urtasun: Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3354-3361. IEEE, USA (2012).
3. Gaurav Pandey, James R. McBride and Ryan M. Eustice: Ford Campus Vision and Lidar Data Set. International Journal of Robotics Research 30(13), 1543–1552 (2011).
4. Will Maddern, Geoffrey Pascoe, Chris Linegar and Paul Newman: 1 year, 1000 km: The Oxford RobotCar dataset. International Journal of Robotics Research36(1), 3–15 (2017).
5. Object Tracking Evaluation 2012, http://www.cvlibs.net/datasets/kitti/eval_tracking.php, last accessed 2020/04/22.
6. Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun: Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 39(6), 1137-1149 (2017).
7. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar and C. Lawrence Zitnick: Microsoft COCO: Common Objects in Context. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision-ECCV 2014. European Conference on Computer Vision 2014, LNCS, vol 8693, pp. 740-755. Springer, Cham (2014).
8. Jonathan Huang and Vivek Rathod and Chen Sun and Menglong Zhu and Anoop Korattikara and Alireza Fathi and Ian Fischer and Zbigniew Wojna and Yang Song and Sergio Guadarrama and Kevin Murphy: Speed/accuracy trade-offs for modern convolutional object detectors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3296-3297. IEEE Computer Society, USA (2017).
9. Andreas Geiger, Philip Lenz, Christoph Stiller and Raquel Urtasun: Vision meets Robotics: The KITTI Dataset. International Journal of Robotics Research32(11), 1231–1237 (2013).
10. S. Koenig, M. Likhachev and D. Furcy: Lifelong Planning A*. Artificial Intelligence155(1-2), 93-146 (2004).
11. S. Koenig and M. Likhachev: D* Lite. In: Proceedings of the AAAI Conference of Artificial Intelligence, pp. 476-483.AAAI Press, USA (2002).
12. Peter E. Hart, Nils J. Nilsson and Bertram Raphael: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics 4(2), 100-107 (1968).
13. Anthony Stentz: Optimal and Efficient Path Planning for Unknown and Dynamic Environments. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp.3310-3317. IEEE, USA(1994).