



Building 3D Virtual Worlds from Monocular Images of Urban Road Traffic Scenes

Ankita Christine Victor and Jaya Sreevalsan-Nair

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

October 3, 2021

Building 3D Virtual Worlds From Monocular Images of Urban Road Traffic Scenes

Ankita Christine Victor and
Jaya Sreevalsan-Nair*[0000-0001-6333-4161]

Graphics-Visualization-Computing Lab,
International Institute of Information Technology, Bangalore,
26/C, Electronics City, Karnataka 560100, India
jnair@iiitb.ac.in
<http://www.iiitb.ac.in/gvcl>

Abstract. Three-dimensional (3D) modeling of urban road scenes has warranted well-deserved interest in entertainment, urban planning, and autonomous vehicle simulation. Modeling such realistic scenes is still predominantly a manual process, relying mainly on 3D artists. Cameras mounted on vehicles can now provide images of road scenes, which can be used as references for automating scene layout. Our goal is to use the information from such images from a single camera sensor on a moving vehicle to build an approximate 3D virtual world. We propose a workflow that takes the human out of the loop through the use of deep learning to generate a dense depth map, an inverse projection to correct for perspective distortion in the image, collision detection, and a rendering engine. The engine loads and displays 3D models belonging to a particular type, at accurate relative positions, thus building and rendering a virtual world corresponding to the image. This virtual world can then be edited and animated. Our proposed workflow can potentially speed up the process of modeling virtual environments significantly when integrated with a modeling tool. We have tested the efficacy of our 3D virtual world-building and rendering using user studies with image-to-image similarity and video-to-image correspondences. Even with limited photorealistic rendering, our user study results demonstrate that 3D world-building can be effectively done, with minimal human intervention, using our workflow with monocular images from moving vehicles as inputs.

Keywords: 3D World-Building · Modeling · Convolutional Neural Networks · Deep Learning · Depth Estimation · Inverse Projection · Collision Detection · Graphics Rendering · Virtual World · Human-in-the-loop

1 Introduction

Building a three-dimensional (3D) world of a scene from a two-dimensional (2D) image is a challenging problem in computer vision. Its technological solutions

* Corresponding author

have applications in entertainment, digital mapping, urban planning, and training simulations for automated driving systems. Testing of autonomous driving software is critical for validation. It is estimated that autonomous vehicles might have to be tested on up to 17.7 billion kilometers of simulation before one can have valid data on their safety to compare with human drivers [19]. While one could record the footage for this entire (simulated) distance using cameras, it is more advantageous to use virtual worlds that can be modeled from real-world traffic data, edited, and animated to produce a variety of scenarios.

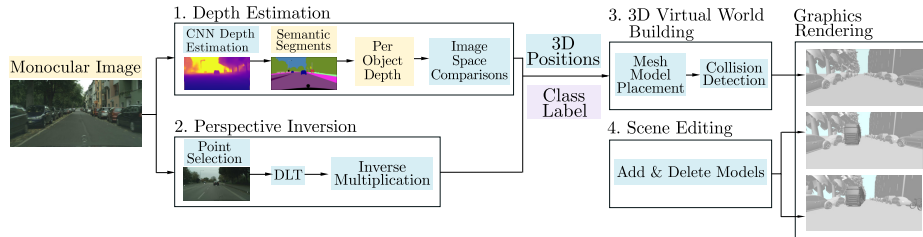


Fig. 1. Overview of our proposed workflow. Given a monocular image, we (1) estimate dense depth using a CNN and obtain per object depth using semantic segments, and (2) correct perspective distortion DLT using selected points. Using the 3D positions and object labels obtained from (1) and (2) for our classes of interest, we (3) place 3D mesh models followed by an AABB collision detection, and lastly, (4) interactively edit the scene. We examine the outcomes of building and editing by rendering the virtual world(s) using a rendering engine, e.g., a basic one using OpenGL, as shown here.

Most methods to automate the process of 3D world-building use data from a combination of sensors and modalities which typically require an elaborate or expensive setup with multiple cameras and sensors. With the onslaught of digital camera devices, the Web has become a source of billions of images and videos of traffic scenes. This image/video data is now used as the starting point for modeling 3D environments, thus eliminating complex data acquisition. In tandem, advances in machine learning and computer vision provide the tools to augment 2D images with 3D information and create new uses for these images.

However, 3D geometric reconstructions from image and sensor data that can directly be used as a high-quality 3D world are still a work in progress. As a result, world-building remains a manual process. Most 3D artists still follow the same workflow that usually starts with gathering inspiration and ideas from the Web, making a simple construction of the scene using basic objects to mock the scene layout. The scene is then lit with the main lights and detailed.

Two important metrics of the modeling process are the extent of visual realism, *i.e.*, how close in appearance an artist can make the virtual environment look when compared to the real world, and performance, *i.e.*, how fast these scenes can be modeled. Sophisticated modeling tools assist in realizing an artist's cre-

ative ideas rather than participating in the building process itself. In this regard, a technology that is capable of delivering an animatable, realistic 3D world with minimal human input is an interesting focus of research that we believe has potential for popularizing the usage of 3D worlds. To summarize, there is a demand for production-ready 3D content that resembles real-world scenes and that can be easily edited and animated. We then pose the following research question: Is it possible to design an automated system for world-building from a monocular image captured by a camera in a moving vehicle?

To enable image-to-3D world-building, we propose a workflow that uses a convolutional neural network (CNN) to estimate the depth and computes a matrix to correct the effects of perspective projection using direct linear transform (DLT). Our proposed workflow automates the initial steps of 3D scene modeling up to the stage where the primary objects in the scene have been placed, and the scene is rendered with ambient light. There are four main steps in the workflow, namely, depth estimation, perspective correction, model loading, and interactive editing (Figure 1). Our workflow exclusively builds 3D virtual worlds of urban road traffic scenes with paved roads, parked or stationary, and moving traffic, people, trees, and buildings. Our novel contributions are in:

- An automated workflow for the building of 3D virtual worlds from monocular images of traffic scenes captured from a moving vehicle, that can be further edited and animated.
- A user study to validate correspondence of the 3D world to its source image.

2 Related Work

We refer to creating an approximate world with the objects of the same type placed, as in the scene, as *3D virtual world-building*. Our approach to image-to-3D world-building is novel. World-building and 3D reconstruction have overlapping solutions as they rely on depth inference and camera approximation. However, the former looks at creating animatable scenes with prefabricated models using relative positions of key objects, and the latter is focused on identifying sizes and types of objects present in the scene. While both can be used in tandem, we focus on the former exclusively here. We discuss literature on both here.

Structure from Motion (SfM): SfM refers to the methods that use observations from two or more viewing directions to derive the position of a point in 3D space by triangulation. Early self-calibrating metric reconstruction systems [2, 18] served as some of the first systems on 3D reconstruction from images. Pollefeys *et al.* [20] used monocular video to deliver 3D models as textured meshes. Incremental SfM is a well-accepted strategy for reconstruction from unordered image collections. Unordered photo collections on the Web have been used to generate sparse 3D models [7, 23]. As an extension, multibody SfM (MSfM) has been used for the reconstruction of dynamic traffic scenes and vehicle/camera trajectories [3]. The camera in these cases is not necessarily mounted on the moving vehicle. Our workflow uses a monocular image captured by a camera

sensor in the moving vehicle, as the input, and gives an approximate 3D world corresponding to the image, unlike the 3D reconstruction of traffic images.

Joint Semantic Segmentation and 3D Reconstruction: Image segmentation and 3D reconstruction can be tightly coupled and jointly implemented, by constraining the solution using priors, thus yielding smoother 3D reconstructions and more precise segmentation. For instance, the surface area has been used as a regularization prior to obtaining the final surface construction indirectly via volumetric optimization [17]. This has been improved by adding class-specific geometric cues guided by image appearances [13]. Joint inference of 3D scene structure and semantic labeling of forward-moving monocular image sequences has been done using class-specific semantic cues and conditional random fields [16]. Geometric cues have been used to simultaneously perform depth estimation from monocular images of traffic scenes and semantic segmentation on a CNN [14].

Generative Adversarial Networks (GANs): GANs have been used as extensions to self-supervised networks for improving depth estimation in monocular images from moving vehicles [12]. Additionally, for image sequences/collections data, GANs are used in image-to-image synthesis tasks and for generating photorealistic images. A noteworthy contribution comes from Isola *et al.* [15] who have proposed a conditional GAN architecture and a corresponding `pix2pix` software as a general-purpose solution for image-to-image translation. `pix2pix` has been tested on a variety of tasks and datasets, including the generation of images from semantic labels trained on the Cityscapes dataset [6]. Conditional GANs have been used for video-to-video synthesis [25]. Similar to `pix2pix`, this method synthesizes images from segmented frames while maintaining temporal coherence. This work has been extended to generate graphics for a driving simulator. One limitation is that the structure of the world is created manually and only the graphics or texture is synthesized. Our work differs in that we infer the scene layout and use off-the-shelf mesh models to create the final scene that can be animated as such. These GAN-based methods predominantly work in cases where the source and destination spaces are of the same dimension, which are not directly applicable for image-to-3D world synthesis.

3D Virtual World-Building: Virtual world rendering from images and LiDAR data has been an active area of research. Modeling of full 3D virtual representations of dynamic events from multiple video streams has been done using a collection of fused visible surface models [21]. There are several methods for the automated extraction of geographic information from LiDAR to support the construction of high-fidelity 3D virtual environment models [1]. Clarke [5] has proposed a pipeline for automated, customized virtual world construction by specifying a set of locations and target style for each location. Our work differs from these in that we consider the construction of approximate environments from a specific data source, namely the monocular images.

Our work is also notably different from the Virtual KITTI dataset generation [4, 8]. The Virtual KITTI dataset consists of photo-realistically rendered

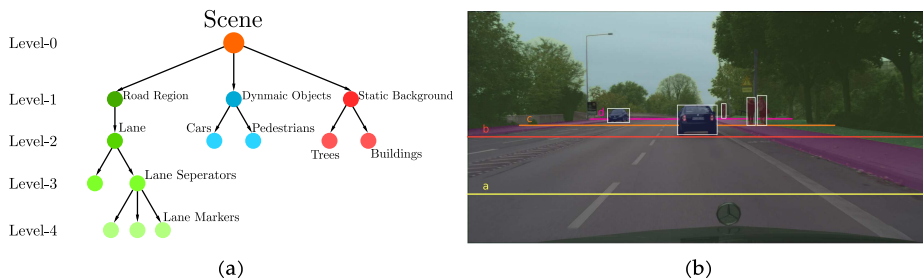


Fig. 2. (a) Interpretation of a road scene using a Latent Hierarchical Part-based Model (LHPM) [24] for the choice of objects in the synthesized 3D world. (b) Depth values along the lines $Y = y$ in image space and on the ground plane can be assumed to have the same depth. Each colored line represents a different value, and all points on the line have the same depth. Note that no line is drawn through objects since those pixels do not correspond to ground points. In terms of Y , $a > b > c > d$, whose order reverses in terms of depth.

videos of image sequences from the camera sensor in the moving vehicles, generated using Unity, which enabled manual world-building through crowdsourcing from the activity user community of Unity. Our work builds approximate virtual worlds using a *mostly* human-out-of-the-loop workflow.

3 Our Method

3D worlds are traditionally constructed manually by 3D artists who take inspiration from or model parts of the real world. In contrast, we propose a workflow to automatically construct 3D worlds of urban road traffic scenes based on a reference monocular image with minimal user intervention. Our proposed workflow is currently intended for images of urban, outdoor scenes with straight roads, that include objects of interest, namely, people, trees, buildings, and sidewalks. Our work pertains exclusively to straight roads owing to the constraints imposed by our perspective correction methodology. We use the Latent Hierarchical Part-based Models (LHPMs) [24] for the choice of objects in the scene which we include in 3D virtual worlds. All objects in the scene are organized in a hierarchical structure in an LHPM. While we conceptually use the tree data structure given in an LHPM, the indexing of the levels in the tree is different in our workflow to suit our requirements. In our current implementation, we have considered level-1 of road regions (*i.e.*, just the roads without identifying structures at finer levels of granularity, such as lanes, separators, etc.), level-2 of dynamic objects (*i.e.*, pedestrians, cars), and level-2 of static background (*i.e.*, trees, buildings), as shown in Figure 2 (a).

Our workflow has four main steps, as shown in Figure 1. In step 1, we use the CNN architecture proposed by Godard *et al.* [11] to estimate depth from a monocular image and ground truth semantic segments to compute per-object

depth for our classes of interest. In step 2, we approximate the projection matrix of the camera that created the image using DLT and apply an inverse of this to obtain world space coordinates. In step 3, we use the obtained 3D position and the semantic class to load a generic mesh model belonging to that class at the computed location. We additionally run an axis-aligned bounding box (AABB) collision detection to eliminate any mesh intersection in the 3D world-building. In step 4, interactive editing of the world is facilitated. We finally render the synthesized world using a graphics engine or shader-based OpenGL software. The novelty of our workflow is in choosing and integrating appropriate methods for our requirement of image-to-3D world-building. We do not perform a comparison with other methods for subcomponents of our workflow, namely, depth generation or perspective correction, as these methods are already well studied, and equivalent methods can be used to replace them in our workflow. Since our work focuses on identifying relative positions of key objects in a scene, we have used the standard validation technique of user studies to evaluate the closeness of the 3D world from our method to the real world.

Step 1: Depth Estimation: Godard *et al.* [11] have posed depth estimation from a monocular image as an image reconstruction problem instead of a supervised learning problem. Their main idea is that given a set of stereo images captured by a calibrated binocular camera, if a network learns a function that can reconstruct one image of a stereo pair from the other, then the network has learned something about the 3D structure of the input image. If the baseline distance b between the two cameras and the camera focal length f are known then depth, \hat{d} can be trivially recovered from the predicted disparity as, $\hat{d} = \frac{bf}{d}$.

Here, given the novelty of our work in 3D world-building, we choose to use Cityscapes dataset over other similar traffic scene image collection, *e.g.*, KITTI [9], owing to the higher resolution, quality, and variety of the images [11]. The recent GANs that are trained on KITTI and generalized on Cityscapes, provide a marginal improvement in per-pixel accuracy of depth estimation [12]. This is especially because batch normalization that is used for adversarial training does not provide significant improvement for the Cityscapes data [11]. Also, for Cityscapes images to work with the GANs, the images are cropped to resolve the artifacts in the images [12]. Overall, the simpler CNN proposed with lower training time by Godard *et al.* [11] is suitable for our experiments, owing to our requirement of the accuracy of relative depths in pixels for world-building as opposed to the per-pixel accuracy. Since the depth estimation component is decoupled from others in the workflow, a depth estimation model suitable for the input data can be selected and *plugged in*.

We use the model that has been trained for 50 epochs, with a 512×256 resolution, a batch size of 8, and a VGG encoder-decoder, using images from both KITTI and Cityscapes datasets [11]. A single input image is run through the depth estimation model to obtain a dense depth map. Using ground truth semantic labels, the average depth of each object of interest is estimated. Depth estimation has been observed to be estimated with higher error especially for occluded objects and objects farther away from the camera [11].

To overcome model limitations and ensure all objects of interest in our work are assigned the right depth value with respect to each other, we propose a post-processing step. Here, we make the assumption that the depth of all points along the same line $Y = y$ in pixel coordinates space and are also on the ground plane in the real world will have the same depth. Y -axis¹ in image space corresponds to the axis along which height of objects is considered in the world space (+ Y -axis). In Figure 2 (b), each colored line represents a different value, and all points on the line (not including the breaks) have the same depth. We compute the bounding box of each segment and consider the Y -coordinate (increases downwards) of the bottom corner to be the point at which the object touches the ground in the image. An object with a higher value of Y must be closer to the camera and have a smaller depth value. We sort the objects in the increasing order of average depth, compare their ground points, and assign depth in the post-processing step, with the following adjustments:

1. If two objects have similar Y values and their average depths differ not less than a certain threshold, δ (we have used $\delta = 1$), then we set the average depth of both to be the average-depth of the closer object.
2. If an object with a higher Y value has greater depth than a closer object, also with a lower Y value, we move the former ‘behind’ the latter in 3D space, along the + Z -axis from the camera.

Step 2: Perspective Correction: Perspective projection is a non-linear, non-affine transformation in Z -axis as lines that are parallel in the original coordinate space appear to intersect in the transformed coordinate space or the projected image. An accurate synthesis of the corresponding 3D world will require that the image coordinates of each object first be remapped to its original world coordinates using perspective correction.

To localize the objects, we use the Z -coordinate estimated in Step 1 and now approximate the X - and Y -coordinates. Since the scenes here pertain to urban roads, we assume that all objects of interest lie on the ground plane, *i.e.*, $Y=0$ for these objects in 3D space. We now use inverse projection to estimate X -coordinate. To construct a 3D world out of an image scene, we compute the value of some matrix P using $I_i = PW_i$, where I_i is a set of projected image coordinates, W_i is the corresponding set of world coordinates and P is a projection matrix. Since all objects of interest have been assumed to be on the ground plane, this simplifies to the 2D case of DLT. Given a point i_i in the image space, z_i can be obtained by a lookup from the corresponding depth map, and x_i can be approximated. The approximation is done for a set of 5-6 selected points on the image with reference to the *vanishing point* in the image. Our experiments show that selecting the nearest ground point from the objects of interest in the scene (in 3D world space) provides the best approximation. We compute P by solving a system of equations given by the world space-image space point correspondences with the requirement that $\|P\|^2=1$ using the least-squares criterion

¹ We refer to the axes in image space as X - Y , and those in 3D world space as X - Y - Z .

and its inverse is used. To obtain the position of each object along the X-axis, we multiply the extents of the bounding box around each segment with the inverse matrix obtained and averaged. The center of the object is now *translated* along X by this value, for the placement of the object/instance in the scene.

Step 3: 3D World-Building: We have manually curated a small database of 3D mesh models based on the classes of interest from Cityscapes. These geometric models are triangular meshes of an instance of the expected class and are in the standard OBJ format. We have fixed the orientation of every model in 3D space as well as set the scale of every object with respect to another other by intuitively comparing unit mesh model sizes. The X- and Z-coordinates, and the class label are sent to the rendering engine. The database is looked up using the class label, and a corresponding mesh model is loaded into the scene at the world space coordinates. For example, if an object in the image has been semantically segmented as ‘car’ the rendering engine loads a generic hatchback or SUV (using a random number generator) to construct the 3D world. A woman in an image is semantically labeled as ‘person’ and the engine will load a generic model of a male human. Similarly, a generic tree is loaded for ‘vegetation’ even if the object in the image might be a shrub. This is acceptable in our workflow as we are focused on creating a virtual world, which mimics the scene in the image.

We position the object mesh in the scene using the position coordinates computed in steps 1 and 2, such that the center of the ‘base plane’ of the mesh (touching the ground) is at the computed position coordinates. We then compute the new centroid of the mesh in the 3D space. We perform a uniform scaling transformation at the centroid, where the scale is computed based on the ratio of the extent of X-coordinates of the road in the image space to that of X-coordinates of the road in the 3D space. This scale value needs to be checked manually and perturbed in some instances to obtain *visually accurate relative-ness* of the sizes of the objects in the scene. The automation of the computation of the scale is currently out of the scope of this paper.

Before the final rendering of the scene, we use an AABB collision detector to ensure that no models intersect with each other along the X and Z axes, and accordingly perturb the models. This serves as an additional corrector for predicted depth and horizontal translation. The 3D position coordinates and semantic label of each model in the final scene are written into a text file for future use or loading by a similar, or even advanced, rendering engine.

Step 4: Editing the Virtual World: This 3D scene can be optionally interactively edited to add, delete, or replace mesh models, before or after rendering the scene. In step 4, scene parameters from step 3 can be passed as input to modeling tools and rendering engines, such as Unity or Unreal.

We introduce the notion of “permissible” edits, as the destination mesh model must fit into the space selected for its placement. The fitting of the mesh model is done after it is scaled to a size comparable to the neighborhood of the new position and without distorting its aspect ratio. In the case of replacing a selected mesh model, one can also now place conditions on the semantic class of the object

that can replace the object type of the selected model. Thus, the permissible edits also include changing the type of object or position of an existing object.

This editing process in the workflow allows an artist to create a variety of scenes using the same skeleton—a feature that can be particularly useful in autonomous vehicle simulation. Edits can be made to the output of step 3, and the edited scene can be directly fed into external modeling and rendering engines. While we have a rudimentary graphical user interface (GUI) to edit the scene interactively, this can be enhanced in the future scope of work.

Rendering the Virtual World: In the current implementation, we render the 3D virtual world using OpenGL [22] graphics system to visualize and demonstrate it. We render the virtual world without any textures, using a perspective projection in OpenGL, ambient lighting, a Phong shader, and a shadow map. The rendered result in the orientation corresponding to the source monocular image may look slightly different from the source, owing to the known differences between the orientation of the camera sensor in the vehicle and the OpenGL settings, *i.e.*, the camera as well as the perspective projection. In Figure 3, we observe that the object models have been loaded at approximately correct relative positions, depth, and scale, albeit with minor visually perceptible differences.

4 Experiments, Results and Discussion

We have tested our workflow on images of straight roads from Cityscapes that contain objects of our interest, which are vehicles, trees, sidewalks, people, and buildings. Our current implementation is limited to straight roads without breaks and curves. We have demonstrated our best results of rendering 3D virtual worlds synthesized from four different monocular images using our workflow, namely, S1, S2, S3, and S4 (Figure 3). These images are also picked for the variety of residential and non-residential areas with straight roads, buildings, trees, vehicles, and sidewalks in the scenes. We have further validated the perceptual similarity of our results of 3D worlds with the corresponding source images. To more comprehensively evaluate the accuracy of our results in terms of inferring the right 3D positions and overall scene layout, we have conducted two user studies. The questionnaire forms for the user studies are available at <https://forms.gle/Zecjbi9UKXTC439Q6> and <https://forms.gle/f9LmpzYg3LNjazPu8>. We do not use any special methodology to ensure that the samples are representative of all possible traffic scenes. However, given that the results of each study corroborate the accuracy of the generated world we believe it is a useful evaluation.

User Study of Video-to-image Correspondences (T1): We have generated a video of fly-through of two 3D worlds synthesized using our proposed workflow that starts from the point of capture of the scene and moves forward into the scene before providing an aerial view. Thus, this video has a blend of overhead aerial view and rotation about the Y-axis of the 3D world space for developing the animation using OpenGL rendering. This user study entails view-



Fig. 3. 2D input image and corresponding view from the camera of the 3D virtual world, built using our proposed workflow. From top left clockwise: Scene S1, S2, S3, and S4. All four scenes are used for our image-to-image similarity user study; and S2 and S3, for our video-to-image correspondences study.

ing the video along with four potential RGB images from which the scene could have been generated, with a task to identify the correct source image.

We have generated a smooth animation of the 3D virtual world, for the best user experience. In the animation, we have additionally included the “source” vehicle where the camera is mounted as an object in the scene, rendered in red. This is to provide the realism and completeness of the virtual world, wherein when the camera is moving outside of the “source” vehicle for the fly-through, the “source” vehicle too is a part of the synthesized virtual world (Figure 4).

For each question in T1, we include images that are similar to the source image. We identify such images from the image collection using the structural similarity index (SSIM). The three images with ~ 0.5 SSIM are selected, and including the source image, we provide four choices for each question (Figure 4). This is to ensure that the choices available are similar in semantic content, but yet differentiable to perceive the correct image-to-3D world correspondence.

User Study of Image-to-image Similarity (T2): We have provided the survey respondents with a source RGB image and its corresponding 3D virtual world construction from the view of the camera sensor in the vehicle, rendered using OpenGL. This user study entails assessing the image-to-image similarity on a five-point Likert scale. Having choices 1 and 5 correspond to the worst and best matches, respectively, the Likert scale captures how close the virtual world perceptually matched with the source image, with respect to the presence and relative positions of the objects of interest, *e.g.*, cars, people, sidewalk (Figure 5).

Outcomes of User Studies: We present our results from a total of 41 survey respondents to both tests. The respondents participated voluntarily in social media groups. In all cases, T1 was administered before T2, in order to provide an appropriate context to the participants. We have used four test monocular images taken from moving vehicles, S1, S2, S3, and S4 (Figure 3).

For T1, the images chosen for S2 in the question have an SSIM of 0.58, 0.50, and 0.48 with the source image. Similarly, for S3, the selected images have an

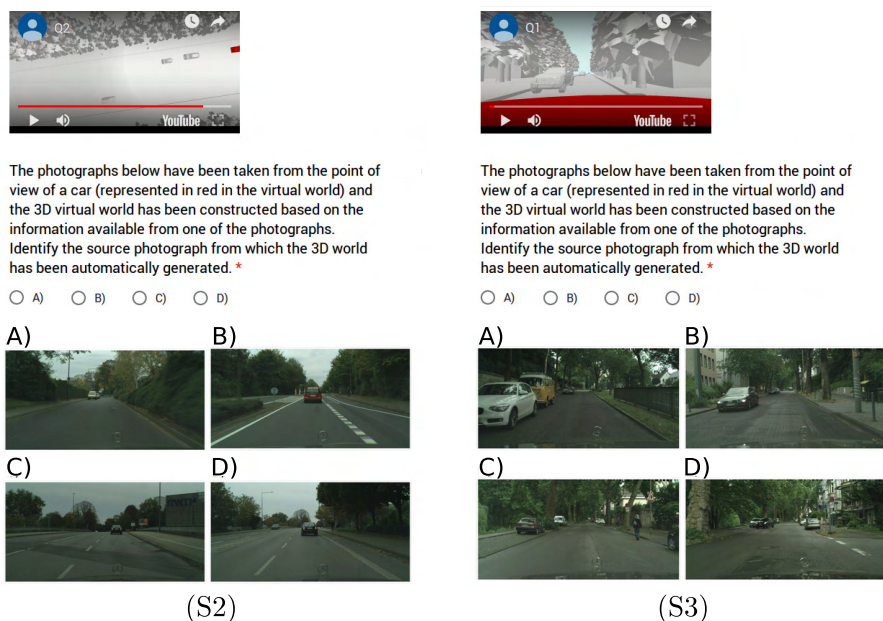



Fig. 4. Questionnaire for the user study of video-to-image correspondences (T1).

SSIM of 0.39, 0.43, and 0.39 with the source image. 90.2% of the users correctly identified the source image for S2, and 95.1% of the users correctly identified the source of the second question S3. Participants scored an average of 1.85 points in the study with a standard deviation of 0.48. Out of the 41 participants, 37 respondents got both correct, 2 got one correct, and 2 got both wrong. For the 2 respondents who got one correct answer, they both got the answers correct for S3. The wrong answers for S2 included all the other 3 choices and the wrong answers for S3 included 2 other choices. We observe that the question on S2 was perceptually more difficult than that on S3, which is validated by a relatively higher SSIM index, where the options for S2 are structurally more similar to the source image than that of S3.

For T2, S1, S2, S3, and S4 (Figure 3) were rated an average of 4.17, 4.24, 4.00, and 3.49 respectively, on the five-point Likert scale. Cronbach's alpha is used for Likert-scale responses to measure latent variables, such as openness and reliability of responses. The alpha value is indicative of whether a designed test accurately measures the variable of interest, which is the perceived similarity between the source image and the synthesized 3D virtual world. We computed a Cronbach's alpha of 0.803 for the responses. By rule of thumb, an alpha of 0.8 is a good goal [10]. We can thus conclude that our survey with ranking questions has been reliable.


(S2)



On a scale of 1 to 5, where 1 is the least match and 5 is the best match, how close does the image on the right match the image on the left in 1? Evaluate on the basis of whether the main scene objects (cars, people, sidewalk) are present and their positions seem to match.*

1 2 3 4 5


(S4)



On a scale of 1 to 5, where 1 is the least match and 5 is the best match, how close does the image on the right match the image on the left in 3? Evaluate on the basis of whether the main scene objects (cars, people, sidewalk) are present and their positions seem to match.*

1 2 3 4 5


(S3)



On a scale of 1 to 5, where 1 is the least match and 5 is the best match, how close does the image on the right match the image on the left in 2? Evaluate on the basis of whether the main scene objects (cars, people, sidewalk) are present and their positions seem to match.*

1 2 3 4 5

(S1)



On a scale of 1 to 5, where 1 is the least match and 5 is the best match, how close does the image on the right match the image on the left in 4? Evaluate on the basis of whether the main scene objects (cars, people, sidewalk) are present and their positions seem to match.*

1 2 3 4 5

Fig. 5. Questionnaire for the user study of image-to-image similarity (T2).

Our workflow implementation is a significant improvement over the manually curated scene construction. It takes ~ 4 -7 seconds to generate the final rendered output for one image. The construction time can be sped up by increased parallelization, particularly during averaging of values within segments. Generation of the depth itself takes to the order of 35 milliseconds for a 512×256 image on a modern GPU [11]. Manual input is required only at the time of perspective inversion where a user selects 5-6 points on the image, and as required, for correcting the relative scales of mesh models of objects of interest. By design, human-in-the-loop is reintroduced in the workflow for editing the scene. We have used untextured models in our user study to focus on relative object placements.

5 Conclusions

We have proposed a workflow to automatically build 3D virtual worlds based on a reference monocular image with minimal manual intervention to reduce the time and manual effort that goes into scene modeling. The workflow we have proposed can be operated to generate 3D worlds of urban traffic scenes which can be used as simulation environments for autonomous vehicle testing. Given the vast collection of reference images on the Web, our proposed workflow builds approximate 3D virtual worlds from these images efficiently creating a scene that

can be detailed and animated. Two key ideas that drive our workflow are the use of deep learning to estimate depth and perspective correction to get real-world positions from an image.

Our proposed workflow can be improved by adding a pose estimation module to automatically detect the orientation of every model and instance-specific labels to load more relevant mesh models. Our prototype implementation works for urban traffic scenes of straight roads, and our results show that our solution presents a useful direction for future research. An important aspect of the rendering of such a world is lighting, which implies determining the lighting parameters from the monocular images is an open problem.

ACKNOWLEDGMENTS

The authors are grateful to all members of the Graphics-Visualization-Computing Lab and peers at the IITB for their support. This work has been financially supported by the Machine Intelligence and Robotics (MINRO) grant by the Government of Karnataka.

References

1. Ahlberg, S., Söderman, U., Elmqvist, M., Persson, A.: On modelling and visualisation of high resolution virtual environments using LIDAR data. In: Proceedings of the 12th International Conference on Geoinformatics. pp. 299–306 (2004)
2. Beardsley, P., Torr, P., Zisserman, A.: 3D model acquisition from extended image sequences. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 683–695. Springer (1996)
3. Bullinger, S.: Image-based 3D Reconstruction of Dynamic Objects Using Instance-aware Multibody Structure from Motion, vol. 44. KIT Scientific Publishing (2020)
4. Cabon, Y., Murray, N., Humenberger, M.: Virtual KITTI 2. arXiv preprint arXiv:2001.10773 (2020)
5. Clarke, M.P.: Virtual World Construction (June 28 2016), US Patent 9,378,296
6. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The Cityscapes Dataset for Semantic Urban Scene Understanding. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3213–3223. IEEE (2016)
7. Frahm, J.M., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.H., Dunn, E., Clipp, B., Lazebnik, S., et al.: Building Rome on a Cloudless Day. In: Proceedings of European Conference on Computer Vision (ECCV). pp. 368–381. Springer (2010)
8. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual worlds as proxy for multi-object tracking analysis. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4340–4349. IEEE (2016)
9. 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)
10. Gliem, J.A., Gliem, R.R.: Calculating, Interpreting, and Reporting Cronbach’s Alpha Reliability Coefficient for Likert-type Scales. Midwest Research-to-Practice Conference in Adult, Continuing, and Community (2003)

11. Godard, C., Mac Aodha, O., Brostow, G.J.: Unsupervised Monocular Depth Estimation with Left-right Consistency. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 270–279. IEEE (2017)
12. Groenendijk, R., Karaoglu, S., Gevers, T., Mensink, T.: On the benefit of adversarial training for monocular depth estimation. *Computer Vision and Image Understanding* **190**, 102848 (2020)
13. Hane, C., Zach, C., Cohen, A., Angst, R., Pollefeys, M.: Joint 3D scene reconstruction and class segmentation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 97–104. IEEE (2013)
14. He, L., Lu, J., Wang, G., Song, S., Zhou, J.: SOSD-Net: Joint semantic object segmentation and depth estimation from monocular images. *Neurocomputing* **440**, 251–263 (2021)
15. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image Translation with Conditional Adversarial Networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1125–1134. IEEE (2017)
16. Kundu, A., Li, Y., Dellaert, F., Li, F., Rehg, J.M.: Joint semantic segmentation and 3D reconstruction from monocular video. In: Proceedings of European Conference on Computer Vision (ECCV). pp. 703–718. Springer (2014)
17. Lempitsky, V., Boykov, Y.: Global optimization for shape fitting. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–8. IEEE (2007)
18. Mohr, R., Quan, L., Veillon, F.: Relative 3D reconstruction using multiple uncalibrated images. *The International Journal of Robotics Research* **14**(6), 619–632 (1995)
19. Nidhi, K., Paddock, S.M.: Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice* **94**, 182–193 (2016)
20. Pollefeys, M., Nistér, D., Frahm, J.M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.J., Merrell, P., et al.: Detailed real-time urban 3D reconstruction from video. *International Journal of Computer Vision* **78**(2-3), 143–167 (2008)
21. Rander, P., Narayanan, P., Kanade, T.: Virtualized reality: Constructing time-varying virtual worlds from real world events. In: Proceedings of 8th IEEE Visualization Conference. pp. 277–284. IEEE (1997)
22. Shreiner, D.: OpenGL reference manual: The official reference document to OpenGL, version 1.2. Addison-Wesley Longman Publishing Co., Inc. (1999)
23. Snavely, N., Seitz, S.M., Szeliski, R.: Photo Tourism: Exploring Photo Collections in 3D. In: *ACM Transactions on Graphics (TOG)*. vol. 25, pp. 835–846. ACM (2006)
24. Venkateshkumar, S.K., Sridhar, M., Ott, P.: Latent Hierarchical Part Based Models for Road Scene Understanding. In: Proceedings of 2015 IEEE International Conference on Computer Vision (ICCV) Workshop. pp. 115–123. IEEE (2015)
25. Wang, T.C., Liu, M.Y., Zhu, J.Y., Liu, G., Tao, A., Kautz, J., Catanzaro, B.: Video-to-video Synthesis. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS). pp. 1152–1164 (2018), <http://dl.acm.org/citation.cfm?id=3326943.3327049>