Application of ADAS Multi-sensor Vision Simulation System for Tree Recognition in Urban Garden Environment

Huang Qingqing, Xie Du, Liu Wenbo, Chen Lairong, Liu Jinhao
Engineering School, Beijing Forestry University, Beijing 100083, China

Abstract

The urban garden environment is common in the study of traffic vision, and the identification of tree objects is a field that is rarely involved in current advanced driver assistance system (ADAS) research. To test the practicality and reliability of the multi-sensor based ADAS software model, we built a virtual driving scenario. This method enables the designer to perform the feasibility and reliability analysis of the design scheme quickly and effectively without the need of actual vehicle testing, and improve the efficiency of the prototype experiment. This paper proposes a tree recognition method based on virtual environment. First, use Unity3D to build a visual simulation system, and then import the vehicle model for testing and ITS intelligent transportation system. Unity cameras can output images and videos captured by on-board multi-sensors, where video can perform SIL tests on ADAS, providing a new way to improve ADAS software; images include depth, optical flow, target bezel, trajectory, pixel level Semantic segmentation, instance segmentation and other annotation information, it is a new virtual dataset ParallelEye, and finally carried out experiments on Faster R-CNN, verifying the validity of the data set for tree recognition.

Key words: Traffic Vision, Virtual Training, Images Dataset, Tree Recognition

1. Introduction

With the development and application of intelligent technology in the field of automobile driving, the automotive industry is transforming and developing into the intelligent direction of self-driving. The main form of expression as a smart car is autonomous driving. However, information identification under complex traffic scenarios still faces certain technical challenges [1]. For example, driverless vehicles need to identify and analyze weather changes and road complexity to achieve automated driving. At present, autonomous vehicles have some defects in scene-aware technology in complex traffic environments, and tree identification in traffic environment is a large vacancy in autonomous driving technology. Nowadays, with the concept of greening environment and eco-city being implemented, trees become a ubiquitous and inevitable existence of urban traffic. Trees, as an immovable obstacle, in the autonomous driving environment (urban roads, highways, green parking lots, etc.) have an important influence on the discrimination and decision-making of the automatic driving algorithm [2]. Usually the driver of the car achieves his driving goals through his own pre-behavior. For example, in the driving process of a car, it is necessary to prejudge the behavior of the driver of the side vehicle to decide whether to evade the vehicle, Autopilot technology mainly recognizes objects through deep learning training methods. Under the traditional method, there are few real image sets for deep learning, it is difficult to carry out large-scale training, and the real image set acquisition and labeling cost is high, and it is difficult to cover complex natural environments. There are few samples of extreme scenes, and there are also problems such as poor adaptability of trained models. With the development of deep learning technology, data sets for autonomous driving are now extremely rich, such as KITTI, Oxford, Cityscape, Comma.ai, BDDV, TORCS, Udacity, and even with GTA5, a very popular game, now also used to train models for autonomous driving [3, 4]. Most of the labeling objects of these training sets are vehicles and pedestrians. Although trees are widely distributed among the collected images, they are rarely labeled.

Since the same tree may have different degrees of occlusion in front of the camera in different orientations and different perspectives, plants and canopies of the same tree have different sizes and shapes, and it is difficult to label them. Therefore, some researchers have proposed a method of constructing a virtual scene to generate a virtual image set, which solves the problem of high cost and sample shortage of real image dataset acquisition and labeling. In the virtual scene, various factors such as weather, illumination, people flow, and traffic flow can be flexibly changed to meet the needs of data diversity for visual model research, and it has the advantage of facilitating large-scale acquisition of image datasets [5, 7]. In this paper, we propose a tree identification method based on virtual environment. First, use the modeling software such as Unity3D and 3dsMax to build the urban garden environment needed, and then import the vehicle model for testing and ITS intelligent transportation system. And adding multiple sensors to simulate the ADAS in the vehicle model in the virtual scene, write,
compile, and debug scripts for virtual driving vehicles and cameras so that Unity cameras can output images and videos captured by on-board multi-sensors. After compilation, the video stream is edited and stored to realize the output of the video stream. This video enables SIL testing of ADAS, providing a new way to improve ADAS software. Using the internal mechanism of the virtual reality tool to process various types of target information of the scene, it can automatically obtain detailed and accurate annotation information, including depth, optical flow, target border, trajectory, pixel-level semantic segmentation, and instance segmentation, we use these image datasets to train the model and compare it to KITTI.

2. Virtual Car Collision Property Settings

2.1. Vehicle dynamics model structure based on Unity3D

The virtual car being tested in the virtual test system is an important interaction model, and including all nonlinear characteristics, it’s dynamic characteristics display is very realistic, the vehicle platform and axle model all modelling by three-dimensional multi-body system, and can automatically perform parametric modelling based on the data of car testing. The multi-body model is an integrated model of the integration of various subsystems of the car, including tires, engines, chassis and aerodynamic modules. These modules can be modified or replaced with homemade submodules, or replaced by the real hardware.

When building a car model, attention should be paid to the quantity of point line planes in the model structure. If the number of these things is too large, many twisted edges or planes will be generated, which will affect the driving performance of the car model in the platform to some extent, so it is imported. Before the model, you need to simplify the process to remove the extra points and lines.

2.2. car model drive parameter binding

The vehicle model physical properties have been given while parameterizing the car model. The motion of the vehicle is described by a coordinate system, so the coordinate system direction of the car model is related to its driving in the virtual environment. In the established car model, the coordinate system of the vehicle body must be at the centroid position of the model, and the wheel is also treated the same. According to the requirements of the behavior module, it can be rotated to the desired direction with a small rotation function: the blue coordinate axis Forward, the green axis points up and the red coordinate axis is on the right. When you import this model, due to the different coordinate systems between 3DMAX and Unity 3D, the 3ds file will realign the coordinates to the homing, so we import files with a .3ds suffix.

2.3 Collision detection technology

Collision detection is used to simulate the instinctive reaction of an object when it encounters an obstacle in a real environment. For example, when a car encounters a building, it automatically stops and cannot move forward. The need for virtual scene simulation systems provides good conditions for collision detection. The collision of an object can be divided into two parts: detection and response. The detection is to detect the position of the collision when the collision occurs, and the response is to respond to the different degrees of the collision after the collision occurs.
Common methods of collision detection include spherical bounding box detection and hexahedron bounding box detection. The two methods are as follows:

In the field of computer graphics and computational geometry, a bounding box of a set of objects is a closed space that completely embraces the combination of objects. By encapsulating complex objects in a simple bounding box and simply replacing the shape of the complex geometry with a simple bounding box shape, the efficiency of the geometric operations can be improved. And usually simple objects are easier to check for overlap. The hexahedron bounding box contains six surfaces, twelve lines and eight vertices. Therefore, these related elements need to be considered when using the hexahedron bound box detection method. So, it is necessary to detect collisions between objects in three aspects:

<1> Collision of each vertex with other hexahedrons.
<2> Collision of each side with other hexahedrons.
<3> Collision of each surface with other hexahedrons. The collision between surface and surface is only possible if the two objects are parallel to each other, so the probability of occurrence is much lower.

The collision characteristics can be set in the property window and set to dynamic continuous detection, that is, when multiple cars meet or encounter a static wall, the car will collide and stop.

The collider uses continuous collision detection when it encounters other objects that are set to be continuous or dynamically continuous. The collision detection set in this paper is suitable for fast moving objects.
3. Urban Garden Environment Settings

Urban garden environment is common in parks and other green parking lots, villas and residential quarters. This paper uses open-air parking lots as an example to build an urban garden virtual environment with the help of 3DsMax and Unity3D. In this scene, there are models of buildings, highways, cars, parking lots, trees, bushes, pedestrians, etc. In order to generate a more easily labelled training set later, we need to simplify the virtual scene.

![Figure 4. Parking lot model created by 3DsMax](image)

First, use 3DsMax to create a simple open-air parking 3D model. This model contains several sub-models, such as restaurants, cafes, newsstands, telephone booths, parking lanes, etc. Then add Texture to the model just created, and then export the entire model to the ‘.fbx’ format for later import into Unity3D for further processing.

Open Unity3D, create a Terrain on the Scene panel, import the ‘.fbx’ file, and drag it into the Scene panel and place it on the Terrain. Adjust the Terrain scale and give it a texture. Use the Terrain component to generate trees at the place that you want.

In order to quickly produce urban buildings without cost too much computer performance, we can use the following methods to make batches: First create a cube and adjust its size, add pre-prepared materials to it, different materials represent different buildings, and drag them into the Prefabs under the Assets folder. Then you can arrange them trimly into the scene.

![Figure 5. Quickly make apartment buildings](image)

Place several static car models in the scene, all of which are available from the Unity3D Asset Store. Put the previously prepared vehicle with the control script in Scene., and set this vehicle as the Player. At this point, our urban garden scene is basically completed.
4. Multi-sensor Dynamic View Simulation in Unity Environment

Actually, there are no sensor simulation modules (3D scanners, laser radars, etc.) in the Unity environment like the automotive ADAS system. But we can use the camera components and collision detection in U3D to simulate the work of these sensors. That is to say, the various cameras mounted on the car in the Unity scene are virtual sensors for capturing information such as images. For this purpose, special scripts need to be added to these cameras to generate the training sets we need.

The camera is an important resource for dynamic view control in Unity3D. In this modelling, according to the requirements, several cameras in different orientations are set in the virtual environment, and set a series of settings for each camera's basic properties and script properties, it can realize multi-azimuth camera to follow the car smoothly and generate video data stream. The ParallelEye dataset is a typical multi-sensor system training set generation method.

The ParallelEye dataset released by the OpenPV platform consists of seven sub-data sets, including 40,251 virtual images and six annotations. Sub-datasets 1–3 provide bounding frame information for three types of traffic targets (car, bus, and truck) that can be used for target detection studies, Sub-datasets 4–7 contain more target types and give more annotation information, including the bounding box of the target, pixel-level semantics/instance, depth, etc., can support various visual computing tasks such as target detection and tracking, semantic/instance segmentation, and depth estimation. Each sub-data set corresponds to one artificial scene, and the same sub-data set contains several pieces of video, corresponding to the image data collected by the artificial scene under different illumination and weather conditions. Figure 6 shows an example of the annotation information for the ParallelEye virtual image set.
5. Algorithm of Multi-Sensor to Generate Training Set

As mentioned earlier, the sensor in the virtual scene mentioned here is actually a camera component with the script ‘ImageSynthesis.cs’. But how does it capture the image and generate the training set?

First, ‘ImageSynthesis.OnSceneChange()’ calls ‘ColourEncoding’ class to encode unique object identifier and layer as RGB colour. These colours are stored in ‘MaterialPropertyBlock’ for each object and are automatically passed into the shaders when rendering. Upon start ‘ImageSynthesis’ component creates hidden camera for every single pass of output data (image segmentation, optical flow, depth, etc). These cameras allow to override usual rendering of the scene and instead use custom shaders to generate the output. These cameras are attached to different displays using ‘Camera.targetDisplay’ property - handy for preview in the Editor. Data can be previewed right inside the Editor (in Play mode) via ‘Display’ drop down in the Game View. ‘Display 1’ provides the final image as usual, while ‘Display 2’ to ‘Display 5’ - displays the additional data like image segmentation, optical flow, etc.

For Optical flow and Depth, cameras request additional data to be rendered with ‘DepthTextureMode.Depth’ and ‘DepthTextureMode.MotionVectors’ flags. Rendering of these cameras is followed by drawing full screen quad ‘CommandBuffer.Blit()’ with custom shaders that convert 24/16bit-per-channel data into the 8-bit RGB encoding.

For Image segmentation and Object categorization pass special replacement shader is set with ‘Camera.SetReplacementShader()’. It overrides shaders that would be otherwise used for rendering and instead outputs encoded object id or layer.

Finally, images are readback with ‘Texture. ReadPixels()’ from GPU, compressed with ‘Texture.EncodePNG()’ to PNG format and stored on disk.

5.1 Algorithm of generate identification box for arbor trees

Since the trees generated by using the Terrain component cannot be found in the Hierarchy panel, it is necessary to find tree models of different types to make them as Prefabs, and then manually arrange to replace the original tree in the scene. Unlike previous trees, all trees are given different tags when they are made into prefabs. These tags represent the type of tree (Tree1, Tree2, Tree3, etc.). Then create a script named ‘CreateBillboardForTree’. Define three arrays, Tree1, Tree2, and Tree3, which correspond to three different tags. Call ‘GameObject.FindGameObjectsWithTag’ to find the trees in the scene to traverse them. For obtain verification set with much more reference value, trees that are more than 200 meters away from the car or covered by other objects by more than 40% in front of the camera are not marked. After filtering out the trees, and placing the eligible trees into the corresponding array according to the Tag. Then traverse all the elements in these arrays to get the position of the tree. This position information is where we need to generate the tree identification box. There are also three bounding box prefabs of different sizes (the bounding box needs to completely wrap the trees), which is actually a texture of the transparent material and a canvas (Canvas). We need to add another script
(LookAtCamera) on the canvas so that it always faces the camera of the car. This way all the trees will be marked when the game is running, and we will have these bounding boxes in our original image when capturing images. We can get a massive training sets and verification sets for machine learning from the above work.

Figure 9. ParallelEye training set

6. Test System IO Performance

6.1 Video stream real-time output test

For different needs, Five 60-degree cameras can be configured in Unity3D, including position, effective range and other parameters, the generated video data stream (VDS) is transmitted to the video processing algorithm in a variety of transmission modes, and can be selected according to requirements. In the camera defined above, the user can select up to 5 cameras for video data output, and each camera generates corresponding video data, including generating a depth picture, and the color of the pixel represents the distance between the object and the camera, these video data can be exported to a file separately using tools (Figure 18). Convert video data such as video signals output from the game interface to computer-identifiable digital data by using Video Capture card; It can also be read by other programs through the TCP/IP protocol, and supports multiple video data transmission forms for testing user-developed image processing algorithms and control algorithms (Figure 19).

The video data stream acquired by the camera in Unity3D can also be transmitted to the soft ECU through the TCP/IP protocol, as shown in the following figure:

Figure 10. Soft ECU embedded in Unity3D simulation environment
7. Experimental Process and Results

7.1 Properties of parallelEye urban tree dataset

Finally, a dataset prepared for tree recognition consists 12000 frames taken from a virtual camera moving through the virtual parking lot for 3 times, with its height limited to the range of [1.1m, 1.5m] above the ground. In the experiment, our scenes can run at 8-12 fps (frames per second), and the frames produced by click of random time manually. This dataset includes seven categories of targets (tree, car, bus, truck, pedestrian, motorcycle, and bicycle) with labels for object detection and 12 categories of labels for semantic segmentation (including sky, car, vegetation, building, fence, sidewalk, road, traffic sign, traffic light, pole, other traffic tools, and pedestrian) compared to the original ParalleEye dataset with six categories of targets.

It is easy to find that there are Tree like objects in our dataset, compared to the image in ParallelEye_04 and ParallelEye_07 and other sub-datasets, under different environmental. The images are created at regular intervals the driving simulator is used for data collection.

7.2 Experimental Result

Experiments for Object Detection on KITTI: Using the KITTI dataset, we investigate the influence of training images with different environmental conditions on object detection performance.
8. Conclusions

In this paper, we built a virtual driving scenario. This method enables the designer to perform the feasibility and reliability analysis of the design scheme quickly and effectively without the need of actual vehicle testing, and improve the efficiency of the prototype experiment. We verified the feasibility and effectiveness of the scheme through experiments.

Acknowledgements

This work was support by Beijing Natural Science Foundation (6184044), and experiment also funded by National Key Research and Development Program of China(2016YFE0203400).

References


