2D Image to 3D Box with Deep Learning Method

Dongjun Wu*

Nanyang Technological University-NTU Singapore, Singapore

Correspondence to: Dongjun Wu, Nanyang Technological University - NTU Singapore, Singapore, E-mail: WU0006UN@e.ntu.edu.sg

Received date: May 25, 2023; **Accepted date:** June 28, 2023; **Published date:** July 05, 2023

Citation: Wu D (2023) U-Pb 2D Image to 3D Box with Deep Learning Method. Sci Rep J. 1:177.

ABSTRACT

This article will show you a method for converting 2d images into 3DBBes. The idea is to split the task into two steps. Firstly, we use neural network to output 3D object features displayed in 2D images. Then, we combine the boundary information of 2D objects in the images to generate geometric constraints on 3D objects, so as to get a complete 3D boundary box. In the first step, we defined an innovative loss function called Combined Orientation Loss (COL). This loss function took into account both the cosine distance and Euclidean distance, so that the length of the direction vector could also be reflected in the loss, which could enhance the sensitivity of the model to the prediction accuracy. In the second step, we return to the dimensions of 3D objects, which are relatively stable and therefore suitable for predicting a wide range of object types. Combining these two steps and the translation constraints of the object in the 2D image, we can accurately and stably restore its 3D border shape. We evaluated our approach on the KITTI target detection benchmark, including the official measure of 3D direction estimates and the accuracy of the resulting 3DBB. In the case of low calculation requirements achieved good results.

Keywords: 2D images; 3D frame; Deep learning; Object features; Geometric constraints; Cosine distance; Euclidean distance; Dimension prediction; Translation constraints

INTRODUCTION

Visual Perception (VP) is a complicated process that allows us to make sense of the vast amount of information that our environment provides. The procedure is made significantly more challenging in the disciplines of Artificial Intelligence (AI) and Computer Vision (CV) since machines are tasked with comprehending and interpreting visual data.

Historically, techniques for transforming 2D photos into 3D have mainly relied on 1 intricate geometry, user input, and a certain amount of guesswork. The scale, accuracy, and adaptability necessary for a variety of actual applications, however, are constrained by these methods. We've witnessed a significant paradigm shift in how to tackle this problem with the introduction of Deep Learning (DL). Deep learning algorithms' adaptability and capacity for learning make them an attractive option for transforming 2D photos into 3D frames.

In this paper, we present a fresh Deep Learning Method (DLM) for generating 3D borders from 2D photos [1]. The technique uses Convolutional Neural Networks (CNNs) to learn reliable 2D to 3D space mapping, allowing the creation of 3D borders from a single input 2D image [2].

In order to generate geometric constraints on 3D Objects (3DOs) and obtain a complete 3D boundary box (3DBB), the main contribution of this paper is to use Neural Networks (NNs) to output 3D object features visible in 2D images. Additionally, by combining 2D object boundary information in the image and improving loss function and data enhancement methods, better results can be obtained quickly by using more advanced Pre-Training Models (PTMs).

The remainder of this essay is structured as follows: We describe how to restore solid pose borders of objects from 2D photos in box mapping, go into detail about how deep learning is used to this method in Deep Learning Application, give experimental data and discuss it in Experiments and Analysis, and then summarize this paper in Discussion.

3D Box Mapping

Our study strengthens the accuracy of 3DBB estimate by extending the resilience of 2D object identification [3]. Our innovative method is based on the idea that a 3DBB's precise projection ought to perfectly match the matching 2D detection window. This alignment assumes a 2D object detector that is optimized and generates boxes that contain an accurate projection of the 3DBB [4].

A 3D boundary frame (3DBF)'s center point $T = [t_x, t_y, t_z]$, dimension $D = [d_x, d_y, d_z]$, and direction $R(\theta, \phi, \alpha)$, parameterized by azimuth, elevation, and roll Angle, are its distinguishing characteristics [5]. Given that the camera's intrinsic matrix K and the coordinate system (R,T) ∈ SE(3), the three-dimensional point Xo = [X, Y, Z, 1] is projected into the object's coordinate system as $X = K[R,T]X_o$ [6].

The vertex of the three-dimensional boundary frame can be exactly determined given the size of the object D and the assumption that the origin of the object coordinate system coincides with the center of the three-dimensional boundary frame. This is further supported by the requirement that every boundary box's 2D edge matches to at least one box's 3D corner. This makes sure that the 3DBB fits precisely inside the 2D detection window.

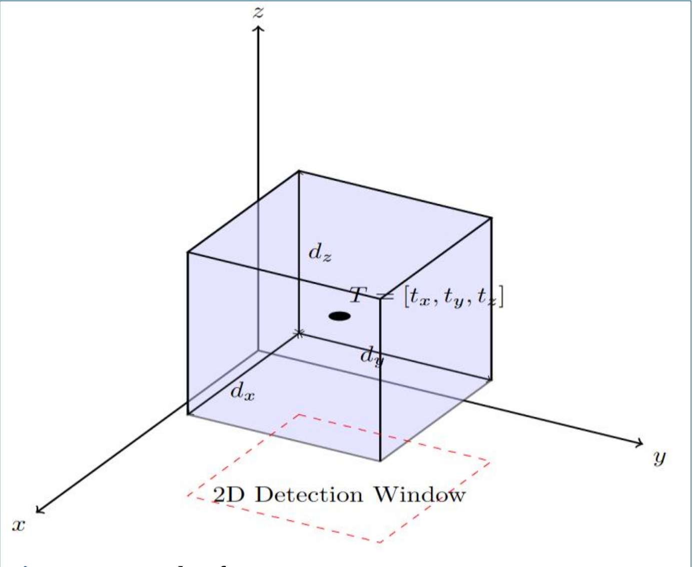

Figure 1: Sample of Mapping.

The orientation around each axis (*θ*, *ϕ*, *α*) and the box size D are the main factors that govern the generation of 3DBBes in this method. We return the box size D using a better way than shifting *t*. This is due to the fact that even if the object's orientation changes, the differences in size estimates and stability are typically negligible. Additionally, there is a direct relationship between size estimation and the presence of particular target subcategories, and correct subcategory classification enhances size estimation recovery accuracy.

Deep Learning Application

In our work, we provide a multi-tasking learning method for predicting vehicle size, orientation, and confidence from a single image [7]. We make use of the well-known ResNet-50 model architecture [2], a well-liked deep residual network noted for excelling at large-scale picture identification tasks.

Data Preprocessing

During the data preprocessing stage, we prepare the images for the deep learning model and normalize the directional data from the vehicle. The image is transformed into the typical shape of (224,224,3) to match the ResNet-50 model's input size. To account for the directional data's cyclical character, the data is transformed into a discrete bin representation [8].

In this section, I introduced further data augmentation techniques because many of the earlier implementations had low resolution of items whose colors were close to the surroundings or objects that were not entirely included in the picture. We expect color transformation to solve the recognition of things whose colors are close to the environment and clipping to solve the recognition of items not fully included in the picture. Rotation, scaling, clipping, flipping, and color transformation methods are included.

Network Architecture

Our network's design is based on the ResNet-50 model [2], which was previously trained using the ImageNet dataset [9]. We use the learnt features from ImageNet to pre-train our model, acting as an automatic feature extractor for our photos. To fine-tune our model for our particular goal, we do, however, permit the final four layers of the base model to be trainable [10].

For each of our prediction tasks, the network output is flattened after the base model and then fed into distinct fully-connected (Dense) layers. Predicting the vehicle's size, orientation, and degree of confidence in the forecast are among the tasks.

The network goes through a Dense layer of 512 nodes with L2 regularization before passing through a LeakyReLU activation function with a negative slope of 0.1 to forecast the vehicle dimensions. The network is then regularized with a dropout layer at a rate of 0.5 [11]. After that, a third Dense layer with three nodes and another LeakyReLU activation function are applied to the output.

Similar to the dimension prediction, the vehicle orientation prediction begins with a Dense layer of 256 nodes, followed by a LeakyReLU and Dropout layer. The output is then subjected to another LeakyReLU activation and another Dense layer with a size equal to twice as many bins. The output is then reshaped to the number of bins by -1 and normalized using a Lambda layer.

The network uses the same initial structure for the confidence level prediction as it does for the orientation prediction. The output is then sent through a Dense layer with a size equal to the number of bins after the Dropout layer. A softmax function is used as the final activation, and it produces a probability distribution across the bins that shows how confidently the prediction may be made.

The final model is then constructed using the results from the dimension, orientation, and confidence predictions like Figure 2 below.

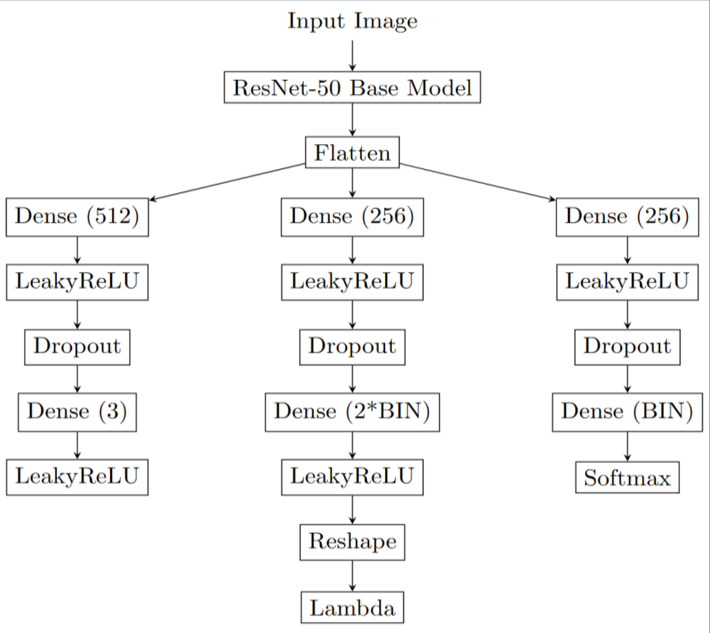

Figure 2: Network Based on ResNet50.

Training Process

With a learning rate of 0.0001, the Adam optimizer [12] is used to train our model. The loss function is the custom combined directional loss function of the "direction" outputs, as well as the mean square error of the "dimension" and "confidence" outputs. One more of our innovations is this directional loss function. The fundamental idea of the basic loss via the cosine distance is to compare the cosine distance between the real value and the anticipated value in order to measure the mistake rather than directly utilizing L2 loss. Additionally, our loss function mixes Euclidean and cosine distance losses. The length of the direction vector can also be reflected in the loss using this loss function, and the difference between the anticipated and actual direc- tions can be calculated. This loss function accounts for both the cosine distance and the Euclidean distance. This may make the model more sensitive to forecast accuracy.

With ten times the patience, we were able to reach an early stop to prevent overfitting [13]. When a new minimum validation loss is obtained, our model is also set up to save its weight, facilitating model checkpoints and allowing us to quickly recover the ideal model even if training is abruptly terminated.

90% and 10%, respectively, of the data are accounted for by the training set and the verification set. To avoid any bias resulting from the sequence of the data, we made sure to shuffle the sets before training. For demonstration reasons, the model is trained with more than 5 epochs, although more epochs can be utilized to produce more accurate results.

EXPERIMENTS AND ANALYSIS

Experiments Setup

The following elements are utilized throughout the experimental setup:

A model that has been trained to do a 2D to 3DBB transition.

The KITTI dataset.

Python code for reading the dataset, using the model to make predictions, and assessing those predictions.

The pre-learned weights from the file ”weights.hdf5” are imported into the trained model. The 2D bounding box annotations and picture data are read from predetermined directories. From the 2D bounding box data and the photos, the model predicts the 3DBB parameters. The anticipated parameters are used for both picture viewing and file storage.

The 2011_09_26_drive_0014 sync dataset from KITTI is used, and it has been processed to separate the data, predictions, labels, and calibrations into their proper directories. Images, 2D bounding box annotations, calibration data, and other sorts of data are among the many various types of data included in the dataset.

The used Python code is built using a variety of packages, including Keras, OpenCV, and numpy. The dataset is read, predictions are made using the model, files with the predictions are written, and the predictions are visualized.

Result And Analysis

Two-dimensional boundary box annotations are read for each image in the data set, and then the related image patches are extracted, normalized, and input into the model. The parameters of the appropriate 3DBF are then predicted by the model (dimension, orientation, confidence).

The direction and size of the boundary frame were modified based on the results of the model’s prediction. This is accomplished by multiplying the Angle by the directional offset and the average vehicle size by the expected dimensions. The 3DBB file is then updated with these adjustments.

In the experiment, calibration data are also used. The operation of flipping, rotating, clipping, and scaling in the data enhancement method will modify the original boundary and center data of the image since I have added a lot of data enhancement methods; therefore, it is required to re-calibrate the data after the operation. The calibration information is used to translate the 3D coordinates from the Velodyne coordinate system to the camera coordinate system after being read from the relevant files. The 3DBB’s center coordinates are subjected to these changes.

On the image are drawn 3DBBes for viewing purposes. This is accomplished by using calibration data to project the vertices of the 3DBB onto the picture plane.

With the help of 2D boundary frame data, the experiment successfully predicts and visualizes 3DBF characteristics. The visualization demonstrates the model’s ability to predict 3DBF parameters with accuracy.

From the perspective of the image (Figure 3 and Figure 4), our improvement has been successfully realized. For the car not fully displayed on the left side of the image and the white car in the middle of the image, the red border has been successfully recognized and generated, which indicates that the idea we adopted is correct, and the improvement of adding data enhancement method and modifying the model is also effective.

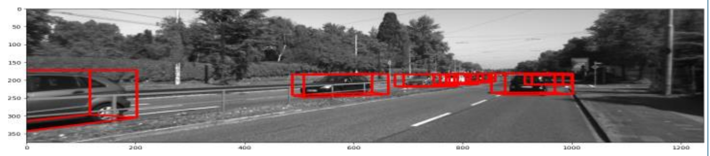

Figure 3: Sample of Visualization 1.

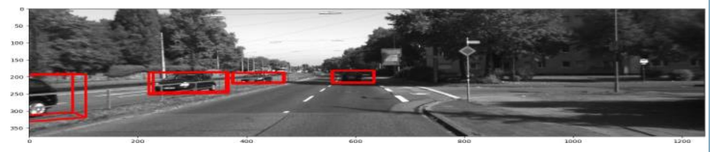

Figure 4: Sample of Visualization 2.

DISCUSSION

In comparison to the previous method, we improved the loss function by combining the cosine distance and the Euclidean distance, ensuring that we were able to train both features together. In addition, we added a number of data enhancement methods to ensure that our model can perform better in certain special scenarios, such as recognizing some light-colored vehicles and recognizing some vehicles that are not fully displayed in the scene.

From the experimental results, we successfully identified objects that were either light in color or not fully displayed in the scene. This means that our improvements are effective and that our approach performs better when dealing with these particular scenarios compared to previous approaches.

However, when we looked at the training process, we found that with the addition of Euclidean distance, our loss learning required more training than before. In other words, the cost of adding more rigorous loss measurements is a greater amount of training. This is a limitation of our model that we need to address in future work.

Therefore, our future work direction is how to further improve the loss measurement, and how to simplify the training process to improve the training efficiency and performance of our model.

CONCLUSION

We demonstrate in this study how to reconstruct a 3DBB of a recognized object cate- gory from a single view. Our approach effectively chooses the box size as a regression parameter to estimate stable and accurate 3DBBes without the need for additional 3D shape models, and it leverages a new MultiBin loss for direction prediction.

To effectively employ time information to estimate the position and velocity of future objects, one future approach is to apply and enhance our algorithms.

Appendix A KITTI Dataset Introduction

The KITTI dataset, a publicly available dataset for autonomous driving and computer vision, was created in collaboration with the Karlsruhe Institute of Technology in Germany and Toyota's US Information Technology Laboratory. The dataset contains detailed annotated data for a variety of tasks including: stereo vision, optical flow, visual ranging, 3D object detection, and semantic segmentation of road and city scenes.

The data set was collected in a variety of road conditions, including city streets, rural roads, and highways. The data is collected using a variety of sensors mounted on the car, including stereo cameras, lidar, and GPS/IMU units.

Specifically, the main components of the KITTI dataset include:

Stereo vision: Including 200 training image pairs and 200 test image pairs, image resolution is 1242x375 pixels.

Optical flow: Including 200 training image pairs and 200 test image pairs, image resolution is 1242x375 pixels.

Visual ranging: including 22K training images and 11K test images, image resolution is 1242x375 pixels.

3D object detection: including 7,481 training samples and 7,518 test samples, including image data and 3D point cloud data.

Semantic segmentation of road and urban scenes: 289 training images and 290 test images were included, and the image resolution was 1242x375 pixels.

This data set is already widely used in autonomous driving and computer vision research, and has already facilitated many important advances in these areas. In our article, we use the semantic segmentation of road and urban scenes part.

Appendix B Network Code

```
# Construct the network

# Use ResNet or similar more modern architecture from keras. applications. resnet import ResNet50

from keras. layers import Dense, Input

from keras. layers. activation import LeakyReLU

from keras. layers. Convolutional (Conv2D, Convolution2D MaxPooling2D, ZeroPadding2D)

from keras. layers. core import Dense, Dropout, Flatten, Lambda, Reshape from keras. models import Model

from keras. regularizers import l 2

from parameters import BIN

from pre processdata import l2_normalize

#Load ResNet model, pre-trained on ImageNet; exclude top FC layer base_model = ResNet50 (weights =' imagenet', include top=False, input shape = (224, 224, 3))

#Make base model layers non-trainable for layer in base model. layers: layer. trainable = False

#You can choose to retrain some of the higher layers for layer in base model. layers [−4:]:

layer.trainable = True

# Now add your custom layersx = base model. output

x= Flatten ( ) ( x)

# Regularization  strength reg_strength = 0. 01

dimension=Dense (512, kernel_regularizer=12 (regstrength)) (x)
dimension = LeakyReLU(alpha = 0.1)(dimension)

dimension = Dropout (0. 5) (dimension)

dimension = Dense (3) (dimension)

dimension = LeakyReLU (alpha=0.1, name='dimension') (dimension)

orientation = Dense (256, kernel_regularizer=12 (regstrength)) (x)

orientation = LeakyReLU(alpha = 0.1) (orientation)

orientation = Dropout (0.5) (orientation)

orientation = Dense (BIN∗ 2) (orientation)

orientation = LeakyReLU(alpha = 0.1) (orientation)

orientation = Reshape ((BIN, −1)) (orientation)

orientation = Lambda (12_normalize, name=' orientation') (orientation)

confidence = Dense (256, kernel_regularizer=12 (reg_strength)) (x) confidence = LeakyReLU(alpha = 0.1) (confidence)

confidence = Dropout (0.5) (confidence)

confidence = Dense (BIN, activation ='softmax', name=' confidence') (confidence) model = Model (base_model.input, outputs =[dimension, orientation, confidence])

#  model. Load_weights (' initial weights. h5')
```

REFERENCES

1. Zhou Y, Tuzel O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4490–4499 (2018).
2. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016).
3. Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? The kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3354–3361 (2012). IEEE.
4. Ren S, He K, Girshick R, Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015).
5. Xiang Y, Choi W, Lin Y, Savarese S. Subcategory-aware convolutional neural networks for object proposals and detection. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 924–933 (2017). IEEE.
6. 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).
7. Geiger A, Lenz, P, Stiller C, Urtasun R. Vision meets robotics: The kitti dataset. The International Journal of Robotics Research 32(11), 1231–1237 (2013).
8. Lin TY, Goyal P, Girshick R, He K, Doll´ar P. Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988 (2017).
9. Deng J, Dong W, Socher R, Li L.-J, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009).
10. Huh M, Agrawal P, Efros A.A. What makes imagenet good for transfer learning? arXiv preprint arXiv:1608.08614 (2016).
11. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15(1), 1929–1958 (2014).

12. Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

13. Prechelt L. Automatic early stopping using cross validation: quantifying the criteria. Neural networks 11(4), 761–767 (1998).