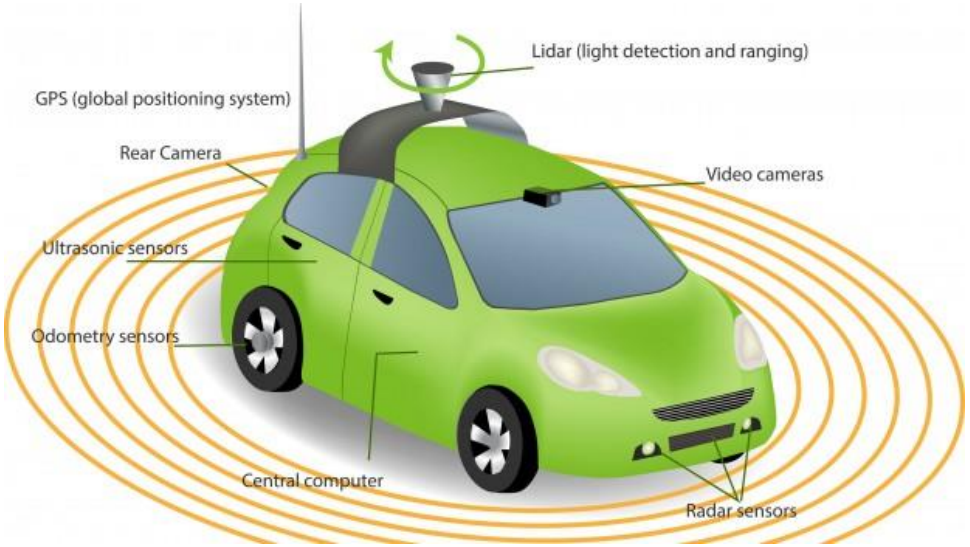
Université Saint-Joseph de Beyrouth
Faculté d'ingénierie
École supérieure d'ingénieurs de Beyrouth

LiDAR: The eyes of Self Driving Cars

9ème Journée de la Recherche

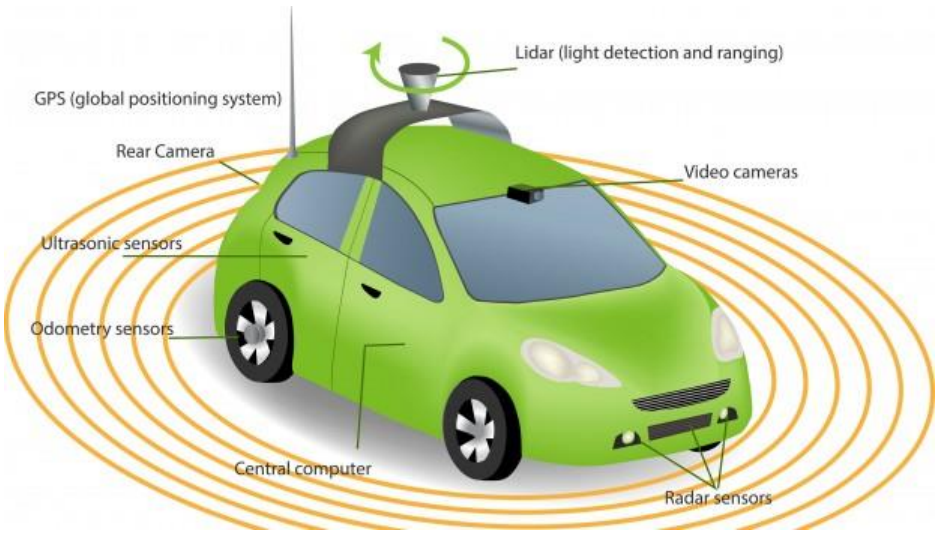


George E. Sakr
ESIB

Université Saint-Joseph de Beyrouth
Faculté d'ingénierie
École supérieure d'ingénieurs de Beyrouth

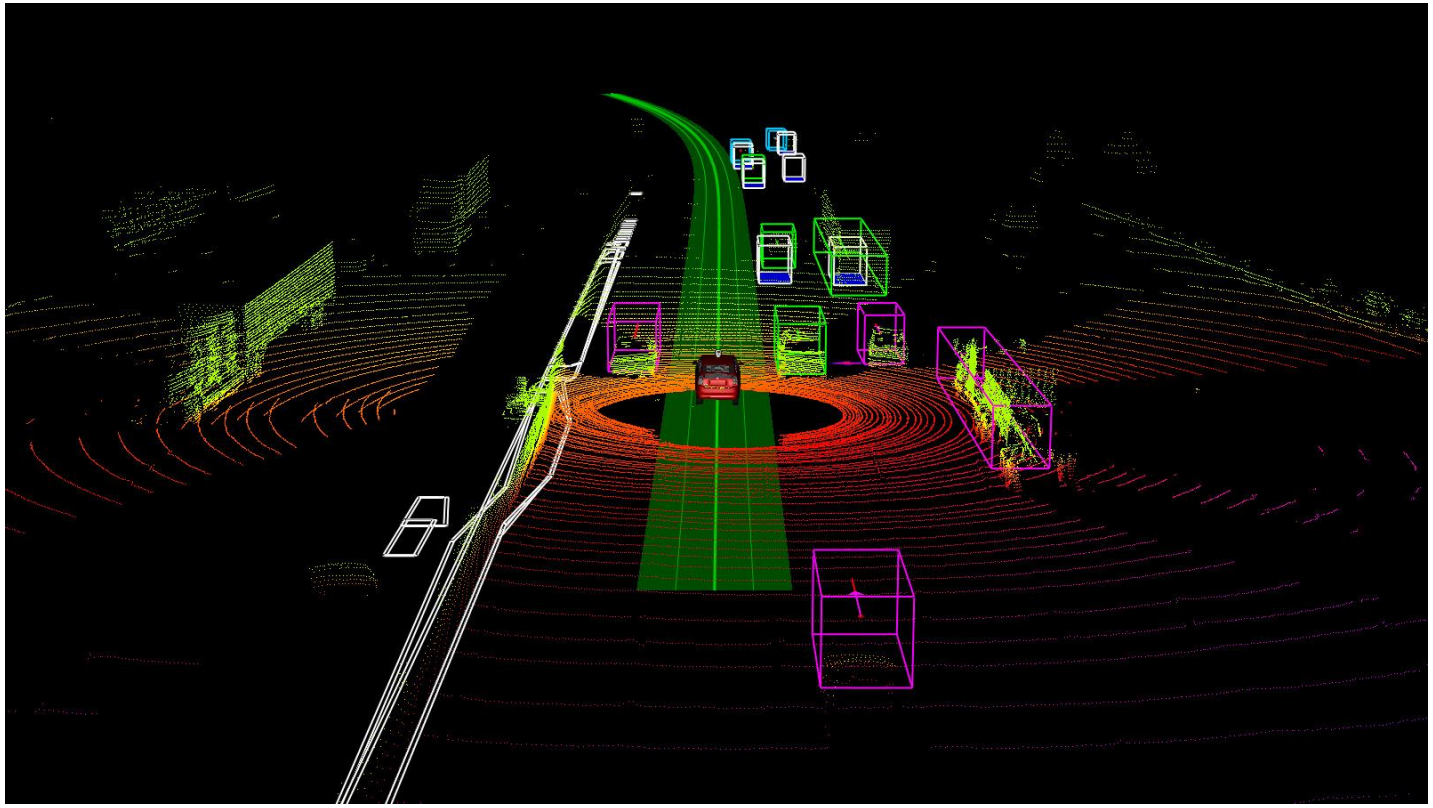
Aim

Aim?



- Radar: Radio Detection And Ranging
  - To detect moving objects using doppler effect.
- LiDAR
  - To detect static objects using amount of energy returned by road assets
- In this research we focus on static objects classification using LiDAR data.

LiDAR – Point Cloud



LiDAR and Big Data

New LiDAR can generate up to 300,000 pulses per second

Every pulse generates 4 doubles (x,y,z,r) = 4x8=32 bytes

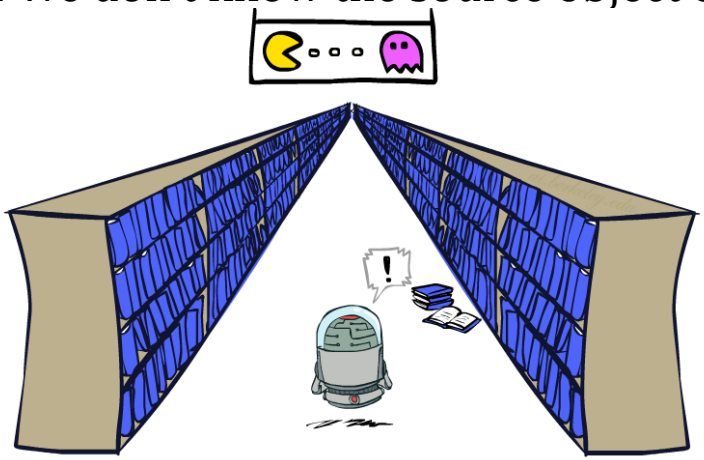
$$N = 300{,}000 \times 32 = 9{,}600{,}000\ bytes/s$$

A 1 hour drive from Jbeil to Beirut during rush hour will generate

$$N = 9{,}600{,}000 \times 3600 \sim 35GB\ per\ hour!!!$$

We end up having multiple huge text files with billion of rows. Each containing 4 values.

The data is unlabeled: We don't know the source object of each reflection



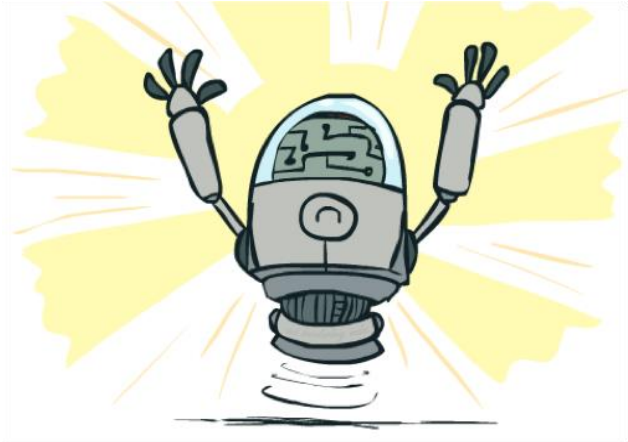
GeoJSON file to the rescue

- Civil Maps, a US based company provided is with the LiDAR data.
- They also provided us with a GeoJSON file containing a summary of the objects found on the road where they collected the LiDAR dataset.
- For example, a triangular traffic sign is represented in the JSON by the coordinate of its 3 vertices.
- Every (x,y,z) triplet that falls within this triangle is labeled as Traffic Sign.
- Similarly, road edges, center lines, bike lanes, traffic lights and many more assets were identified.

Phase 1: Data Labelling

- Write a C++ code that scans the LiDAR data and labels every row based on the JSON file.
- The code is optimized to run on the massively parallel GPUs that we have on premise.
- A lengthy process that takes up to 3 days per GB.

Phase 1 Results



We have got labelled data

Phase 2: Choosing a Classifier

- The target is to train a classifier that is able to receive LiDAR points and generate a label for all the quadruplets received.

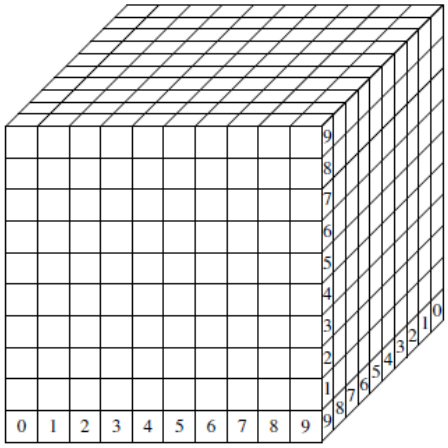
- The classifier must have a very high classification accuracy:
  - At least 90% for the initial prototype
  - At least 95% for the beta version
  - 99.9% for the final version.

Phase 2: Choosing a Classifier

- A classifier that has been showing state of the art results in 2D image classification is Convolution Neural Networks.
- CNN is optimized to run on 2D images where every pixel is made up of (R,G,B) components.
- However, our data is made up of (x,y,z) coordinates and r.
- Hence the need to transform the 3D data into an equivalent 2D image.

Phase 3: Data Preprocessing

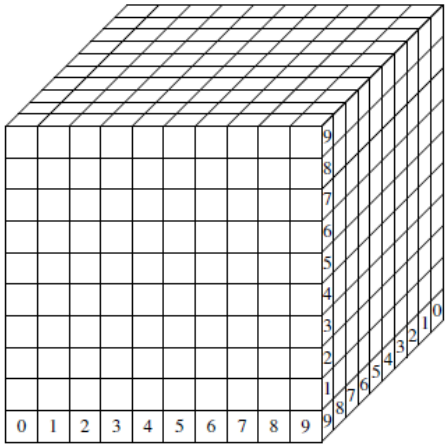
- A new method for transforming the 3D data from the LiDAR into a 2D hyper image that can be handled by CNN.



- 1 meter cube voxel divided into 1000 sub voxels ($10cm^3$) each.
- Find the LiDAR quadruplets that fall inside every sub voxel.
- Give a value for every sub voxel equal to the average reflectivity of all quadruplets that falls within its boundaries.
- Slide this voxel in x, y and z direction from the lower left corner to the upper right one. (Stride of 7cm)

Phase 3: Data Preprocessing

- Voxel Flattening



- A hyper image is created by flattening the voxel.
- Replace the value of every sub voxel in the x-y plane by an array of 10 values of the sub voxels in the z plane.

Phase 3: Data Preprocessing-Results

- We end up representing every second of LiDAR data with a series of flattened hyper images
- Why we call them hyper image: because they are 2D images where every pixel is now represented by 10 components instead of only RGB values.
- These hyper images represents the input of the CNN classifier.

Phase 4: Training

- Now the easy part ☺
- Design a deep architecture based on CNN for hyper image classification.
- The design process is a sequence of trial and error until we end with a classifier that satisfies our criteria.
- The data is split into 3 parts (Training (74%), Validation (7%) and testing (19%))
- The classifier is trained on the training set and its parameters are tuned to give the highest accuracy on the validation set. The reported accuracy is only on the testing test.

Phase 4: Training 6 classes

Train a classifier to differentiate between:

- Bike Lane
- Lane Centerline
- Road Edge
- Yellow Single Solid Line
- Yellow Single Broken Line
- Null: Nothing detected

- These are essential for the car to drive on the road and not exceed the road limit.

Phase 4: Training 6 classes

Data

- 9246 images of each class.
- Shuffle the images
- Pick randomly 55473 images for training
- Pick randomly 5547 images for validation
- The remaining 13869 for testing.
- Make sure that the training, validation and testing sets are perfectly balanced.

Phase 4: Training 6 classes

The best classifier has the following architecture

- 4 convolution layers in cascade (ReLU activation functions in all)
  - Layer 1: 16 kernels each of size 3 x 3
  - Layer 2: 32 kernels of size 3 x 3
  - Layer 3: 64 kernels of size 3 x 3
  - Layer 4: 128 kernels of size 3 x 3.
  - Dense layer 1: size of 512 with a ReLu activation
  - Dense layer 2: size 6 and a softmax activation function

- Adam optimizer was used with a batch size of 128

Phase 5: Testing 6 classes

The best classifier applied to the 6 classes yielded the following per class accuracy:

- Bike Lane: 90.8%
- Lane Centerline: 90%
- Road Edge: 90.1%
- Yellow Single Solid Line: 91%
- Yellow Single Broken Line: 90.6%
- Null: Nothing detected: 91.1%

- With an overall accuracy of 90.6%.

Phase 5: Testing 6 classes

The best classifier applied to the 6 classes yielded the following confusion matrix:

	Bike lane	Center Line	Road Edge	Yellow single	Yellow Brocken	Null
Bike Lane	90.8%	8.2%		1%		
Center Line	8%	90%		2%		
Road edge	5.9%	4%	90.1%			
Yellow single				91%	2.5%	6.5%
Yellow Brocken				2.2%	90.6%	7.2%
Null	2.3%	3.1%	0.5%	0.6%	2.4%	91.1%

Conclusion

- We were able to achieve the accuracy for the prototype phase.
- Next step is to include more data from different sources to get a better accuracy
- Develop a more complex network for classification
- Classify more static objects.

Conclusion

- We were able to achieve the accuracy for the prototype phase.
- Next step is to include more data from different sources to get a better accuracy
- Develop a more complex network for classification
- Classify more static objects.
- Use 3D CNN

Questions

E-mail: georges.sakr@usj.edu.lb