Depth Estimation using DNN Architecture and Vision-Based Transformers

Uday Kulkarni[1*], *Shreya B* Devagiri[1], *Rohit B* Devaranavadagi[1], *Sneha* Pamali[1], *Nishanth R* Negalli[1] and *Prabakaran* V[2]

[1]Computer Science, KLE Technological University, Vidyanagar, Hubballi, 580031, Karnataka, India
[2]Bommanahalli, Bengaluru, 580076, Karnataka, India

Abstract. Depth Estimation is the process of estimating the depth of objects with a 2D image as an input. The importance of Depth Estimation is that it provides some critical information about the 3D structure of a scene given a sequence of 2D images. This is helpful in various applications like robotics, virtual reality, autonomous driving, medical imaging, and so on and so forth. Our main objective here is to make the environment more perceivable by autonomous vehicles. Unlike the traditional approaches which try to extract depth from the input image, we apply instance segmentation to the image and then use the segmented image as an input to the depth map generator. In this paper, we use a fully connected neural network with dense prediction transformers. As mentioned above, the image after instance segmentation is given as an input for the transformers. The Transformer is the backbone for producing high-resolution images. Our experimentation results have shown many improvements related to the details and sharpness of the image as compared to the traditional depth estimation techniques. The architecture is applied and tested on the KITTI data set.

1 Introduction

As one of the most useful intermediate representations for any action in physical contexts, depth prediction or depth estimation is a traditional task in many applications including computer vision, robotics, autonomous driving, and so forth [1]. The depth of objects in the picture can be determined using a variety of techniques, such as stereo vision, which compares two pictures of the same scene taken at slightly different perspectives or angles. Other methods include utilizing specialized depth sensors like Light Detection and Ranging (Li-DAR) [2] or Machine Learning algorithms that were trained on huge data sets of labelled depth images. By considering the visual features from the image to determine the relative distances of elements in the scene, one can estimate the depth of the image. The camera must be calibrated and the algorithms used for depth estimation must understand the geometry of the scene and any known information about the objects in the image in order to do this task efficiently.

*Corresponding author: nishanth13.rajesh@gmail.com

In computer vision, depth estimation refers to the process of estimating the distance of objects in an image from the camera or the viewer. This information is used to improve the accuracy and effectiveness of various computer vision tasks, such as object recognition, scene understanding, and 3D reconstruction [3]. Stereoscopic vision [4], structure from motion, multi-view stereo, depth from focus, depth from defocus, depth from motion, depth from structure, and machine learning are some of the techniques that can be used for depth estimation in computer vision.

Depth estimation in machine learning refers to the process of using algorithms to estimate the distance of objects in an image or video from the camera or viewer. It can be achieved by training a machine learning model on a large data set of images or video sequences with corresponding depth information. Supervised learning [5], Unsupervised learning\cite [6], and Semi-supervised [7] learning are the several approaches used for depth estimation. There are several Machine Learning algorithms that can be used for depth estimation, including Convolution Neural Networks (CNN) [8], Recurrent Neural Networks (RNN) [9], and Auto Encoders [10]. These algorithms are trained on a variety of depth estimation tasks, including single-image depth estimation, multi-view depth estimation, and video depth estimation.

Image depth estimation is used for diverse applications. In Robotics, depth estimation is used to enable the root to perceive and understand its environment [11], which is necessary for tasks such as navigation and object manipulation. Depth Estimation is an important step in many computer vision tasks, such as object recognition, scene understanding, and 3D reconstruction. It is used in photography [12] to improve the quality of photographs by adjusting the focus and depth of field to create more visually appealing images. It is also critical for creating realistic and immersive virtual reality experiences, as it allows the system to accurately render objects in 3D space and respond to user movement. Depth estimation is also used in medical imaging [13] to improve the accuracy and effectiveness of medical imaging techniques, such as CT and MRI scans, by providing a more detailed understanding of the spatial relationships between objects in the body. It is used in geospatial mapping to create accurate 3D models of the world for applications such as mapping and surveying. Depth estimation is also used in augmented reality [14] to create more realistic and immersive augmented reality experiences by accurately placing virtual objects in 3D space relative to real-world objects.

In this paper, we are following an image preprocessing approach. We use a Fully Convolutional Network (FCN) network with dense prediction transformers [15] an architecture that is the backbone for the depth map generation. Here, we are predicting the depth by identifying the object (done by Object Detection) and then we apply the Instance Segmentation on the object identified once the object is instantly segmented [16]. This is the input for the Transformers i.e., attention-based modules that we use instead of convolution networks. The transformer is the backbone for high resolution of images. The tokens are generated at various fields for image representations for multiple resolutions and then combine to form full-resolution predictions. Our experiment results show that this architecture output has a lot of improvements compared to the other depth predictions. This architecture has been implied on the KITTI data set.

This paper is organized into five sections. Section 1 consists of the Introduction to our study field. Section 2 analyzes the Related Work techniques that have already occurred to produce depth maps. The Methodology proposed is discussed in section 3 whereas Section 4 illustrates the Results and Analysis. Section 5 provides the Conclusions.

2 Related Work

Here, we provide an overview of the most recent Supervised, Self-supervised, Unsupervised, and Transformers depth estimation techniques that were developed and used to process the input image in order to obtain the end results.

2.1 Supervised learning of depth estimation

Supervised learning is a technique where the proper output is already known for each input when a model is being trained using labeled data. The model can estimate the results based on the input data by determining the correlation or mapping between the inputs and their corresponding outputs then using fresh, unforeseen data the model can be utilized to make predictions. The most crucial machine learning methodology is supervised learning, which is equally crucial for handling multimedia data. The purpose of supervised learning in classification problems is frequently to teach the computer a categorization scheme that we have developed [17].The categorization problem is a typical formulation of the supervised learning task: By studying various input-output instances of the function, the learner must learn or approximate the behavior of a function that translates a vector into one of several classes [18].

Basically, the learning process is done in two steps, training, and testing. In training, the data set is trained with labels. In the testing phase, when we give the input, the model should be able to predict the respective label. The primary aim is to create an estimator that can predict an object's label using a set of features [5].

2.2 Self-Supervised learning of depth estimation

Self-supervised Monocular Depth Estimation (MDE) was recently presented to optimize a network based on the photo-metric error between the projected picture and the real image, representing depth as the geometric attribute of an image projection transformation between stereo image pairs [7]. There are two self-supervised approaches for training the depth models namely stereo pairs and monocular videos. Stereo training predicts the disparity pixel between the pairs of pictures using synchronized stereo pairs of images [19]. When training a network for monocular video, a series of frames performs both the training signal and the camera posture prediction that is only necessary during training to reduce the constraints on depth prediction [20].

Performing tasks with stereo pairs and monocular video introduces challenges thus to overcome, Godard Clement [21] proposed a model appearance matching loss to deal with the obstructed pixels issue that arises while adopting monocular supervision and auto-masking method to discard pixels in monocular training where no relative camera motion is seen also an appearance matching loss that conducts all image sampling at the input resolution in order to minimize depth issues. Peng Rui [22] introduced an approach of data augmentation called data grafting, this method is used for enhancing data that challenges the model to look beyond the vertical image orientation for indications of depth and proposed selective post-processing to obtain the self-distillation label and use it to construct a new loss for the model, called the Self-Distillation Loss and also the full-scale network, created to provide the encoder a focus on the depth estimation problem and boost the model representational capability.

2.3 Unsupervised learning of depth estimation

Unsupervised learning techniques do not require labels, unlike supervised learning techniques, hence they solve the drawback of supervised learning's reliance on labels. The unsupervised depth and camera ego-motion estimation just need raw video sequences in the unsupervised learning approach, these techniques improve the model using video data collected from a new scene [23] enabling quick deployment in real-world scenarios. Despite the lack of ground truth depth data, Godard Clement [24] offered a unique training objective that enables CNN architecture to learn to carry out a single picture depth estimate. They also created disparity pictures by training the network with an image reconstruction loss and making use of epipolar geometry restrictions.

They demonstrated that resolving for picture reconstruction alone yields depth images of poor quality. In order to solve this issue, unique training loss was introduced that ensures consistency between the disparities generated relative to both the left and right pictures, resulting in better performance and stability compared to current methods. Zhou T [6] proposed an end-to-end learning pipeline that supervises single-view depth and camera position estimates using the objective of view synthesis. Despite having been trained on unlabeled video, the model performs on an equal level with methods that depend on ground-truth depth. In order to represent the complete vision synthesis pipeline as a CNN's inference process, it is necessary to train the network on a significant number of video data for the "meta-task" of view synthesis. This forces the network to learn about intermediate tasks such as depth estimation and camera posture estimation, in order to develop a theoretical base of the visual world.

2.4 Transformers

Transformers have been evidenced to be effective in a few machine learning tasks such as image classification, object detection, and semantic segmentation. In recent years, there has been significant growth in using transformers for image depth estimation, which involves predicting an image's depth map [1].

A transformer-based depth estimation technique with a two-stage mechanism. A CNN model retrieves features from the input image during the initial step, and a transformer obtains the depth map in the second stage. The transformer-based architecture features a pyramid structure to enhance the effectiveness of dense prediction tasks such as depth estimation [25]. An end-to-end transformer-based depth estimation method here utilizes a self-attention framework to grasp long-range dependencies with the given image. According to studies that compared CNN designs namely ResNet and Vision Transformer, transformers seem to be more resistant to corruption as well as difficult instances in classification [26]. An author proposed a hybrid network for monocular depth estimation, something which consists of a Transformer-base utilization and even a CNN-based decoder.

2.5 Object Detection

Finding every single instance of a real-world thing, such as a human face, a flower, an automobile, etc., in pictures or videos in real-time and at the greatest precision possible is termed object detection. The object identification technique recognizes every instance of an item category using derived features and learning techniques [27].

High inference speeds are attained by one-stage detectors [26], while high localization and recognition accuracy are attained by two-stage detectors [25]. A Region of Interest (RoI) Pooling layer can be used to split up the two stages of a two-stage detector [1]. Without considering the region proposals, a one-stage detector might predict the bounding boxes in a single-step process [27]. To localize the region of identification in the picture and restrict the form of the object, it makes use of guidance and anchors [26].

2.5.1 Two-Stage Detector

RoI Max - pooling can be applied to split up the two stages of a two-stage detector. Faster R-CNN is indeed one of the well-known two-stage object detectors. The RPN, which is also known as the Region Proposal Network, is the initial stage, and it generates potential bounding boxes [25]. For the following classification and bounding-box regression tasks, features are pooled by RoI from each candidate box in the next stage.

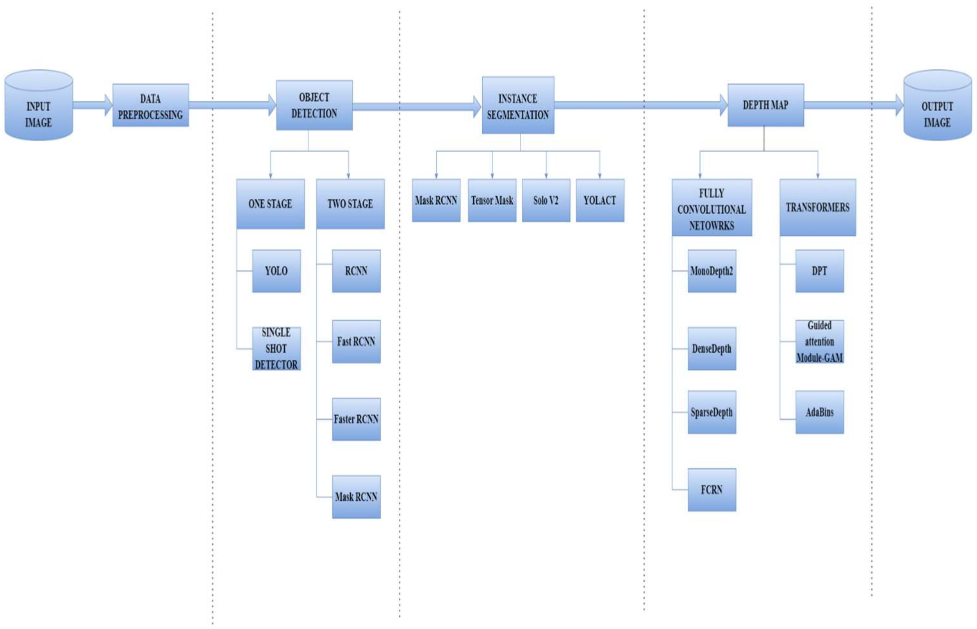

Fig. 1. The pipeline illustrates the different phases the input picture goes through to produce a depth map, including object detection as the first step, instance segmentation after that, and depth map methods at the end.

3 Proposed Methodology

This section describes how we designed an image depth prediction network using DNN Architecture transformers. We first identify the object in the input image and then apply instance segmentation to the object before passing it to the transformers in order to generate a depth map without the assistance of ground truth depth as shown in Fig1.

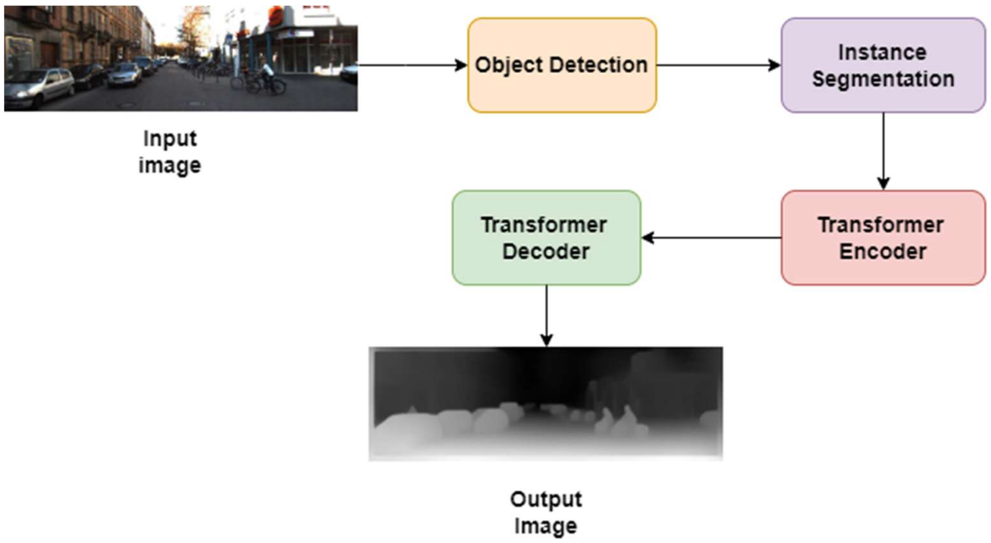

Fig. 2. The above figure gives a description of the proposed methodology. The input image is segmented after object detection. Then, it is passed through the transformer to get the output image which is the required depth map.

3.1 Object Detection and Instance Segmentation

For object detection in our research, we apply a two-stage object detector. The objects are detected in two phases by a two-stage object detector. In the first phase, from the input images features are extracted and an RoI is proposed, which consists of a possible box in which an object might be detected. The features and RoI are used in the second phase; it generates the final bounding box and divides the bounding box into classes, and determines the probabilities associated with each class detected. As it has two separate networks its accuracy is better when compared to single-stage object detection. When compared to single-stage detectors, two-stage object detectors can identify objects at different scales and provide more precise item localization, something that frequently occurs in real-world situations, even when some of the objects in the image are partly covered by other objects. Two-stage object detectors can be used in applications like robotics, surveillance, and autonomous vehicles.

Mask RCNN [28] is a deep classification algorithm for object detection and instance segmentation. It is a two-stage approach that first creates potential object areas in a picture using an RPN, and then uses a different network to categorize and segment the regions. Mask RCNN's [28] ability to produce excellent object masks, which allow it to correctly separate individual objects in an image, gives it a significant edge over other object recognition techniques.

It is an advancement of the Faster R-CNN architecture [29] which introduced the use of an RPN to create possible RoIs in an image. Mask R-CNN [28] extends the Faster R-CNN architecture [29] by adding a third branch that generates a binary mask for each identified object. Mask R-CNN [28] detects objects in two stages. The first stage includes a feature extractor and a region proposal network. The feature extractor is often a CNN that has been pre-trained on a large data set, such as ImageNet [30], to extract important features from an input image. The RPN generates a set of RoIs using these features. It contains a detection network and a mask generation network in the second stage. The RoIs are used by the detection network to classify and locate each object in the image. Finally, the mask generation network uses the RoIs to build a binary mask for each observed object. Mask R-CNN uses a binary mask of size M x N, where M and N are the height and width of the identified object's RoI. The mask is represented as a 2D binary matrix with 1's in the object's places and 0's everywhere else. The mask representation is created by superimposing an FCN on top of the RoI feature map generated by the RPN. The FCN takes the RoI feature map as input and outputs a M x N x C tensor, where C is the number of object classes. Each tensor slice corresponds to an anticipated binary mask for a certain class. The final mask for a detected item is generated by selecting the slice matching to the expected class of the object and thresholding it at 0.5 to obtain a binary mask.

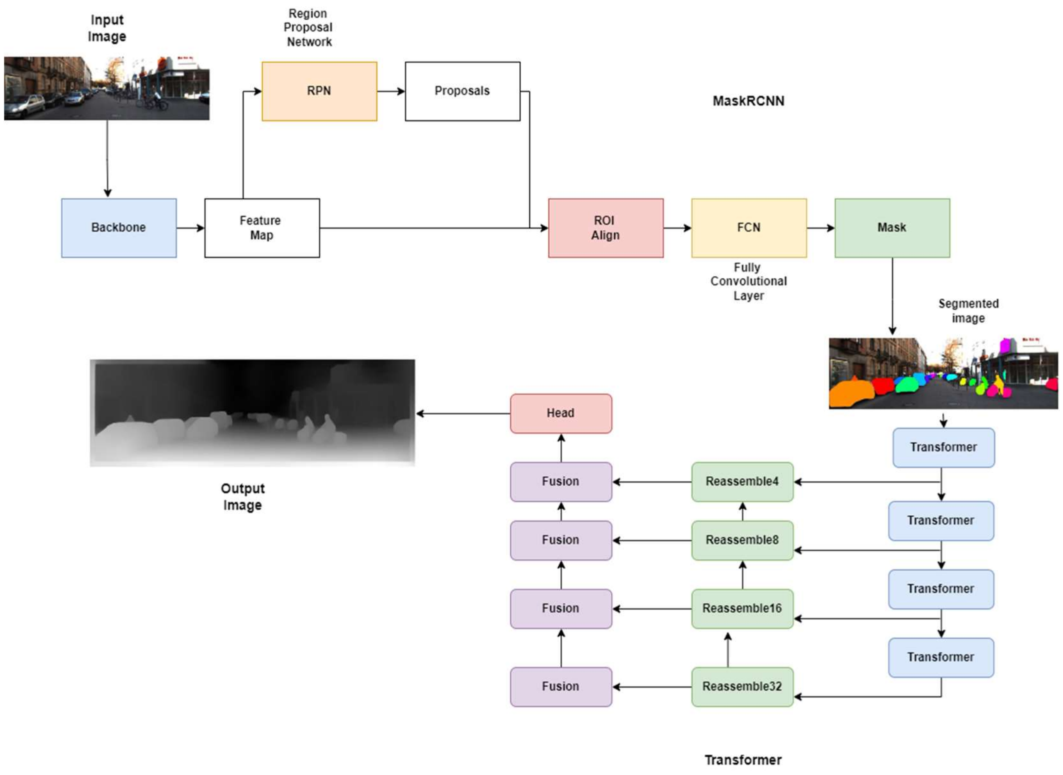

Fig. 3. The Proposed architecture of the model which uses a Mask RCNN and Transformer to produce the required heat map. The backbone of the input image is used to get the feature map which will be passed through the FCN to get the segmented image. The segmented image is then passed through the transformer to get the final heat map.

3.2 Transformers

Depth Estimation is the task of determining the distance of objects in an image from the camera that captured the image. A Vision Transformer (ViT) is used to estimate depth by taking an image in which objects are segmented (masked) i.e. Instance Segmentation is done to the input image and predicting the depth of each pixel in the image. The model can learn to do this by being trained on a huge collection of images with associated depth maps. Transformers are made up of encoders and decoders. A CNN as the encoder and a fully connected network as the decoder is used in an encoder-decoder model for depth map generation. The encoder uses the CNN to extract the features of the images and these features are called tokens the decoder uses the tokens of the images to generate the depth map.

3.2.1 Encoders

The output of DNN architecture is masked objects in the given image, which will be the transformer encoder's input. To begin, we must divide the images into non-overlapping patches before sending them to a transformer-based neural network. The input image is first divided into a grid of non-overlapping patches, with each patch containing a square region of pixels. The patch size is 16x16 pixels i.e., p=16. To construct a patch embedding, each patch is flattened into a one-dimensional vector and processed through a linear projection layer. Each patch is projected into a higher-dimensional vector space, allowing the network to record more complicated patterns and interactions between patches. A sequence of self-attention layers is applied to the input image by the transformer encoder for images. The self-attention layers also assist the network in comprehending the links between various portions of the image. The encoder uses the CNN and using ResNet-50 as the backbone to extract the features of the images the output of the CNN is the feature vectors of the input images. These feature vectors of images are called tokens and the decoder uses the tokens of the images to generate the depth map. The tokens are featured vectors of the given part of the input images and each token contains the different parts of the images and has a one-to-one correspondence with other tokens and consists of spatial resolutions of the input embedding of the given input image. The encoder assigns the positional encoding to the tokens so that while reassembling the tokens we take care of the positions to generate the depth map.

When this process is applied to a picture of size M X N pixels, the outcome is a collection of $\boldsymbol{T^0} = \left\{\boldsymbol{T^0_0}, \dots, \boldsymbol{T^0_{N_p}}\right\}, \boldsymbol{T^0_n} \in \mathbb{R}^{\boldsymbol{F}}$ tokens, where $\boldsymbol{N_p} = \frac{\boldsymbol{MN}}{\boldsymbol{p^2}}$, T^0 refers to Readout token and F is the feature dimension of each token. The input tokens are changes into into new representations T^i using T transformer layers, where i refers to the output of the i-th transformer layer. We have used T= 12 transform layers, p=16 and attract the features at 1/16 input resolution and F= 1024 respectively.

3.2.2 Decoders

The encoder output i.e., tokens are the input to the decoders which generates the detailed depth map of the given input image. The decoder consists of multiple layers of self-attention and fully connected layers. The self-attention technique enables the network to focus on tokens at different layers, assisting in the capture of global dependencies in the image; it is local in nature and only considers tiny parts of the input image at a time. The use of fully connected layers is to generate the depth map. To generate the depth map first we read the input tokens based on the positional encoding then we concatenate the tokens and then re-sampling of tokens is done to form the reassembled image output i.e., depth map is generated of the given input image. The reassembling is done in three stages. The three stages are Resample, Read, and Concatenate.

(i) Read: We begin by mapping the Np + 1 tokens to a set of Np tokens that can be spatially concatenated into an image-like representation. Read functionality has three phases. The first phase is ignored, the second phase is added and the third phase is projection. In ignore phase we ignore the token and in the second phase, we pass the token information to all the other tokens by adding the representations the third phase is to transmit the information to the other tokens by concatenating the tokens to all other tokens and then projecting the representation to the original feature dimension F using a linear layer and GELU non-linearity.

$$READ: \mathbb{R}^{N_p+1\times F} \to \mathbb{R}^{N_p\times F} \quad (1)$$

$$\mathrm{READ}_{\mathrm{IGNORE}}\,(T) = \left\{t_1, \dots, T_{N_p}\right\} \quad (2)$$

$$\mathrm{READ}_{ADD}(t) = \left\{T_1 + T_0, \dots, T_{N_p} + T_0\right\} \quad (3)$$

$$\mathrm{READ}_{\mathrm{PROJECTION}}\,(t) = \left\{\mathrm{mlp}(\mathrm{cat}(T_1, T_0)), \dots, \mathrm{mlp}\left(\mathrm{cat}\left(T_{N_p}, T_0\right)\right)\right. \quad (4)$$

(ii) Concatenate: By positioning each token in accordance with the location of the original patch in the image, the resulting Np tokens can be molded into an image-like representation. Technically, we perform a spatial concatenation operation that yields a feature map with D channels and a dimension of $\frac{M}{p} X \frac{N}{p}$.

$$\text{Concatenate}: \mathbb{R}^{N_p\times F} \to \mathbb{R}^{\frac{M}{p}\times\frac{N}{p}\times F} \quad (5)$$

(iii) Resample: The Concatenate representation is then given to a spatial resampling layer, which scales it to size $\frac{M}{s} X \frac{W}{s}$ with F features per pixel where s- s output size ratio of the image and $\hat{F}$ denotes the output feature dimension. To carry out this operation, we first project the input representation to F using 1X1 convolutions. Next, we perform a 3X3 convolution when s < p or a stride 3X3 transpose convolution when s < p, to carry out spatial down sampling and up sampling procedures, respectively.

$$\text{Resample}_{\,s}: \mathbb{R}^{\frac{M}{p}\times\frac{N}{p}\times F} \to \mathbb{R}^{\frac{M}{s}\times\frac{N}{s}\times\hat{F}} \quad (6)$$

4 Results and Analysis

In this section, experimental setup, data-set description, information of KITTI and Indian road data-set, implementation, and results are shown respectively.

4.1 Experimental Setup

Object recognition and masking, i.e., semantic segmentation are implemented on Keras 2.0.8, while transformers for depth map generation are implemented on PyTorch 1.12.1. We utilized NVIDIA DGX-1 on GPU 0 Tesla V100-SXM2 and GPU 1 Tesla V100-SXM2 with 3 cores for training and testing. In the training step, we use a part of a pre-trained model from Monodepth. Our Customized Depth Estimation model needed over 56 hours to be trained on the mentioned system. The model size is 627 MB and 110 million parameters.

4.2 Data-set Description

The KITTI data set includes a variety of sensor data from a car-mounted platform traveling through urban and rural parts of Germany, including stereo photos, LiDAR data, and GPS/IMU measurements. For a variety of applications, including object detection, tracking, depth estimation, and semantic segmentation, the KITTI data set contains annotated data. The Indian Road data set consists of some pictures taken from the dashcam of vehicles on the Indian road. The images in this collection were taken while a car was traveling on Indian roadways, including highways and urban areas. It presents a number of difficulties, including traffic, pedestrians, and road signs. For applications like item detection, categorization, and tracking, the data set offers labeled data.

4.2.1 KITTI Data-set

The KITTI [31] data collection contains stereo [32] images that can be used to generate the depth maps. The data set was developed by the Karlsruhe Institute of Technology and the Toyota Technological Institute in Chicago. It includes a variety of data types, such as stereo [32] and LiDAR data, high-resolution colors and gray-scale pictures, and vehicle and object motion data. The collection also includes ground truth data, such as detailed depth maps, 3D object labels, and vehicle odometry [33] information, allowing for quantitative evaluation of various algorithms. The collection contains data taken from a variety of locations, including urban, suburban, and rural places, and it covers a variety of lighting and weather situations. This data set is frequently used as a benchmark data set in autonomous driving research for several computer vision tasks.

4.2.2 Indian Road Data-set

For testing our models to determine if they generate accurate depth maps or not, we created a custom data set for Indian roads. We captured the images in various lighting conditions and at different times of the day. Moreover, a diverse data set is created so that it includes various regions and different types of Indian roadways is important. The image resolution is 3024x3024 jpg format, uncompressed and the aspect ratio is 1:1. The data set was numbered from 1 to 3000.

4.3 Implementation

There are two stages in the training phase. We use the KITTI-RAW [31] in the first stage to recognize the object and instantly segment on the picture, i.e., mask on the object, using the [28] i.e., two-stage object detection method using MS COCO model. There are 80 classes in the MS COCO model. We proceed to the second stage after masking the object in the training data set. The second stage is set up for training and creating a model, after which testing will be performed on the model. The following are the transformer stage's default hyper-parameters: The training epochs are set to 80 where each epoch step consists of 72,000 steps and the batch size is 16. The optimizer used here is Adam with a learning rate of 1e-5. Once the model has been generated, it is tested on two data sets: KITTI and Indian Road.

4.4 Results

We have the results for KITTI and the Indian road data set shown below. The input image is taken randomly from the KITTI or Indian data set and the output is produced as shown.

4.4.1 KITTI Data-set

We can visualize the results of the experiment performed on the KITTI data set in Fig 4. The image collage consists of three columns, the images in the first column are the original images fed as input to the proposed model, followed by the segmented image in the second column and the final output of our model i.e., the Depth Map in the third column.

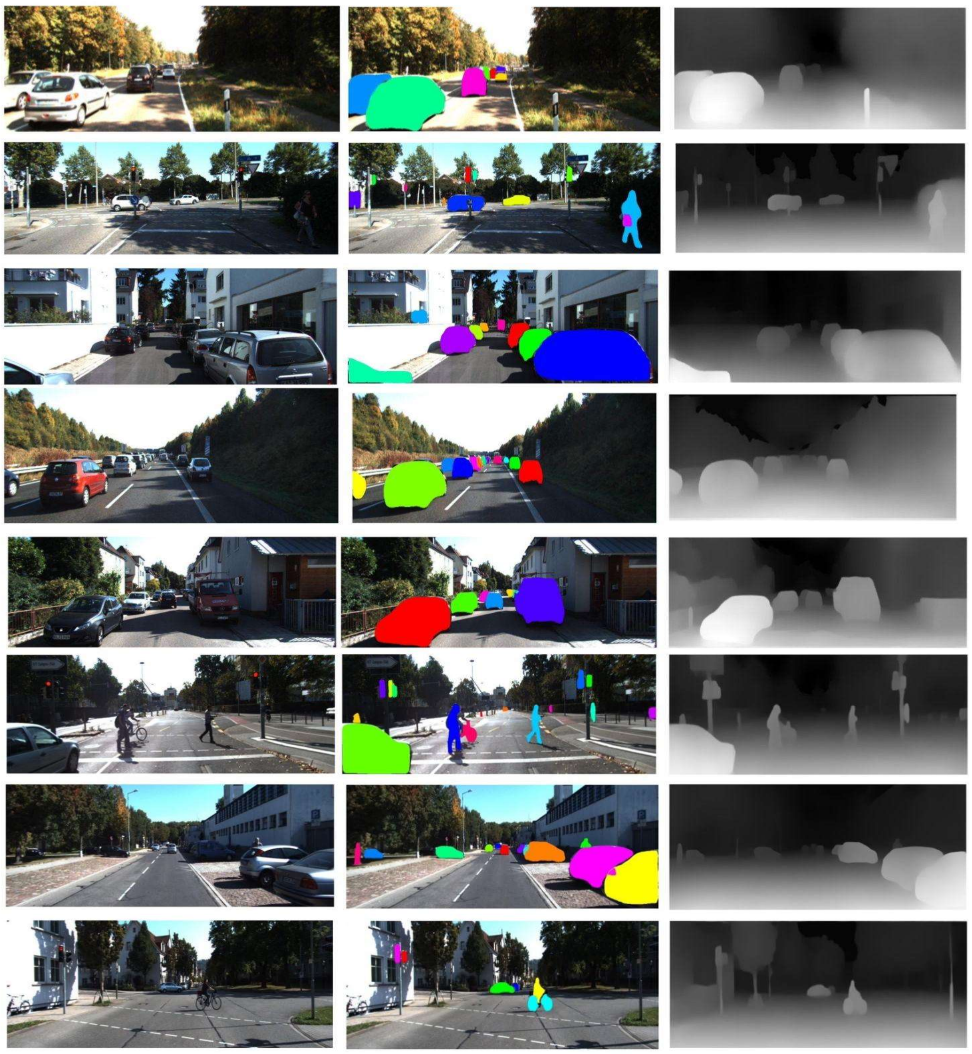

Fig. 4. Column 1 of the above figure represents the input image from the KITTI data set. Column 2 represents the images after instance segmentation. Column 3 has the depth maps generated from the segmented images.

4.4.2 Indian Road Data-set

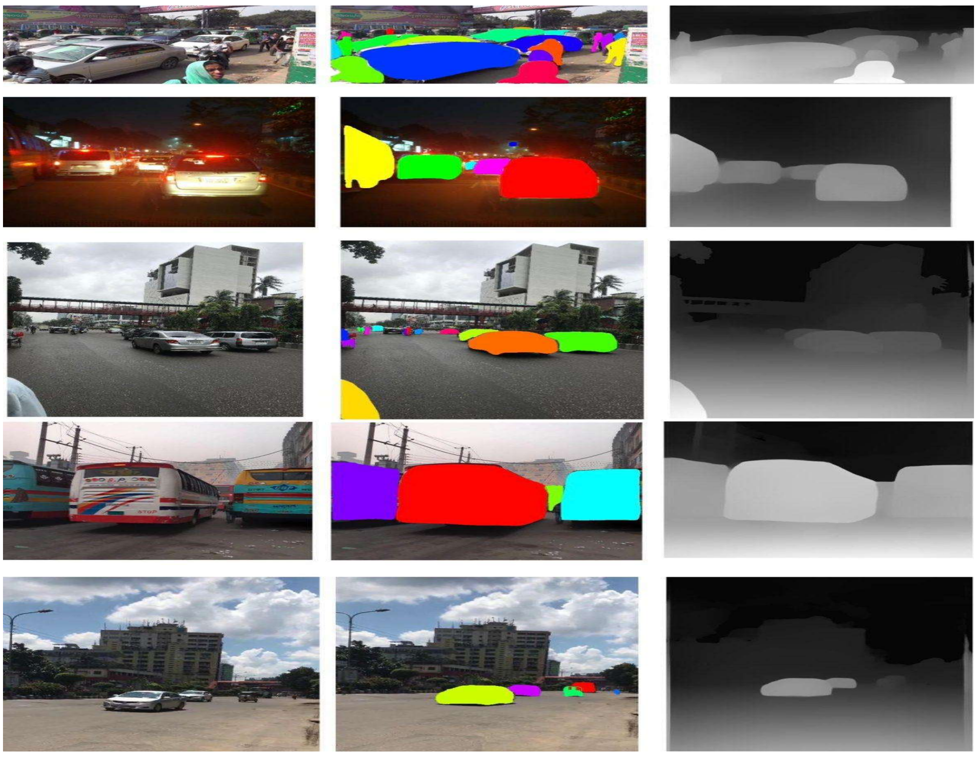

Fig. 5. Column 1 consists of the input image, column 2 contains the segmented images and the last column contains the depth map. The images in Fig 5 are the results of the experiments conducted on Indian road conditions. These image collages also consist of the images in the same manner in the order of original, segmented, and depth map.

5 Conclusion

We proposed a novel methodology for generating an image's depth map by designing a depth map network with DNN and Transformers. Our experiments on depth map generation work on large data sets and different types of input, such as images with different resolutions or aspect ratios. The results were able to capture the farthest vehicle in the given input image and generate the depth map for it. The future scope of depth map generation is expected to remain active, with continued efforts to improve algorithm efficiency and accuracy, as well as the development of new tools and technologies that utilize the depth data. Overall, creating depth maps is an active research topic, with ongoing efforts to improve algorithm precision and efficiency, as well as the development of new technologies and applications that rely on depth data. Depth maps are projected to remain an important tool for understanding the 3D structure of scenes and the relative placements of objects, with the potential to impact a wide range of applications and technologies.

References

1. Yang, J., An, L., Dixit, A., Koo, J., Park, S.I.: Depth estimation with simplified transformer. arXiv preprint arXiv:2204.13791 (2022)
2. Wang, Y., Chao, W.-L., Garg, D., Hariharan, B., Campbell, M., Wein-berger, K.Q.: 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, pp. 8445–8453 (2019)
3. Geiger, A., Ziegler, J., Stiller, C.: Stereoscan: Dense 3d reconstruction in real-time. In: 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 963–968 (2011). Ieee
4. Kyto, M., Nuutinen, M., Oittinen, P.: Method for measuring stereo camera depth accuracy based on stereoscopic vision. In: Three-Dimensional Imaging, Interaction, and Measurement, vol. 7864, pp. 168–176 (2011).SPIE
5. Nasteski, V.: An overview of the supervised machine learning methods. Horizons. b 4, 51–62 (2017)
6. Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1851–1858 (2017)
7. Klingner, M., Term¨ohlen, J.-A., Mikolajczyk, J., Fingscheidt, T.: Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance. In: European Conference on Computer Vision, pp. 582–600 (2020). Springer
8. O'Shea, K., Nash, R.: An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458 (2015)
9. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE transactions on Signal Processing 45(11), 2673–2681 (1997)
10. Hou, X., Shen, L., Sun, K., Qiu, G.: Deep feature consistent variational autoencoder. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1133–1141 (2017). IEEE
11. Ye, M., Johns, E., Handa, A., Zhang, L., Pratt, P., Yang, G.-Z.: Self-supervised siamese learning on stereo image pairs for depth estimation in robotic surgery. arXiv preprint arXiv:1705.08260 (2017)
12. Wang, T.-C., Efros, A.A., Ramamoorthi, R.: Occlusion-aware depth estimation using light-field cameras. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3487–3495 (2015)
13. Mahmood, F., Chen, R., Durr, N.J.: Unsupervised reverse domainadaptation for synthetic medical images via adversarial training. IEEE transactions on medical imaging 37(12), 2572–2581 (2018)
14. Poggi, M., Aleotti, F., Tosi, F., Mattoccia, S.: Towards real-time unsupervised monocular depth estimation on cpu. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp.5848–5854 (2018). IEEE
15. Wang, Z., Ma, Y., Liu, Z., Tang, J.: R-transformer: Recurrent neural network enhanced transformer. arXiv preprint arXiv:1907.05572 (2019)
16. Cai, Z., Vasconcelos, N.: Cascade r-cnn: high quality object detection and instance segmentation. IEEE transactions on pattern analysis and machine intelligence 43(5), 1483–1498 (2019)
17. Ayodele, T.O.: Types of machine learning algorithms. New advances in machine learning 3, 19–48 (2010)
18. Osisanwo, F., Akinsola, J., Awodele, O., Hinmikaiye, J., Olakanmi, O.,Akinjobi, J.: Supervised machine learning algorithms: classification and comparison. International Journal of Computer Trends and Technology (IJCTT) 48(3), 128–138 (2017)

19. Watson, J., Firman, M., Brostow, G.J., Turmukhambetov, D.: Self-supervised monocular depth hints. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2162–2171 (2019)
20. Shu, C., Yu, K., Duan, Z., Yang, K.: Feature-metric loss for self-supervised learning of depth and egomotion. In: European Conference on Computer Vision, pp. 572–588 (2020). Springer
21. Godard, C., Mac Aodha, O., Firman, M., Brostow, G.J.: Digging into self-supervised monocular depth estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3828–3838 (2019)
22. Peng, R., Wang, R., Lai, Y., Tang, L., Cai, Y.: Excavating the potential capacity of self-supervised monocular depth estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 15560–15569 (2021)
23. Casser, V., Pirk, S., Mahjourian, R., Angelova, A.: Depth predictionwithout the sensors: Leveraging structure for unsupervised learning from monocular videos. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 8001–8008 (2019)
24. Godard, C., Mac Aodha, O., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 270–279 (2017)
25. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-timeobject detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 39(6), 11371149(2017).https://doi.org/10.1109/TPAMI.2016.2577031
26. Zhang, S., Wen, L., Bian, X., Lei, Z., Li, S.Z.: Single-shot refinement neural network for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4203–4212 (2018)
27. Cetinkaya, B., Kalkan, S., Akbas, E.: Does depth estimation help object detection? Image and Vision Computing 122, 104427 (2022)
28. He, K., Gkioxari, G., Doll´ar, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969(2017)
29. Girshick, R.: Fast r-cnn. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448 (2015). https://doi.org/10.1109/ICCV.2015.169
30. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: Imagenet:18 Depth Estimation using DNN Architecture and Vision Based Transformers A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848
31. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR) (2013)
32. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
33. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving?the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)