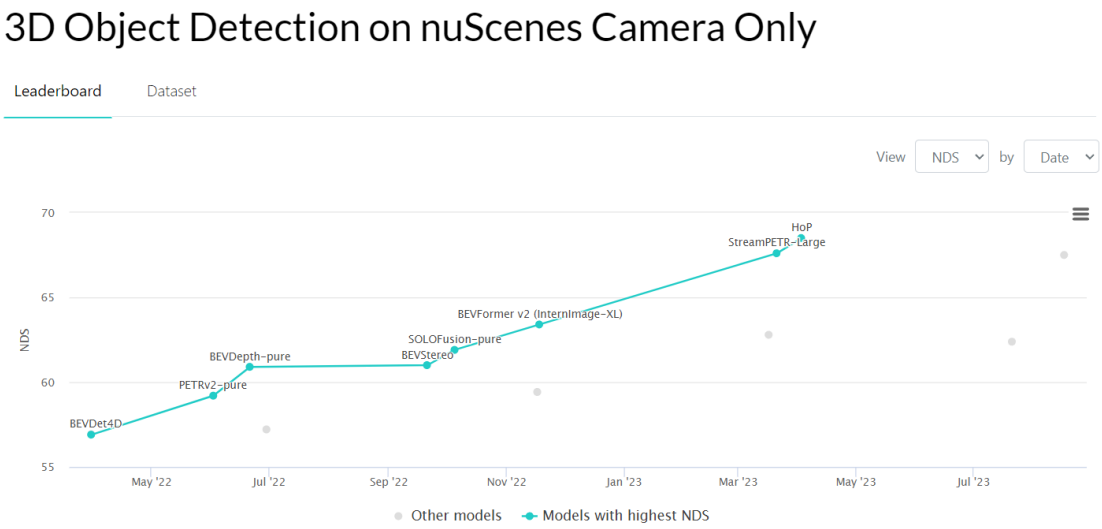
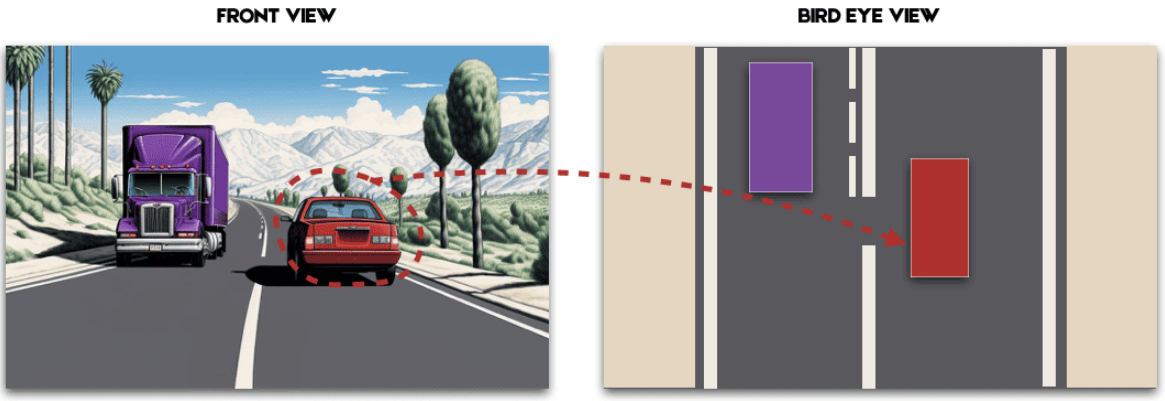
Post 2021, when Tesla switched to Hydranets, Camera-only solutions are becoming more and more popular. Among Camera-only solutions, Birdseye view (BEV) based approaches have almost been the go-to approach, as seen in the nuScenes Camera only 3D Object detection leaderboard. For example, as of submission in August 2022, the gap between first-ranking camera-only and LiDAR methods exceed 20% on nuScenes dataset and 30% on Waymo benchmark



What is meant by the BEV approach? What is the reason for it to be adopted by all top performing models? Let's find out.

What is BEV?

BEV (Birds Eye View) is nothing but a top-down view of the physical world . It's mostly expressed in euclidean space (in metres), and can be a 2D / 3D coordinate frame.



Fundamentally, Perceiving the world in Bird's eye view offers three advantages over Perspective view in Vision-based systems

- No scale problem - Scale problem refers to the issue of objects of same size, appearing differently, based on the distance from the camera. This consistent relationship with the 3D world makes the algorithm robust and generalized.
- Occlusion representation - Representing objects with occlusion or cross traffic is easier in BEV space. Also, It is convenient for subsequent modules (e.g. planning, control) to develop

and deploy. The methods to get to BEV space are flexible in that you can use them to cover part of the entire FOV, or entire 360 degree based on the application.

- Fusion opportunities - As we'll explore in later stages of the blog, BEV space serves a more direct method for Spatial as well as Spatio-Temporal fusion, particularly, Lidar-Camera fusion

Now that we've seen the advantages of BEV or PV, let's explore some of the techniques to get BEV of the environment.

How to get a BEV representation of the world?

In the context of Self-driving cars, Cameras, Lidars and Radars are the commonly used Perception sensors. Among these, Camera and Lidar have attracted a lot of research among Deep Learning enthusiasts. Camera sensors operate mostly in 2D Image plane, whereas Lidars operate in 3D space. On the other hand, images capture more semantic information, compared to sparse point clouds. The methods to process the sensor data varies significantly due to the inherent contrasting nature of sensor data. Hence, its sensible to categorise the methods based on the sensor modalities as follows:

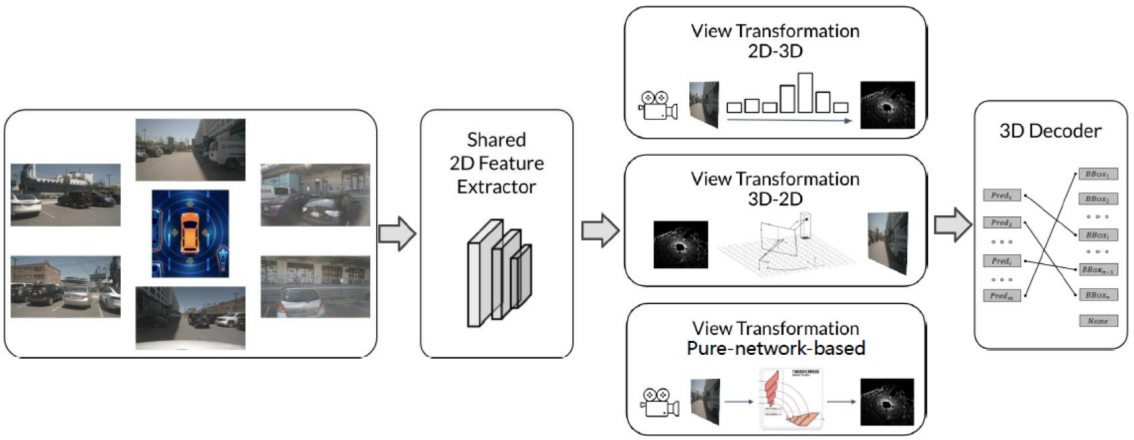
- BEV Camera
- BEV Lidar
- BEV Fusion

BEV Camera

BEV Camera indicates vision only or vision-centric algorithms for 3D object detection or segmentation from multiple surrounding cameras; Eg: BEVFormer. The very essence of designing a camera-based pipeline such that it outperforms LiDAR, is to understand the view transformation process from a 2D, appearance input to a 3D, geometry output.

Any Camera based 3D Perception consists of three core components - Features extractor, View Transformation module and 3D decoder. The goal of the Feature Extractor module is to extract rich features from input 2D image space that embed the real-world information. The Feature extractor module is very mature, thanks for huge experience in 2D perception, resulting in number of pretrained backbones, including

- CNN based backbones: Resnets, Regnets, EfficientNets etc
- Transformer based backbones: Vision Transformer, Swin Transformer, Segformer etc



source

View Transformation module

- The View Transformation module largely differentiates 2D perception from 3D perception.
- Note that you can have a 3D perception stack without the View transformation module.
- The View transformation module can be categorized into following 3 types:
    - 2D-3D methods
    - 3D-2D methods
    - Pure Network based methods

$$\mathcal{F}_{3D}(x,y,z) = M_{trans}\left(\mathcal{F}^{*}_{2D}(\hat{u},\hat{v}), \left[\boldsymbol{R} \quad \boldsymbol{T}\right], \boldsymbol{K}\right)$$

3D / voxel feature

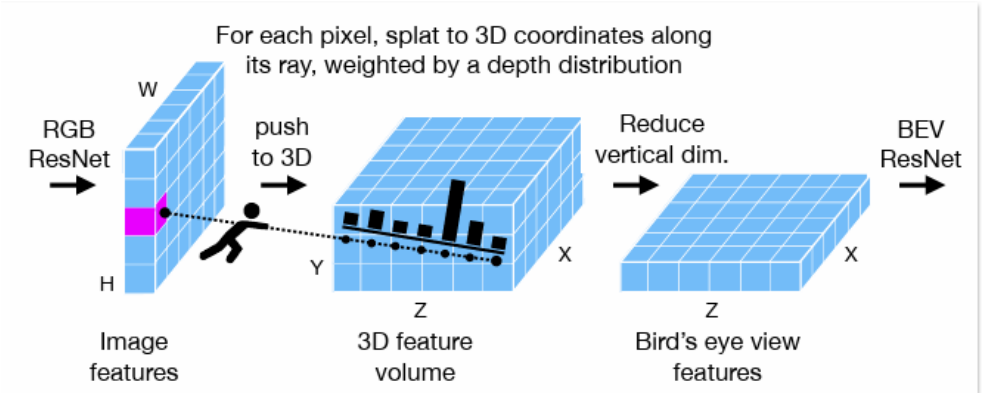
View Transformation module

2D Image features

Camera Intrinsics, extrinsics

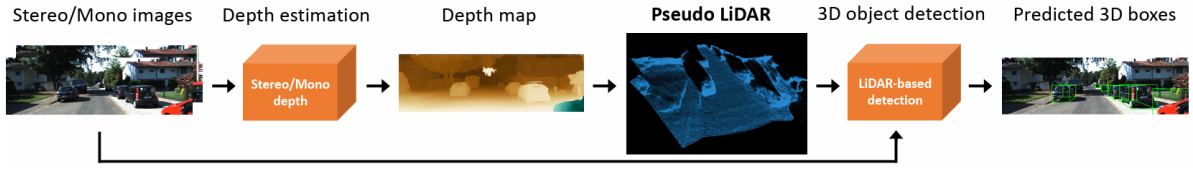
2D-3D methods

- Such methods use 2D image features as input and “lifts” 2D features to 3D space via depth estimation. Usually, these methods predict a grid-wise depth distribution on 2D features, then“lifts”the 2D features to voxel space based on depth.



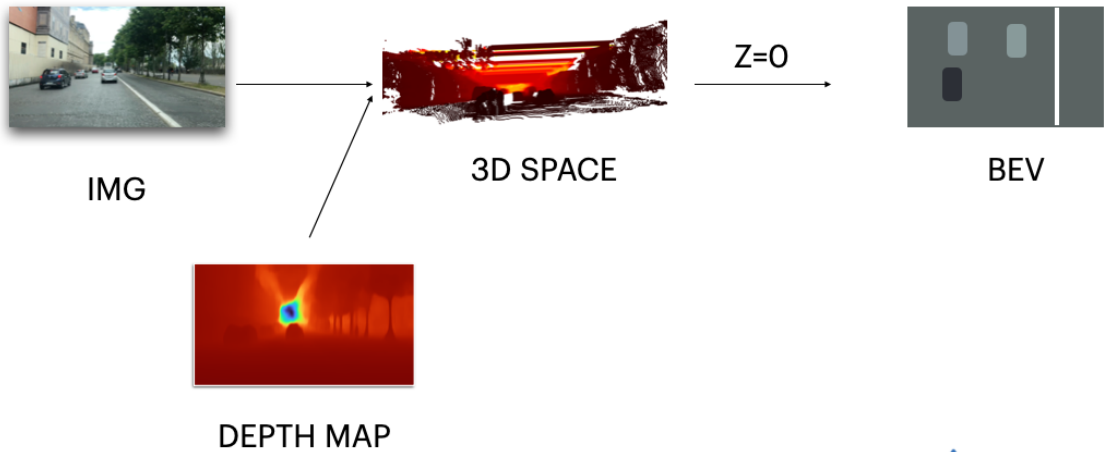
source

- Pseudo-Lidar methods such as Pseudo-Lidar and Pseudo-Lidar++ models achieve the same, but predict the depth information is extracted from a pretrained depth estimation model and the lifting process occurs before 2D feature extraction. Supervising the depth prediction using ground truth depth improves the network’s performance considerably.

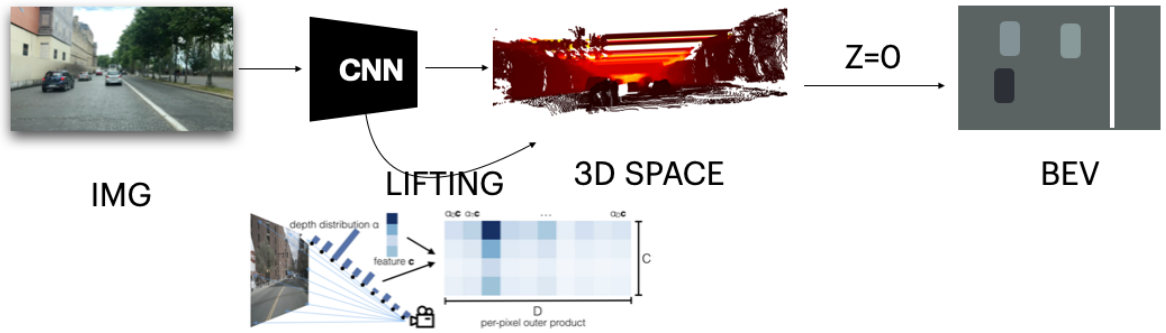


[source]

Surya J My understanding is this for both papers:



Just that on one case, it uses a Depth Map, on the other case, it uses LSS with depth distributions. Let me know if you understand the same thing.



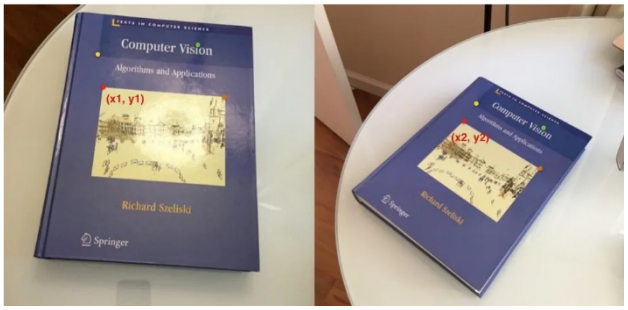
3D-2D methods

Traditional approach

In Computer Vision, homography (H) is a transformation matrix, which when applied to a Plane (in our case, Image), maps to another plane. Using the homography matrix, we can go from one perspective to another, just as shown below.

$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Image plane2 | Homography matrix | Image plane2

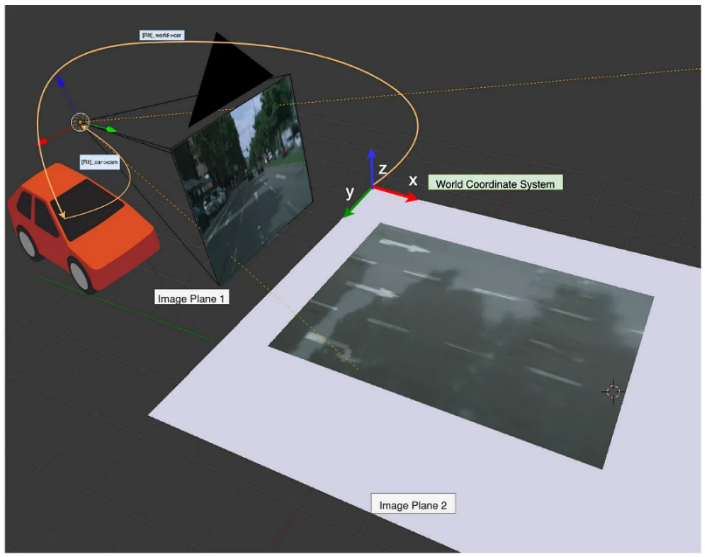


source

The method of producing a birds-eye view image of the scene from the front-facing image plane is referred to as Inverse Perspective Mapping (IPM). Assuming the world to be flat on a plane, we need to map each pixel on image to a point on a 2D plane. To understand better, we define the following coordinate frames:

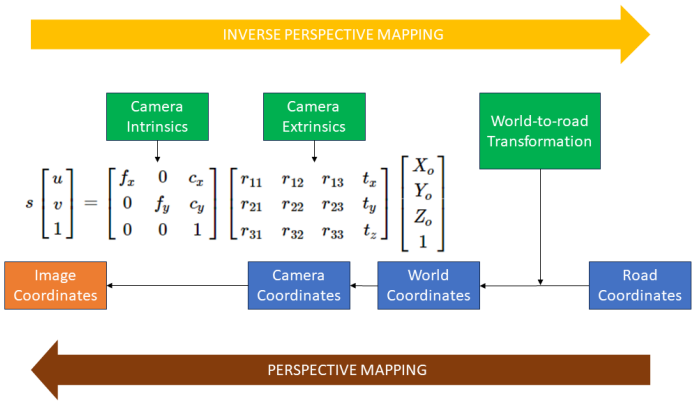
- Image Coordinate frame (in pixels)
- Camera Coordinate frame
- World Coordinate frame

NOTE: The world coordinate frame is often discretized (in metres). Hence an approximation on resolution is to be made.



source

Mathematically, inverse perspective mapping can be represented as follows:



Once we have the Camera intrinsics and extrinsics, IPM procedure can be summarized as follows:

1. Take a point in the Road plane (X, Y, Z=0).
2. Apply the projection matrix (comprising of World to road, camera intrinsics and extrinsics) to get image coordinates (in pixels)

3. Find the pixels, closest to image coordinates and project to the BEV plane.
4. Repeat steps (1-3) for all points in discretized road plane. Additionally, we need to perform some form of interpolation to prevent holes in the output.

Sounds straightforward. It might, but IPM makes some assumptions about the environment, which limits its real-world applications.

Drawbacks of IPM

1. Flat road surface assumption. Any deviation would produce artefacts / distortions
2. Zero height object assumption. Objects with height would appear distorted / warped in BEV output
3. IPM works well only in areas, closer to ego vehicle
4. IPM works well only for static objects. So, dynamic objects are generally filtered before computing the Homography transformation

Following image highlights the above issues

source

Even with the above limitations, IPM is still commonly useful in specific environments with static objects, mostly for Lane detection algorithms.

Geometric prior based Deep Learning approach

The Geometric prior to project 3D world points to 2D image planes can be combined with Neural networks for View Transformation modules. Spatial Transformer Networks are a special class of Neural Network, that help CNN architectures be more robust to spatial variations (Rotation, Translation, Shearing etc). For example, in the below example from German Traffic Sign Recognition Benchmark Dataset (GTSRB), we can see that the network learns to zoom on to the traffic sign, irrespective of its initial distortion / orientation

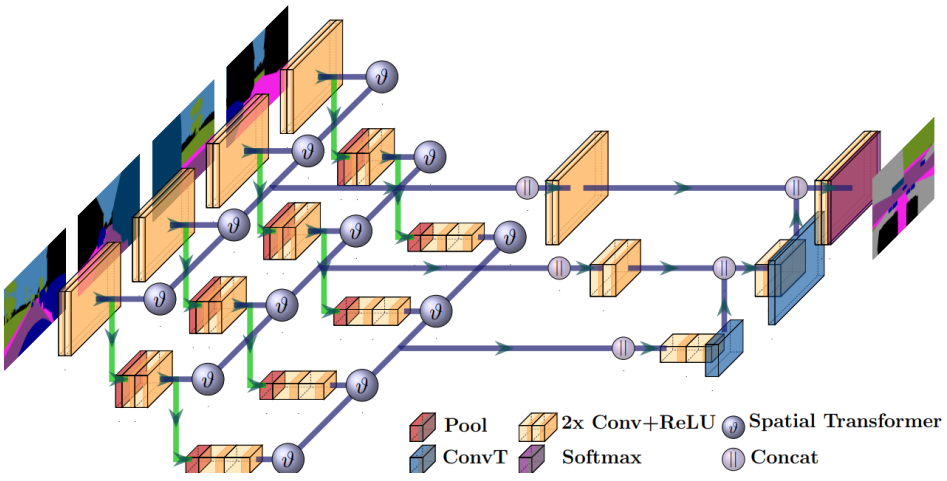
source

Cam2BEV provides a simple, yet efficient framework to convert

- A single front-facing image to BEV image
- Combine multiple front-facing images to get 360 degree BEV image

UNetXST, the model in the paper, is derived from the UNet model, which was developed for Biomedical Image Segmentation. It's similar to the UNet model in following ways:

- CNN model - It uses Convolution + MaxPooling blocks to extract features
- Encoder-Decoder architecture - The input is downsampled for `n` times in the encoder, while being upsampled in the decoder
- Skip-connections - Feature map from the encoder is concatenated at each layer in decoder, to improve boundaries segmentation



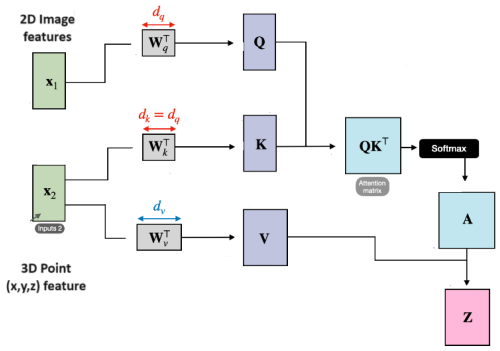
source

UNetXST has following additional components

- Extension (X) - The original UNet model was for single image semantic segmentation. Here, its being extended to multiple images
- ST stands for Spatial Transformers

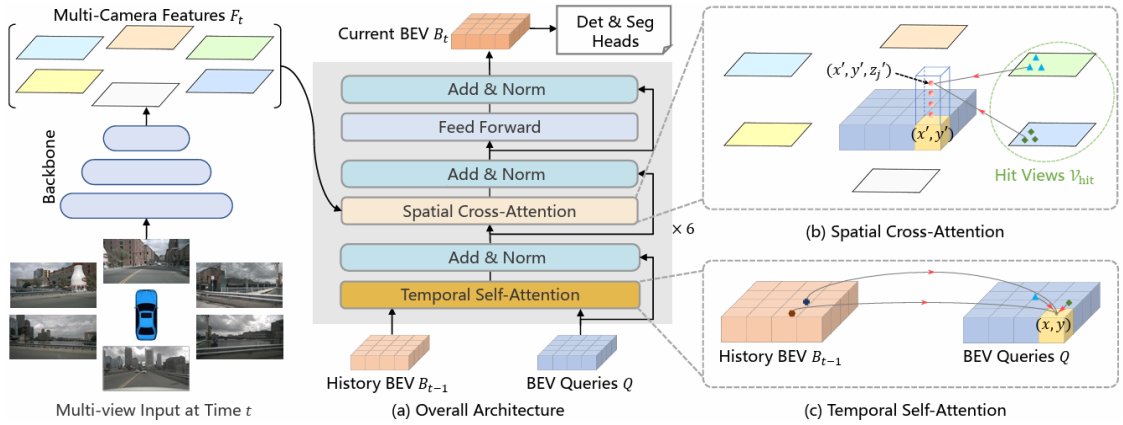
With a single image input, we use the encoder module to get Feature maps of varying resolution. Traditionally, these feature maps capture features in the frame that image was taken - front view. The spatial Transformer (ST) modules convert the features to BEV space. For multi-view inputs, the BEV projected features are combined using Conv-BatchNorm-ReLU blocks to get 360deg features at each depth. The decoder module, up samples features from each scale, concatenates with encoder output (skip-connection) and finally predicts a 360deg semantic segmented output.

Cross-attention mechanism in transformer architecture conceptually meets the need of such a geometric projection. The 2D Image features are queried with 3D point as key, value to get 3D voxel features.



source

Models such as DETR3D, BEVFormer, GitNet employ such attention mechanisms to enrich the 3D Voxel features. As using cross-attention for all querying all 2D images is very compute intensive, most models use efficient implementations such as Deformable attention, grid samplers.

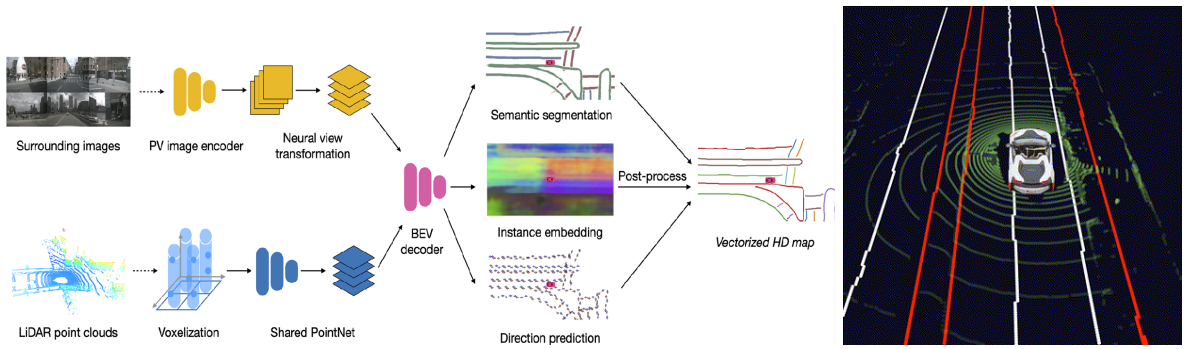


source

Note that these methods use some form of geometric prior for the 3D-2D transformation.

Pure Network Methods

Some methods depend on Neural Networks entirely, to learn the implicit camera projection relationships. Most BEV Map Segmentation models employ a MLP or Transformer based architecture to regress the transformation values.



source

The Neural view transformation module in the HDMapNet architecture is one such MLP module that does the same. BEVSegformer uses a combination of MLP and Transformer modules to semantically segment lane lines in BEV image space.

Initial works on Camera only 3D perception focussed on improving 3D localization from Perspective View (2D image space). After moving to BEV space, there is a remarkable improvement in localization accuracy of the models. Fundamentally, moving to BEV space solves the scale problem - Objects of the same size appear differently in the Image plane, based on the distance of the object from the camera.

3D Decoders

- The 3D decoders receive the feature in 2D/3D space and output 3D perception results such as 3D bounding boxes, BEV map segmentation, 3D lane keypoints and so on.
- Most 3D decoders come from LiDAR-based methods which perform detection in voxel-space/BEV space, but there are still some camera-only 3D decoders that utilize the feature in 2D space and directly regress the localization of 3D objects

Advantages

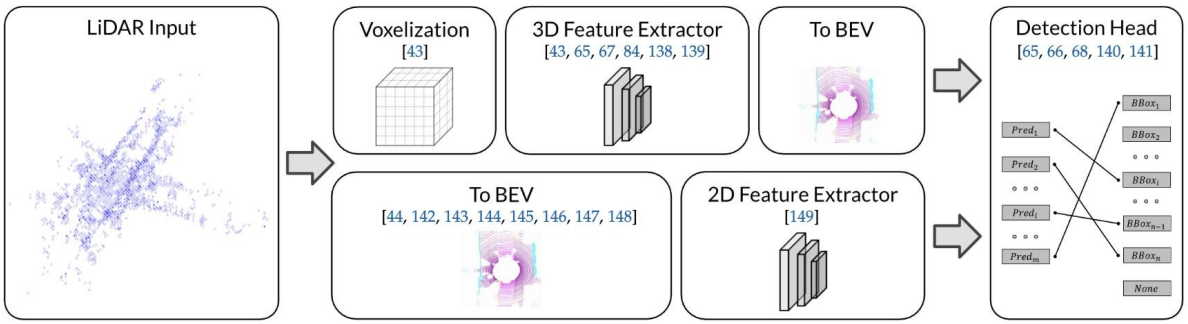
- Recognize long-range distance objects
- Detect color-based road elements (e.g., traffic lights)
- Performance is superior compared to Perspective view based methods

Disadvantages

- Performance is inferior to Lidar-based and Fusion-based models.

BEV Lidar

BEV LiDAR methods use only point cloud data for the detection or segmentation task. Features are extracted from the point cloud using a combination of point based and voxel based modules. The Voxel-based methods discretize the point cloud, aggregate features, mostly using Sparse 3D convolutions. The point-based methods use variants of PointNet to extract point cloud features. In both the branches, the features are transformed to BEV space at some point

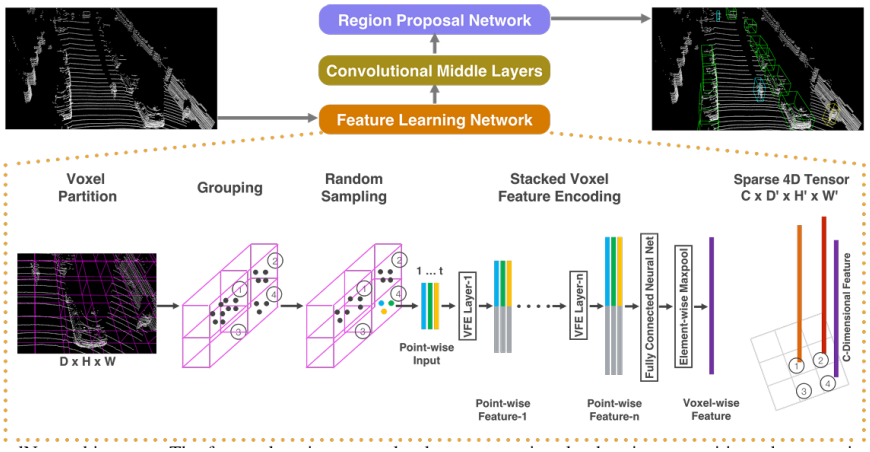


source

Based on whether input for Feature extractor backbone, these methods can be categorized into two categories - pre-BEV and post-BEV.

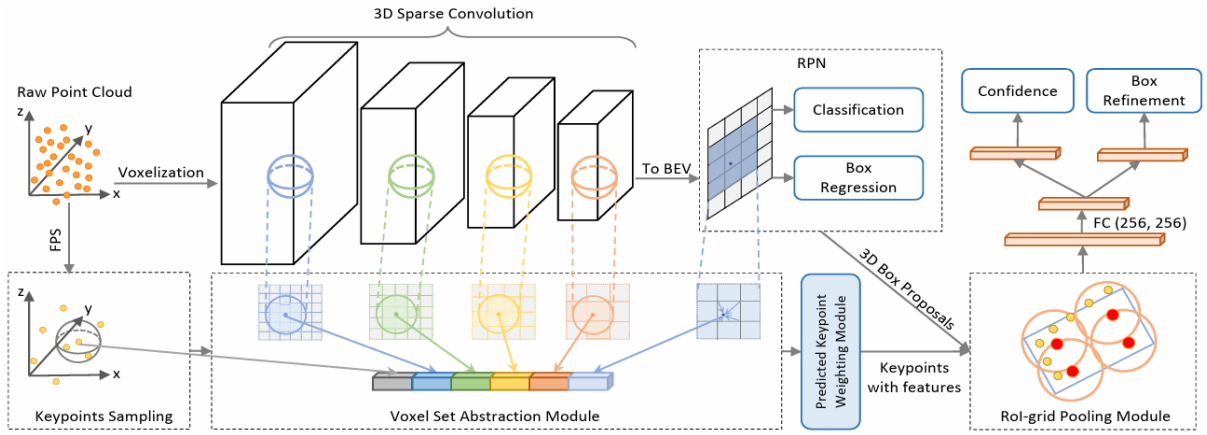
Pre-BEV Feature Extraction

In this method, the backbone extracts features from the point cloud using point-based or voxel-based methods and then converts the extracted features to BEV space to be used by Detection Heads. VoxelNet, a Voxel-based method, stacks multiple voxel feature encoding (VFE) layers to encode point cloud distribution in a voxel as a 3D voxel feature. After merging the dimension of channel and height, the feature maps, which are transformed implicitly into BEV, are processed by a region proposal network (RPN) - 2D CNN module to generate object proposals.



source

PV-RCNN combines point and voxel branches to learn more discriminative point cloud features. Specifically, high-quality 3D proposals are generated by the voxel branch, and the point branch provides extra information for proposal refinement.



source

Newer approaches like Voxel Transformer, SST and AFDetV2, use techniques such as Local Attention, Dilated Attention, Sparse Regional attention for efficient querying of BEV features.

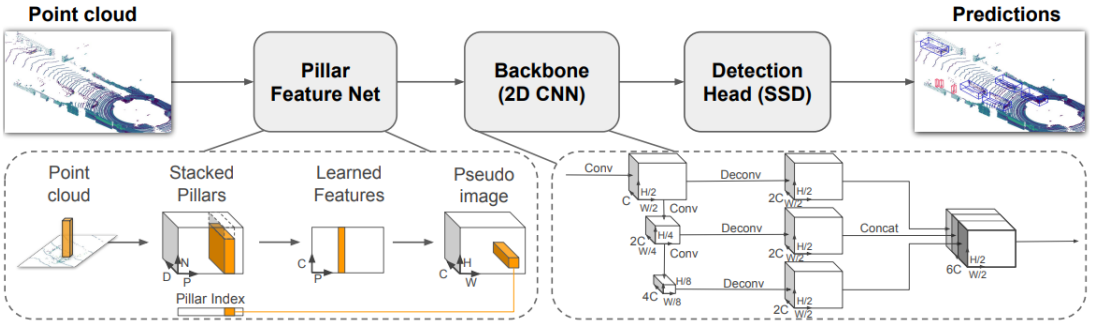
Post-BEV Feature Extraction

In Spite of using sparse convolutions, 3D convolutions are compute intensive for industrial applications, which have real-time processing constraints. Converting to a BEV grid in early stages causes considerable loss of information.

MV3D is the first method to convert point cloud data into a BEV representation. After discretizing points into the BEV grid, the features of height, intensity, and density are obtained according to points in the grid to represent grid features.

The raw point cloud is converted to a stacked pillar tensor and pillar index tensor. The encoder uses the stacked pillars to learn a set of features that can be scattered back to a 2D pseudo-image for a convolutional neural network. The features from the backbone are used by the detection head to predict 3D bounding boxes for objects. Note: we show the car network's backbone dimensions.

PointPillars introduced the concept of pillar, which is a special type of voxel with unlimited height. The encoder uses the stacked pillars to learn a set of features that can be scattered back to a 2D pseudo-image for a convolutional neural network. The features from the backbone are used by the detection head to predict 3D bounding boxes for objects.



source

Though the performance of PointPillars is not as satisfactory as other 3D backbones, it and its variants enjoy high efficiency and thus are suitable for industrial applications.

Advantages

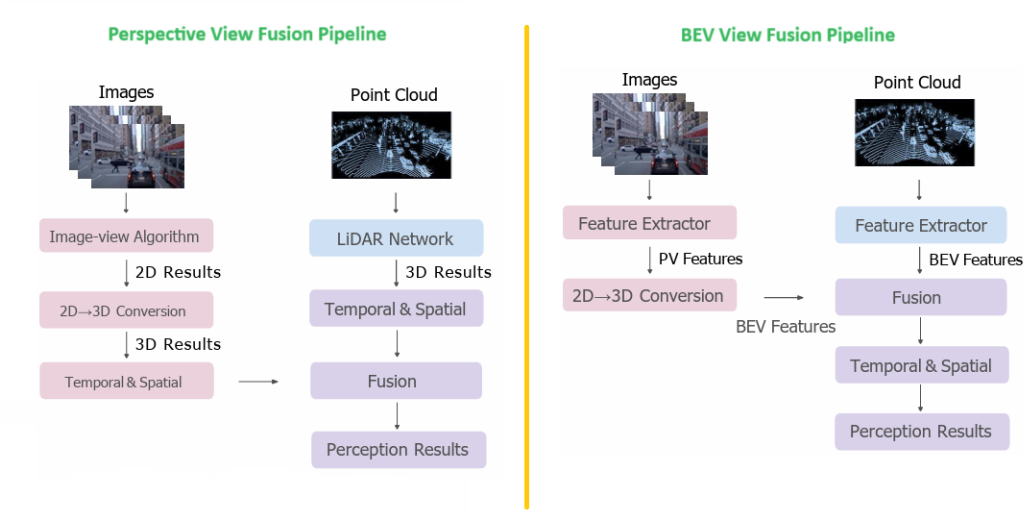
- Lidar based methods offer better Localization accuracy compared to Camera-based methods owing to built-in depth information from point cloud data.

Disadvantages

- Processing point cloud data in the continuous 3D space is time consuming, even with techniques such as Sparse convolutions.
- Post-BEV methods such as PointPillars are more efficient, but with reduced performance.

BEV Fusion

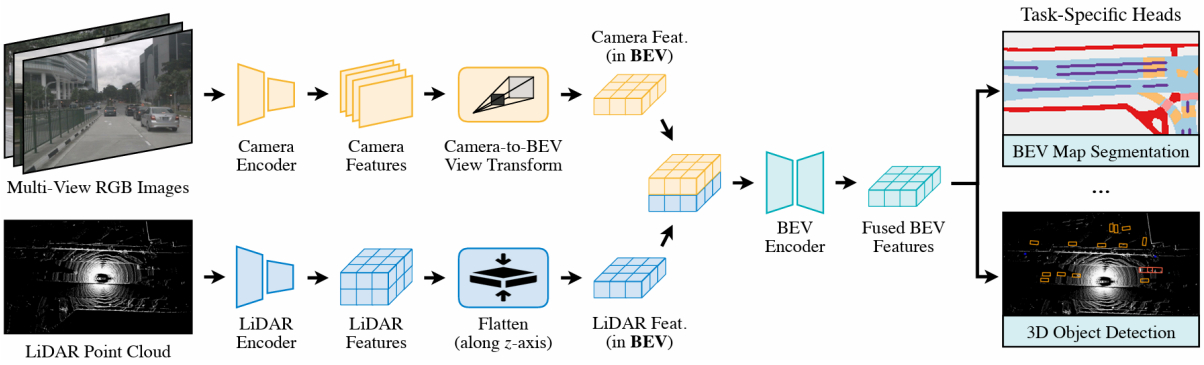
Fusion in Perception generally refers to Lidar-Camera fusion, particularly in the context of Deep learning in Self-driving cars. BEVFusion is easier as camera and LiDAR data can be projected to a common BEV space. Feature extractors vary, according to whether Fusion is done in the Perspective View or Bird's Eye View space. Following image shows the general pipeline in both views



source

In the PV perception pipeline, results of different algorithms are first transformed into 3D space, then fused using prior or hand-craft rules. The BEV perception pipeline first transforms PV features to BEV, then fuses features to obtain the ultimate predictions, thereby maintaining most original information and avoiding hand-crafted design. Thus, BEVFusion offers more opportunities for early fusion compared to Perspective Fusion based techniques

BEVFusion model uses an efficient camera-to-BEV transformation method, which projects the camera features into BEV and fuses with Lidar BEV features using Convolutional blocks. Important point to note is that the BEV space is shared between the Camera and Lidar features spaces.

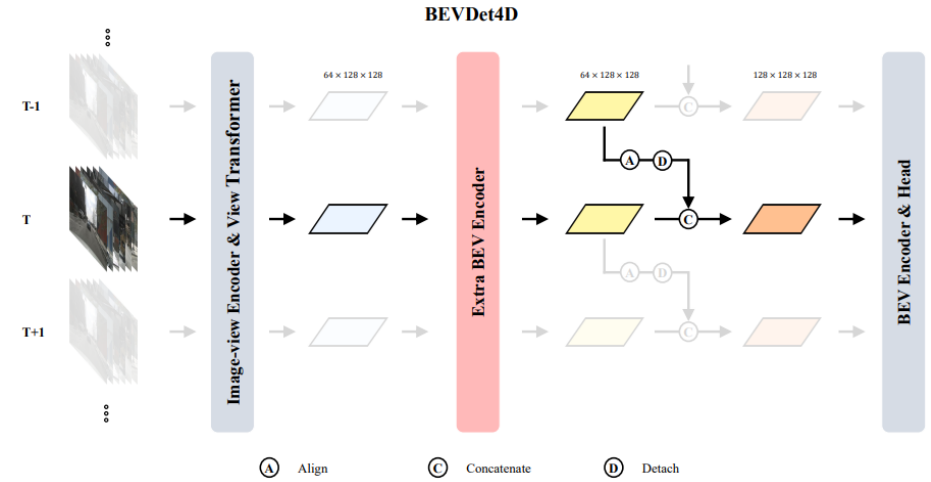


source

Temporal Fusion

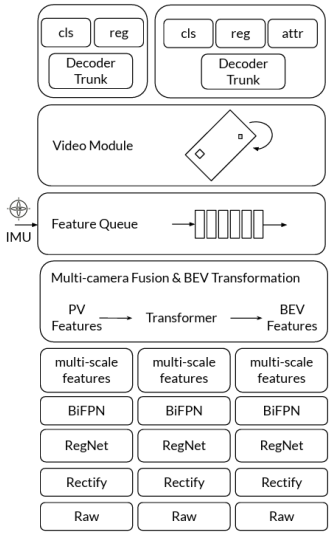
Temporal information plays an important role in inferring the motion state of objects and recognizing occlusions. BEV provides a desirable bridge to connect scene representations in different timestamps, as the central location of the BEV feature map is persistent to the ego car.

BEVDet4D retains the intermediate BEV feature of the previous frame and concatenates it with the ones generated by the current frame. Before that spatial alignment in the flat plane is conducted to partially simplify the velocity prediction task.



source

Temporal fusion can be achieved using a variety of modules like Transformers and RNN modules. For example, BEVFormer and Uniformer models use soft-attention to fuse temporal information, while Tesla was previously using Spatial RNN modules.



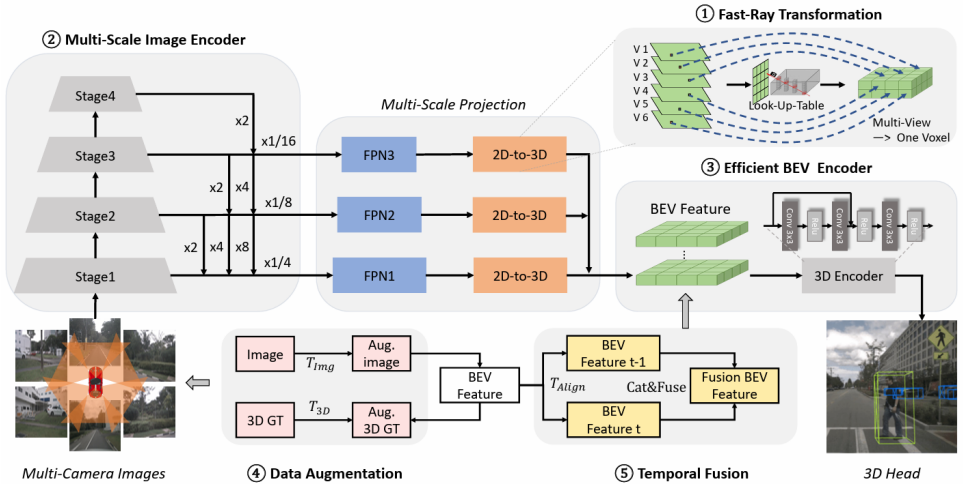
[source]

Research Trends in BEV Perception

1. Depth Estimation - Core issue for Vision-based BEV Perception is accurate depth estimation. Methods to improve that using a combination of pseudo-lidar generation, Lidar-camera distillation, Stereo-disparity, and temporal information are most sought after. Another important direction is usage of Lidar information (eg: depth supervision) during training.

2. Generalization capability of model - Domain adaptation is one of key metrics, as BEV Perception is costly, in terms of training, data annotation, testing, compute resources etc. Ideally, models trained on one dataset should be generalizable to other datasets, with very little modification.

3. Real time Perception - Usage of compute intensive blocks like attention modules, makes Perception models unsuitable for real world applications with Real time processing needs. E.g: [FastBEV] is one of recent approaches that uses a fast ray transformation method to predict the 3D voxel space from 2D image space. It is similar to a lookup table, avoiding the need to expensive operations such as Attention (Transformers) or Depth estimation (pseudo-lidar methods)



[source]

When should they use it?

Having seen the different methods to achieve BEV, its advantages and disadvantages, when should one opt for it? As with most topics in Self-driving cars, it mostly depends on the application. But here are some pointers on when you should use it.

- Require best performance, particularly with Camera only system
- Lots of sensors available with different views and you want 360deg monitoring
- Lesser constraints on Compute power
- Okay to spend on Ground Truth generation

Birdseye View offers a unified framework to create detailed representations of Ego vehicle's surroundings with following advantages:

- Flexible representation - We can use the BEV framework to have 2D / 3D / 4D representations of the world, in discretized form, with minimal change in architecture.
- Modular Fusion - We can use the same framework for Multi-Camera Fusion, Lidar-Camera fusion, and even extend it to Spatio-Temporal Fusion.
- Efficient - We can have a single model to process all sensor data, saving on computation power, closer to real-time processing.
- Rich information – Boundaries of different objects and infrastructure in top view, is very useful for Path Planning, Mapping applications.
-

Traditional method code?

Although BEV has so many advantages, it has its own set of challenges. Cameras are mostly mounted on ego vehicles, parallel to ground and facing outwards. The images are captured in a Perspective View (as seen in the Front View of 1st image), which needs to be transformed to BEV. Let's explore how it's done.

Approaches to Birdseye view

As with any Computer vision task, Birds eye view can be achieved via Traditional image processing as well as Deep Learning. In either approach, the input is a set of images in the sensor perspective. First, we'll look at traditional methods.

Deep Learning based approach

The Deep learning approach involves using Neural Networks to solve the aforesaid issues, by learning from Ground truth data. Most DL approaches combine the BEV transformation with either Object detection and /or Segmentation. This is different from the traditional approach, where we were trying to get the semantic texture information in BEV. We'll look at one such architecture from context of BEV Semantic Segmentation

UNetXST

To convert the features from this view to BEV, we use Spatial Transformer networks

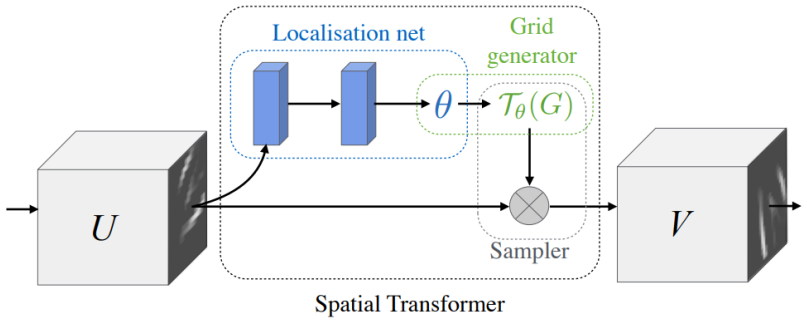
Spatial Transformer Networks (STN)

So, how is the network able to focus on the specific parts in the dataset?

As seen in image below, the network architecture consists of 3 main pieces:

- Localisation network
- Grid Generator

- Sampler



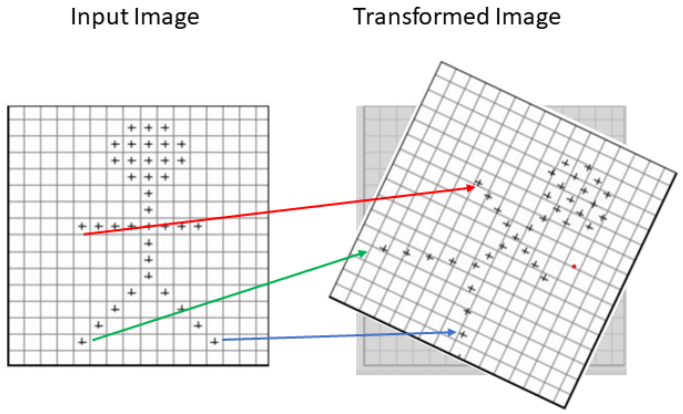
source

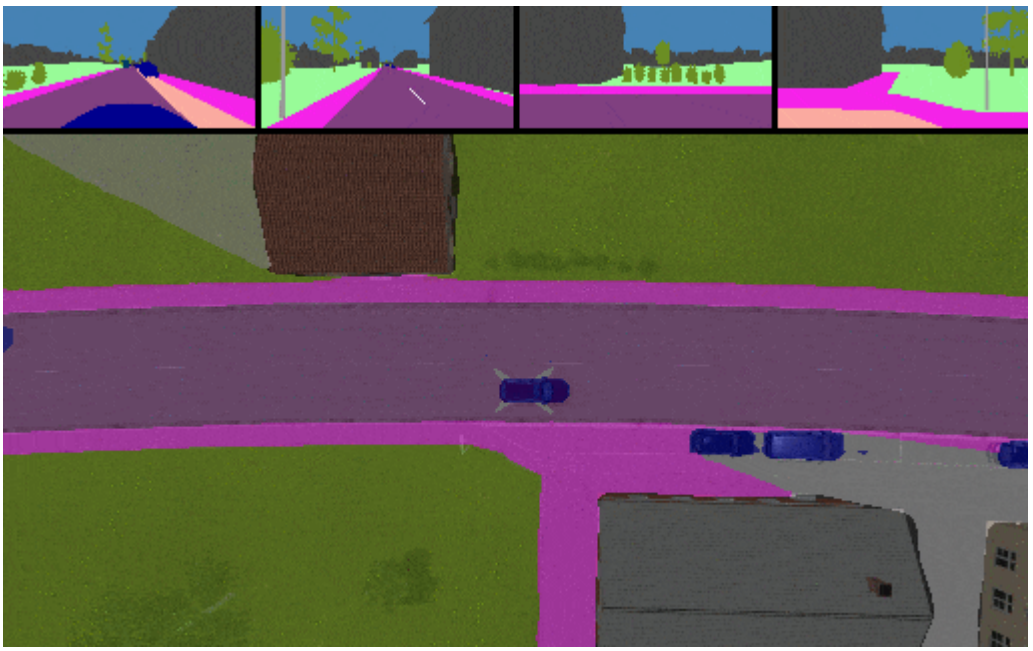
Localization Network

The Localization network tried to regress the 6 parameters of the transformation (2x3) using a combination of CNN + Fully connected layers. This ensures that the transformation depends on input data and is not fixed, as was the case with the traditional IPM method.

Sampling Grid

Once we have the transformation parameters,





The Encoder and Decoder modules are common

1. architecture
    a. Spatial Transformers
    b. Simulated data usage
    c. BEV segmentation from 4 cameras notebook
    d. https://www.youtube.com/watch?v=TzXuwt56a0E
2. SOTA Approaches to BEV
    a. BEVFormer Preview
        i. Deformable DETR module
        ii. BEVFormer architecture
        iii. Temporal Extension possibilities

References

Dummy

Such Perception tasks, require fusion of data from different sensors–Monocular, Stereo Cameras, Lidars, Radars etc, for redundancy purposes. As highlighted in previous post, there are multiple ways to fuse sensor data, each with its advantages and disadvantages.