DiffTransBEV: Improved Bird-Eye-View representation using Diffusion

Hari Veeramallu[1]

***Abstract*— In the current age, autonomy in robotics is at the pinnacle of development. One of the key aspect to attain autonomy is to perceive the environment around it. Particularly for Autonomous Vehicles, sensors are required to perceive the surroundings and generate a set of conditions for the motion planning module to perform a safe drive. Given that the surrounding environment cannot be perceived by one single sensor, the output of multiple sensors must be fused together. The sensor fusion has a certain cost and unwanted errors due to the noise within each sensor output. One such scenario where the sensor fusion noise is critical is during the generation of Birds-Eye-View (BEV) representation. BEV representation is important as it is the foundation block of motion planning, vehicle control, and motion prediction in autonomous vehicles. Diffusion models have the ability to denoise noisy samples. Currently, only one work has been done using Diffusion to generate BEV representations to date, DiffBEV [1]. This paper proposes a end-to-end architecture, DiffTransBEV, a diffusion backed transformer architecture to generate the BEV representation for an autonomous vehicle. This architecture makes use of the powerful DiT [2] architecture as a backbone for BEV generation. This work relates to it's predecessor DiffBEV as it replaces the U-Net architecture with the Diffusion based Transformer architecture that operates on latent patches.**

I. Introduction

Bird's Eye View (BEV) is a top-down projection of the scene with all the agents within them (depicted in Figure. 1). BEV projection plays a key role in multiple domains as they provide a compact and accurate representation of the surrounding environment and agents to provide a 360° and top-down view. It provides a more enhanced perception for better understanding of the surroundings and improved object detection capability. BEV plays a key role in the motion planning module of Autonomous Vehicles.

One of the most recent work done by Nvidia-labs in the form of Lift, Splat Shoot (LSS) [3] which uses multi-view pooling layers to lift the perspective images into voxel level point clouds and then splats the point clouds onto a 2D grid giving the BEV representation of the scene. However, the perception results of LSS are often distorted since the feature distribution in BEV is usually sparse and there is noise associated with the sensors. One of the other primary challenge with the LSS is that it has poor background-foreground discriminative ability.

One of the most important aspect of the BEV representation is that it should be of the highest quality and should not have any noise within them. Addressing the above challenges, Zou et. al. introduced a diffusion based end-to-end architecture to generate the BEV representation in the form of DiffBEV[1]. DiffBEV uses a image backbone to extract the features for the LSS[3] to convert them into a BEV features. These generated BEV features are then refined by the U-Net backed conditional DPM[5] model. Finally a cross attention module fuses the denoised BEV features with the original BEV features to generate the final BEV representation. Although DiffBEV addresses most of the challenges of LSS, it still does not perform well in complex road scenarios and under bad visibility scenarios.

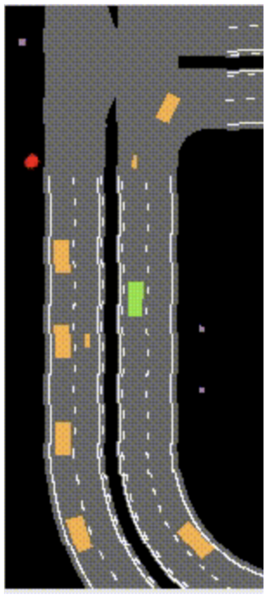


Fig. 1: A sample BEV representation from `carla-birdeye-view` package [4]

DiffTransBEV is an end-to-end architecture similar to its predecessor DiffBEV, which uses a Diffusion based Transformer architecture inplace of the UNet-backed DPM to improve the generative ability of the diffusion models along with improving the inference speed of the BEV representations.

II. Related Work

This work of DiffTransBEV builds upon the recent work of DiffBEV[1]. DiffBEV is an end-to-end architecture which generates a BEV representation from multiple sensor inputs.

A. *Monocular Object Detection*

Monocular object detection depends on how the transformations are modelled from an image plane to a 3-dimensional reference frame. The standard technique is use a 2D object detector in the image plane and use a second network to convert the 2D bounding boxes into 3D

[1]Hari Veeramallu is a Data Science graduate student with the department of Computer Science and Engineering, University of Minnesota. veera047@umn.edu

boxes[6][7][8][9]. Another novel approach which obtained recent success to use an architecture which separately trains two networks, one for depth estimation and the second for object detection[10][11]. These approaches, also known as "pseudolidars", gained immense success due to their cost-effectiveness and versatility.

A third category of monocular object detectors use 3D object primitives that extract features based on their projection onto all available camera sensors[12]. OFT[13] builds on top of Mono3D[12] by projecting a fixed cube of voxels onto images and training a second CNN network to detect on the 3D voxel features. Based on these works, Philion et. al. introduced LSS[3] which works on the principle that a pixel contributes the same feature to every voxel independent of the depth of the object at that pixel. DiffTransBEV exploits the advantages of LSS for object detection.

B. Bird's Eye View representation

The very first effort of generating a BEV was proposed as IPM (Inverse Perspective Mapping)[14], converting the perspective view (PV) into BEV through geometric transformations (Homography) by knowing the intrinsic and extrinsic parameters of the camera. This had quite a few limitations due to it's inability to detect occlusions and creating distortions as it relies on the constraint that the earth is flat. Even GANs[15] are being used nowadays to generate information about occluded objects and places and also to minimize the distortion in areas above the ground plane. Although IPM is simple and efficient, it addressed the BEV transformation through the assumption of rigid and flat-ground. This limits its application in real life.

To overcome the challenges of IPM; MLP based, depth based and transformer based models have been introduced. Some of the recent ones include PETR[16] projects the multi-view images into 3D positional embeddings, BEVDet[17] uses data augmentation in image and BEV view, LSS[3] makes use of camera parameters for BEV and BEVDepth[18] uses the explicit depth supervision of multi-view images. Out of all these, transformer based models[19] have proven to be more and more popular due to their performance and their data driven approach. We base our approach on DiffBEV[1] which exploits the ability of conditional diffusion models for BEV representations.

C. Conditional Diffusion Models

Conditional Diffusion models[20] [5] are a type of generative models which extend the concept of diffusion models[21] by incorporating conditional input conditions. They have emerged as a powerful class of generative models that generate high quality and precise context-aware outputs across wide-range of applications, including image generation[22] [23], detection[24], segmentation[25][26], super resolution[27], text-to-image[28][29] and video generation[30][31].

III. Methodology

A. Framework Overview

The overall architecture of DiffTransBEV is shown in Figure. 2. It comprises of a feature extractor, view transformer, conditional diffusion model, scalable diffusion models with transformers, a cross-attention module and a decoder head. The feature extractor, Swin Transformer[32], extracts the features from each image independently. The extracted features are then lifted from the perspective view to BEV features by the view transformer, LSS[3]. The DiT[2] backed conditional diffusion models refines the noise from the features and generates high quality BEV features. These high quality features are fused with the original BEV features using a cross-attention module, and finally a decode head is used to generate the final BEV representation of the scene.

B. Feature Extractor: Swin Transformer

Shifted **Win**dow transformer (SWin)[32] uses an hierarchial structure with shifted windows to efficiently model the images. It processes images similar to CNNs but with self-attention mechanism and is computationally simple and efficient as compared to transformer architectures[33]. The self attention layer is applied within the local windows instead of the entire image and then the windows are shifted in subsequent layers enabling cross-window connections and reducing computational complexity.

C. View Transformer: LSS

Lift-Splat-Shoot (LSS)[3] is an end-to-end architecture that directly extracts the Bird's-Eye-View representation of a scene given n arbitrary number of cameras. The "lift" module of LSS works on each image in isolation and lifts them from 2D to 3D frame of reference shared across all the cameras. Since LSS works on monocular sensor setup, the lift step predicts the depth at all possible depths inorder to transform into reference frame.

Let $X \in \mathbf{R}^{3XHXW}$ be an image and let p be a pixel within the image with coordinates (h, w). We associate $|D|$ points $\{(h, w, d) \in R^3 | d \in D\}$ to each pixel where D is a set of discrete depths for instances defined by $\{d_0 + \Delta, d_0 + 2\Delta ..., d_0 + |D|\Delta\}$. At each pixel p, the network predicts a context $c \in \mathbf{R}^C$ and a distribution over depth $\alpha \in \Delta^{|D|-1}$ for each and every pixel. The feature vector c_d associated to a point p_d is defined as the context vector for pixel p scaled by α_d. This step is visualized in Figure. 3.

$$c_d = \alpha_d c$$

The large point cloud generated by the "lift" step is converted into PointPillars[34]. PointPillars is a method which utilizes PointNets[35] to learn a representation of point cloud organized in vertical columns known as "Pillars". "Pillars" are voxels of infinite height. Every point is assigned to it's nearest pillar and sum pooling is performed to a tensor that can be processed by convolutional network for BEV inference.

We do not use the "Shoot" module of LSS as it is used for Motion Planning which is out of scope for this paper.

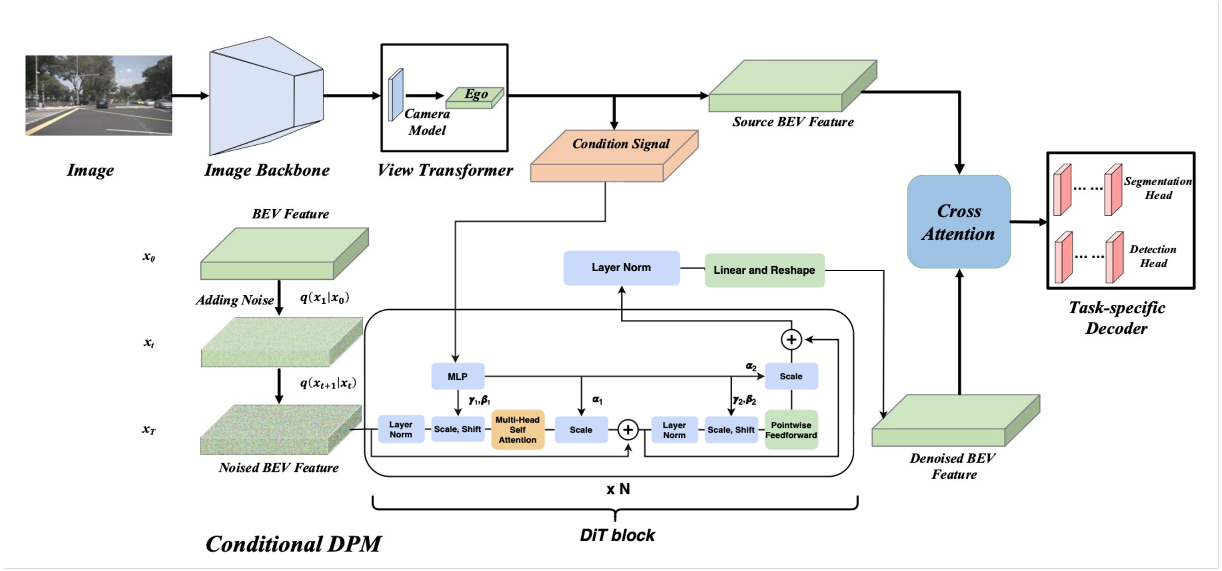


Fig. 2: The proposed model architecture: a) The features are extracted by the Image Backbone (Swin Transformer[32]) b) These features are used by the View Transformer (Lift-Splat-Shoot[3]) to transform the perspective view to BEV c) Gaussian Diffusion model[5] generates the Noised BEV features from the BEV features d) The DiT[2] backbone uses the noisy BEV features and denoises them to obtain denoised BEV features e) The cross attention layer then fuses both the BEV features and denoised BEV features, which are then used to generate the Bird's Eye View representation.

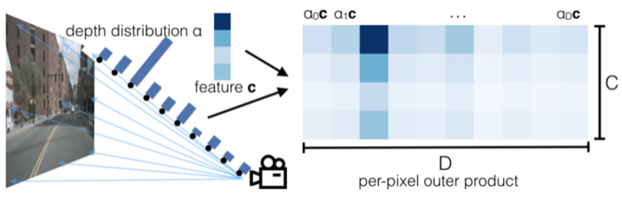

depth distribution α
feature c
$\alpha_0 \mathbf{c}$ $\alpha_1 \mathbf{c}$ ... $\alpha_D \mathbf{c}$
C
D
per-pixel outer product

Fig. 3: Visualization of the "lift" step of the LSS model[3]. For each pixel, a categorical distribution over depth $\alpha \in \Delta^{D-1}$ and a context vector $c \in \mathbf{R}^C$ (left). Features at each point along the ray are determined by the outer product of α and c (right).

D. Conditional Diffusion Probabilistic Models

We formulate the conditional diffusion probability model in this part. The feature generated by the view transformer is treated as the condition for the diffusion model. Diffusion model transforms the noise $x_T \in \mathcal{N}(0, I)$ to the original sample x_0 in a progressive way. The forward pass of the conditional diffusion model can be represented as

$$q(x_t|x_{t-1}) \sim \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

where β_t is the variance at step $t(0 \leq t \leq T)$ and $q(X_{1:T})$ is called the forward/diffusion process.

Similarly the reverse diffusion process can be written as

$$p_\theta(x_{t-1}|x_t) \sim \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

where $\Sigma_\theta(x_t, t)$ is the covariance predictor, μ_θ is the mean function which the diffusion model aims to learn and $p_\theta(X_{0:T})$ is called the reverse process.

E. Diffusion Transformers

Diffusion **T**ransformers (DiT)[2] is a diffusion backed transformer architecture which is similar to the Vision Transformer (ViT) [36]architecture. Instead of the typical U-Net backbone used in diffusion models, DiTs operate on latent patches of the image, utilizing transformers to denoise and refine them progressively. DiTs are computationally efficient as their complexity grows linearly as compared to the quadratic growth of U-Net based architectures.

F. Cross Attention Module

After obtaining the output from the DiT module, we make use of a Cross Attention module to refine the original BEV features.

IV. Implementation

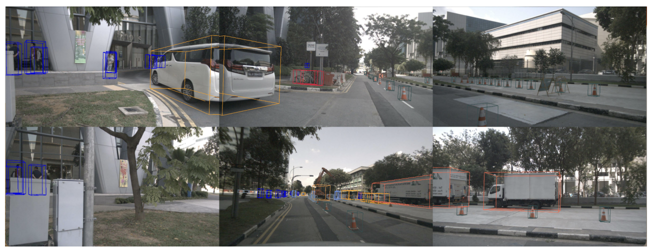


Fig. 4: An instance from a sample scene captured within the NuScenes dataset. Top Row: Front Left Cam, Front Cam, Front Right Cam, Bottom Row: Back Left, Back and Back Right Cam [37].

A. Dataset

The DiffTransBEV model architecture is evaluated on the NuScenes dataset[37]. This dataset was released in March 2019, consisting of 1000 driving scenes in Boston, MA and Singapore. The scenes are annotated with 23 objects 3D bounding boxes at 2 Hz (complete 360° view). The sensor setup consists of 6 RGB cameras, 1 LIDAR, 5 RADARs and 1 GPU/IMU unit. Given our goal to make this approach user affordable and freindly, we only make use of the 6 RGB cameras. A sample scene with it's annotations can be seen in Figure 4.

B. Training

The model is trained on MSI Agate A100 and Mesabi V100 clusters. The models were trained at 225 X 400 image resolution on the NuScenes RGB camera data. Pre-trained ImageNet1K V1[38] weights were used for Swin transformer architecture. The model is trained with AdamW optimizer[39].

A learning rate of 1e-4 with a cosine annealing schedule to decrease the learning rate. A weight decay of 1e-2 and batch size of 4 were used. Small batch size is used since the number of images for one instance is 6, since we have 6 RGB cameras facing in each direction, resulting in a total of 24 images processing together. Anything above batch 4 cannot be computationally handled by the available hardware. The images are normalized using the mean and std of the ImageNet and no other augmentations were performed. 1000 timesteps were used for the diffusion process and was trained for 20 epochs. Due to the huge number of images in the train dataset, to expedite the training process, data prefetchers were used. Warmup epochs were set to 10% of the total epochs, 2 in our case. A base momentum of 0.3 was used to smooth the gradient update process by only considering 70% of the new gradient.

The noise schedule of diffusion model has a starting β_0 value of 1e-4 and an ending β_T value of 1e-2 with 1000 timesteps of diffusion process. Since the training was being performed across multiple GPUs, the batch normalization were synchronized across all the machines. The model training was allowed to run on PyTorch AMP's full precision training in contrast to mixed precision training.

The logs were captured using Weights & Biases (wandb) and were captured for every 10 batches to keep the logs clean.

C. CARLA Simulations

CARLA[40] simulator was used to simulate the real-world traffic and capture surrounding data for the model to be evaluated on. 6 RGB cameras were configured in a setup similar to the setup of NuScenes data setup as seen in Figure.5. One additional spectator camera was setup in a game view to observe the motion of the car in the simulated environment as shown in Figure. 6.

CARLA works as a server-client based application and can be controlled through a python sdk. Using python sdk, we can configure the scenarios, maps, vehicles, NPC (Non-Player Characters) and the weather. For the purpose of simulations for this paper, a Tesla Model 3 vehicle was used. Two straight facing cameras, one forward and the other backward were placed on top of the vehicle along the axis. Two cameras were placed on each side of the vehicle, capturing the front-left, front-right, back-left and back-right angles of the vehicles to ensure maximum visibility of the surroundings (seen in Figure.5). The cameras were all synchronized using a pygame clock and were captured at a refresh rate of 20*Hz*. 40 NPC vehicles and 20 NPC pedestrians were randomly placed in the scene and were set in an autopilot mode along with the main vehicle.

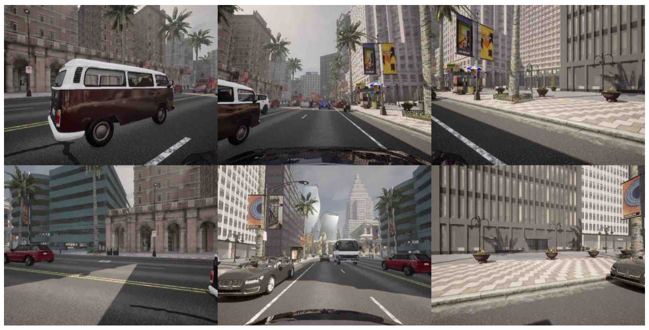


Fig. 5: 6 camera setup in CARLA simulator capturing the scene. Top Row: Front Left Cam, Front Cam, Front Right Cam, Bottom Row: Back Left, Back and Back Right Cam

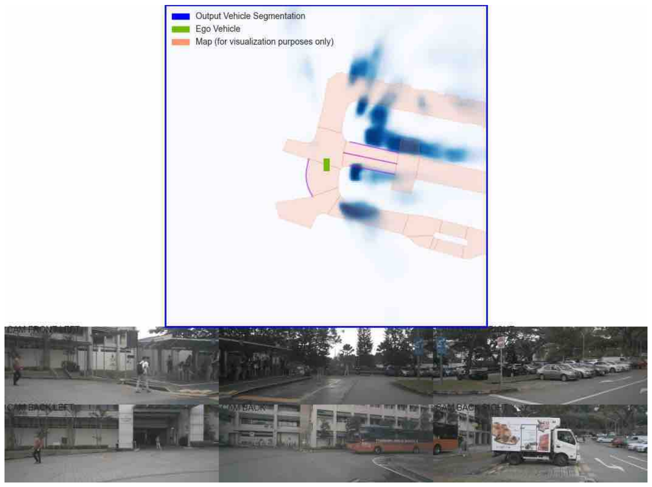
Fig. 6: Spectator view of the car in CARLA.

V. Results and Discussions

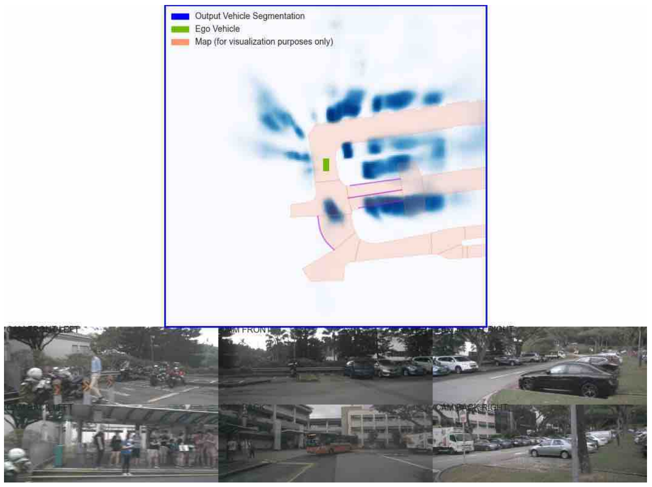
The model was only being able to train for 20 epochs on the entire dataset due to hardware limitations and allocation of resources. During the training the loss decreased steadily without without any indication of spikes or sudden increase in losses. The BCE loss was used as a loss function to verify if there was a vehicle at a certain pixel location or not.

A. Visualizing the results

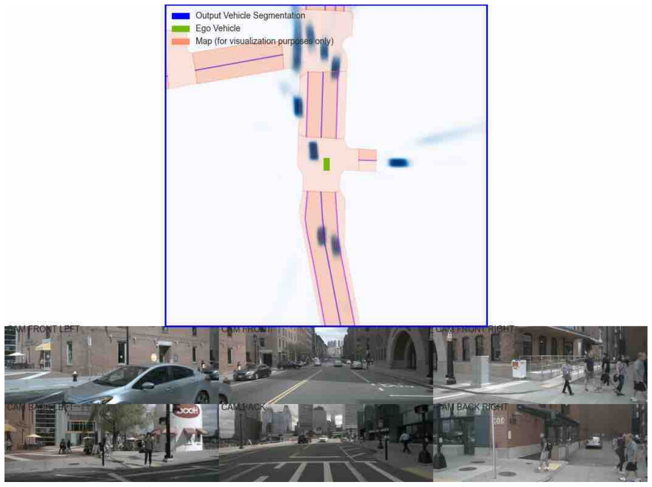
The DiffTransBEV model was evaluated on a few scenes to visualize the performance of the model. Some of the instances of these predicted visualizations are shown in Figure. 7.


Output Vehicle Segmentation
Ego Vehicle
Map (for visualization purposes only)

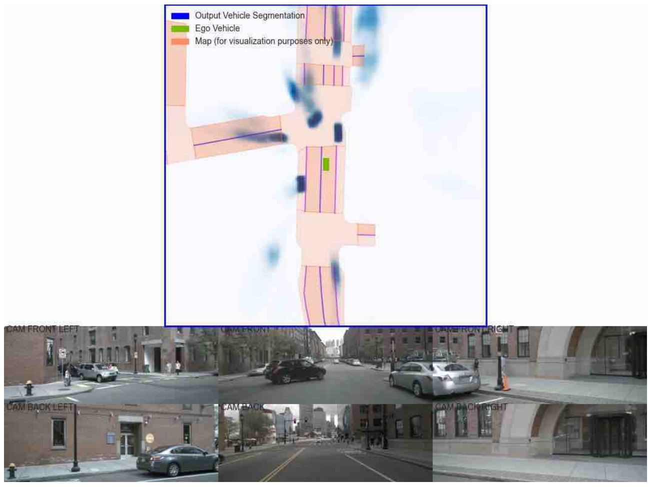
(a) Scene 1 Instance 1


Output Vehicle Segmentation
Ego Vehicle
Map (for visualization purposes only)

(b) Scene 1 Instance 2



(c) Scene 2 Instance 1


Output Vehicle Segmentation
Ego Vehicle
Map (for visualization purposes only)
CAM FRONT LEFT
CAM BACK LEFT
CAM BACK
CAM BACK RIGHT

(d) Scene 2 Instance 2

Fig. 7: BEV Visualizations generated on the test data along with the actual input camera views.

From Figure 7a. we can see that the model does not perform well in identifying the incoming bus behind the vehicle when the bus takes a turn. Similarly, it can also be observed that the model fails to identify pedestrians, but gives a faint hint of the precense of two-wheeler motocycles in the front which can be observed from the front camera. Even though the model could not detect the buses, pedestrians and motocycles accurately, it did predict the four-wheeler vehicles almost correctly.

Similarly, looking at Figure. 7b. of the same scene after a while, we see that the model is able to predict the pedestrians when there are a large number of pedestrians together. Here the model is able to predict the bus and the surrounding cars as well. Although it did perform well for this frame, it gave a very faint prediction for the single motorcycle parking in the front.

Looking at a different scene in Figure 7c., we see the autonomous vehicle driving in a straight road with more moving vehicles around it. We see that the model is able to predict the moving vehicles accuractely but completely ignores the pedestrians walking on the sidewalks. We can observe that the model even identifies the car parked in the alley as visible in the back right camera, even though the pedestrians are closer than the car itself.

Looking at the same scene as before in Figure 7d., the autonomous vehicles approaches an intersection where a car merges from the left. We can see that the model has the ability to retain information as it is able to predict that there is car on the opposite side which is obscured by the merging car. Looking at the next corresponding frames (not displayed in the report) we can observe that there is indeed a car obscured by the merging car.

Similarly, we deployed our model on CARLA simulations to verify the performance in sim. The results were not as good as the evaluation on real data as seen in Figure 8. This sudden drop of performance on simulated environment might be due to the fact that the training was entirely done on real data and evaluation was performed on completely new environment.

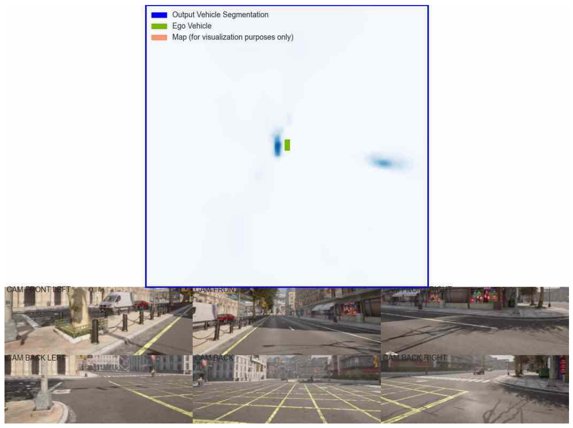

Output Vehicle Segmentation
Ego Vehicle
Map (for visualization purposes only)

Fig. 8: Evaluation on simulated environment.

VI. Conclusions

This work proposes a novel architecture, DiffTransBEV, which uses the power of conditional diffusion models and diffusion transformers to generate accurate and more reliable BEV representations. DiffTransBEV uses the advantage of shifted windows of Swin Transformer to process the images at multiple scales to extract sensitive and important details. Then it uses the transformation power of the well versed LSS architecture to generate voxel level BEV features which are then further refined using the diffusion process. The cross attention module at the end of the architecture has the capability to attentively learn the relationship between the original BEV features and the refined features generated by the diffusion process. Qualitatively DiffTransBEV achieves good performance even though it has been trained on a very few number of epochs, indicating that it could be further improved by performing extensive training and improvements.

VII. Future Work and Extensions

Autonomous vehicle industry is a booming field in the current age and this work on generating BEV representations has great practical application. DiffTransBEV could not achieve its goals for this paper due to the numerous challenges that we faced during the model development, training and simulation. DiffTransBEV is a successor to DiffBEV[1] which does not have a publicly available codebase, making it an extra effort to first develop the DiffBEV architecture and then build on top of it. Apart from developing the model architecture from scratch, we faced multiple memory leak issues due to the distributed architecture that was being used. We had to try different methods before ultimately reaching the suitable one. The major challenge for this work was in the form the massive dataset of NuScenes which consisted of around 28,130 train samples with 6 camera views each. The complex model architecture combined with the huge dataset became a challenge to train on the MSI shared resources.

The challenges faced within the simulator were due to the synchronization issue of the RGB camera outputs from CARLA. We tried various multi-threading methods but the frames always were inconsistent. We finally tried synchronizing using the pygame clock and achieved synchronization of the camera outputs. After synchronizing the camera frames, we evaluated the model on simulated data and found the results to be bad.

DiffTransBEV was theorized with an intention to improve the accuracy and speed of the BEV generation process but did not achieve the computational efficiency it aimed. One of the potential reason for this was due to the computational complexity of Swin Transformer[32] and Lift-Splat-Shoot[3] architecture. Potential future works could include looking at faster view transformer and feature extractor architectures such as FastBEV[41] and MobileNetV2[42]. Domain adaptation can be performed in the future to improve the performance of the model on simulated data. Simulations can be further extended to test the model on adverse climatic conditions to test the denoising power of the diffusion models.

References

[1] J. Zou, Z. Zhu, Y. Ye, and X. Wang, "Diffbev: Conditional diffusion model for bird's eye view perception," *arXiv preprint arXiv:2303.08333*, 2023.

[2] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4195–4205.

[3] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*. Springer, 2020, pp. 194–210.

[4] M. Martyniak, "Bird-eye's view for carla," https://github.com/deepsense-ai/carla-birdeye-view/tree/master, 2023, accessed: 2023-11-17.

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[6] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1521–1529.

[7] Z. Qin, J. Wang, and Y. Lu, "Monogrnet: A geometric reasoning network for monocular 3d object localization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 8851–8858.

[8] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder, "Disentangling monocular 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1991–1999.

[9] K. Mani, S. Daga, S. Garg, S. S. Narasimhan, M. Krishna, and K. M. Jatavallabhula, "Monolayout: Amodal scene layout from a single image," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1689–1697.

[10] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8445–8453.

[11] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving," *arXiv preprint arXiv:1906.06310*, 2019.

[12] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2147–2156.

[13] T. Roddick, A. Kendall, and R. Cipolla, "Orthographic feature transform for monocular 3d object detection," *arXiv preprint arXiv:1811.08188*, 2018.

[14] M. Bertozz, A. Broggi, and A. Fascioli, "Stereo inverse perspective mapping: theory and applications," *Image and vision computing*, vol. 16, no. 8, pp. 585–590, 1998.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[16] Y. Liu, T. Wang, X. Zhang, and J. Sun, "Petr: Position embedding transformation for multi-view 3d object detection," in *European Conference on Computer Vision*. Springer, 2022, pp. 531–548.

[17] J. Huang, G. Huang, Z. Zhu, Y. Ye, and D. Du, "Bevdet: High-performance multi-camera 3d object detection in bird-eye-view," *arXiv preprint arXiv:2112.11790*, 2021.

[18] Y. Li, Z. Ge, G. Yu, J. Yang, Z. Wang, Y. Shi, J. Sun, and Z. Li, "Bevdepth: Acquisition of reliable depth for multi-view 3d object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 2, 2023, pp. 1477–1485.

[19] T. Roddick and R. Cipolla, "Predicting semantic map representations from images using pyramid occupancy networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 138–11 147.

[20] G. Batzolis, J. Stanczuk, C.-B. Schönlieb, and C. Etmann, "Conditional image generation with score-based diffusion models," *arXiv preprint arXiv:2111.13606*, 2021.

[21] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[22] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[23] A. Graikos, N. Malkin, N. Jojic, and D. Samaras, "Diffusion models as plug-and-play priors," *Advances in Neural Information Processing Systems*, vol. 35, pp. 14 715–14 728, 2022.

[24] S. Chen, P. Sun, Y. Song, and P. Luo, "Diffusiondet: Diffusion model for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 830–19 843.

[25] T. Amit, T. Shaharbany, E. Nachmani, and L. Wolf, "Segdiff: Image segmentation with diffusion probabilistic models," *arXiv preprint arXiv:2112.00390*, 2021.

[26] D. Baranchuk, I. Rubachev, A. Voynov, V. Khrulkov, and A. Babenko, "Label-efficient semantic segmentation with diffusion models," *arXiv preprint arXiv:2112.03126*, 2021.

[27] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, "Image super-resolution via iterative refinement," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 4713–4726, 2022.

[28] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[29] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36 479–36 494, 2022.

[30] R. Yang, P. Srivastava, and S. Mandt, "Diffusion probabilistic modeling for video generation," *Entropy*, vol. 25, no. 10, p. 1469, 2023.

[31] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, *et al.*, "Imagen video: High definition video generation with diffusion models," *arXiv preprint arXiv:2210.02303*, 2022.

[32] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[34] A. H. Lang, A. H. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds in: Ieee conference on computer vision and pattern recognition (cvpr), 2019," 2018.

[35] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 918–927.

[36] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[37] W. K. Fong, R. Mohan, J. V. Hurtado, L. Zhou, H. Caesar, O. Beijbom, and A. Valada, "Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3795–3802, 2022.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[41] Y. Li, B. Huang, Z. Chen, Y. Cui, F. Liang, M. Shen, F. Liu, E. Xie, L. Sheng, W. Ouyang, *et al.*, "Fast-bev: A fast and strong bird's-eye view perception baseline," *arXiv preprint arXiv:2301.12511*, 2023.

[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.