An Automatic 3D Scene Generation Pipeline Based on a Single 2D Image

Alberto Cannavò[1(✉)], Christian Bardella[1], Lorenzo Semeraro[1], Federico De Lorenzis[1], Congyi Zhang[2], Ying Jiang[2], and Fabrizio Lamberti[1]

[1] Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, Italy
{alberto.cannavo,christian.bardella,lorenzo.semeraro,federico.delorenzis, fabrizio.lamberti}@polito.it
[2] The University of Hong Kong, Pokfulam Road, Hong Kong, China
cyzhang@cs.hku.hk, yingjiang@connect.hku.hk

Abstract. In the last years, solutions were proposed in the literature to alleviate the complexity of using sophisticated graphic suites for 3D scene generation by leveraging automatic tools. The most common approach based on the processing of text descriptions, however, may not represent the ideal solution, e.g., for fast prototyping purposes. This paper proposes an alternative methodology able to extract information about the objects and the layout of the scene to be created from a single 2D image. Compared to previous works, experimental results reported in this work show improvements in terms of similarity between the 2D and 3D scenes.

Keywords: Image-based modelling · Scene-object modelling · Machine learning

1 Introduction

Today, the commonly used software to generate graphics assets are suites, such as Blender[1], and Autodesk Maya[2], since they provide a number of functionalities and tools that allow users to produce 3D animated scenes [10]. Although these suites are generally characterized by a higher degree of flexibility in terms of available features, they also present a very steep learning curve [8]. Difficulties in learning such software prevent unskilled users to operate with them or, more in general, make the generation of 3D contents a time-consuming task requiring a lot of effort even for professional users [2].

One of the most common operation being performed with such suites is represented by the objects' layout. Although the commands to move and rotate objects are generally simple to activate, the standard input interfaces to control

This work has been supported by VR@POLITO initiative.

[1] Blender: https://www.blender.org/.

[2] Autodesk Maya: https://www.autodesk.com/products/maya/overview.

L. T. De Paolis et al. (Eds.): AVR 2021, LNCS 12980, pp. 109–117, 2021.
https://doi.org/10.1007/978-3-030-87595-4_9

the transformations are based on mouse and keyboard. Therefore, users can handle only two degrees of freedom at a time, resulting in an increase of the users' mental effort [3]. As a consequence, the scenes generated by unskilled users are generally unrealistically simple [14]. However, operating with such software is becoming inescapable for non-professional users [5], e.g., to share a prototypical idea of 3D scenes that can be later used by experts to build the ultimate version.

In order to overcome these limitations, the literature proposed a number of solutions able to automatically reconstruct 3D scenes by processing, e.g., text [4], images [1] and audio clips [12]. The high number of works based on text descriptions (e.g. [4,5]) suggests that this approach represents the most common solution. However, in the case of fast prototyping, it may not be the ideal solution, since the need to write the entire description of the scenes can slow down the process [1]. Leveraging a simple 2D image (e.g. a sketch or a photo) of the desired scene could be significantly faster than writing a text description. Image-based approaches introduce a number of new challenges in the understanding of the input regarding, for example, the need to infer/detect objects to model them individually, solve the depth ambiguity to correctly place objects, recognize relationships among objects, etc. To cope with the above issues, this paper proposes a methodology that breaks down the complexity of the overall process into a number of simpler steps. The main idea is to leverage information regarding the camera framing the scene (e.g. point of view, perspectives cues, relative positions of the objects in the view, etc.) to generate a similar layout in the reconstructed environment. Machine learning algorithms complement this methodology to infer the number and types of objects as well as determining the distance from the camera.

Moreover, differently from solutions presented in the literature, which were usually implemented as standalone applications, the proposed pipeline is integrated within a traditional graphics suite. Besides the possibility to improve the quality of the generated scene by leveraging the sophisticated methods embedded in the software, the integration allows users to explore scenes into an immersive environment using virtual reality technologies, thus helping them to better understand the scene layout. The integration within a traditional software suite was made possible by developing a dedicated add-on that can be installed into the open-source 3D graphics software named Blender. The add-on is also released as open-source at https://github.com/logicesecutor/3D_scene_generator.

In order to assess the effectiveness of the proposed system it was compared with a previous work [1]. Results showed promising improvements in terms of scene similarity.

2 Related Works

Different solutions have been proposed to solve the problem of generating 3D graphics assets from 2D images. For example, in [13] perspective cues, such as perspective lines or distorted planes, identified within a single 2D image, are leveraged to reconstruct both indoor and outdoor environments as a combination of flat textured planes represent walls, floor, and ceiling. Another example

is reported in [6], which describes a number of tools developed for Matlab to reconstruct a 3D environment from multiple calibrated images. With respect to the previous work in [6] the shapes of the objects identified in the images are preserved. However, the resulting 3D scene is built as a single high-poly mesh. This approach, named photogrammetry, may be considered not the ideal solution for fast prototyping, since the high number of vertices in the resulting mesh could make it difficult to apply changes or further operate with it. In [11] a neural network was introduced for recognizing and reconstruct boxes and spheres identified into the 2D image provided as input to the network. Although this approach partially solves the issue regarding the shape of the objects, the poor flexibility related to the limited set of recognized objects could represent a weakness for general-purpose applications.

The works reviewed so far described systems and tools developed as standalone applications not integrated into existing 3D graphics software. However, in the context of fast prototyping, the integration with such software can allow expert developers to further edit on the generated scenes seamlessly without the need for additional import/export operations.

To the best of the authors' knowledge, a few works were reported in the literature that are able to generate 3D scenes from single 2D images that is already integrated into a graphics suite. An example is the add-on for Blender proposed in [1]. The environment automatically built with this tool can be explored and modified into an immersive environment. However, in this work, the positions of the objects in the source image and their relations were not considered in the generation of the scenes. The layout was generated automatically through a probabilistic method, that could introduce significant differences in the objects positioning.

Moving from the above review, a system is proposed able to automatically generate 3D scene from its 2D representation. The idea was to leverage information extracted from the source image not only to infer the number and types of objects but also to understand their positions and relations in the environment. The tool is integrated within a well-known suite for enabling further editing.

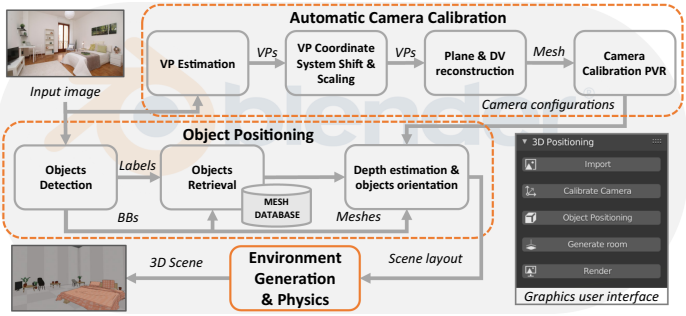


Fig. 1. System architecture.

3 System Overview

The overall architecture of the system is illustrated in Fig. 1. The expected workflow starts with the selection of the source image used as input to the following steps. By analyzing this image, a number of data are inferred, i.e., information of the camera framing the environment represented in the 2D image and the objects identified in the scene. These data are then combined in order to reconstruct the 3D environment (more details on each step will be provided in the remaining part of this section).

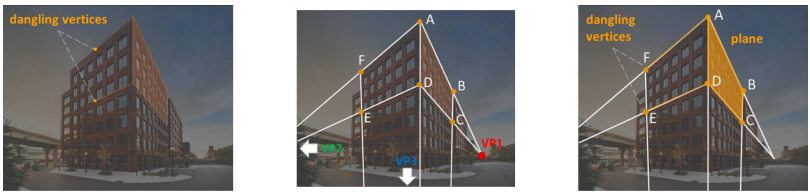
The entire workflow is integrated within the well-known 3D computer graphics software (Blender). After the installation of the proposed add-on, a graphical user interface is shown in Blender's 3D Viewport Editor to let users select images and start the elaboration as shown in Fig. 1. Once the 3D scene has been generated, the user can apply further changes by using the functionalities of Blender and explore the scene into an immersive environment by means of Blender's native add-on that enables the VR visualization.

3.1 Automatic Camera Calibration

The main objective of this step is to calibrate the Blender's camera to match as much as possible the external and internal parameters of the camera framing the scene represented in the 2D image. In particular, the main idea behind the camera calibration is to look at the scene with the same camera setting used to obtain the source image in order to extract the relative positions of the objects in the scene and their relationship. To this aim, the add-on Camera Calibration PVR[3] was used. The add-on matches the Blender's 3D Camera to the perspective observed in the source image. As requirements, the user has to manually shape the mesh of a plane to match the perspective lines and add two dangling vertices corresponding to perpendicular lines observed in the image. An example of a valid input is shown in Fig. 2a. In order to automate this process, the Camera Calibration PVR add-on was integrated within the proposed system and its functionalities were combined with a further library, i.e., XiaohuLuVPDetection[4], that implements the algorithm for the automatic vanishing points estimation proposed in [9]. In particular, with the XiaohuLuVPDetection library, the three vanishing points in the source image are detected (Fig. 2b). Then the computed positions of the three points, expressed in pixel unit are converted in meters and expressed into the image space reference system. The coordinates of the intersection between lines (points A to F in Fig. 2b) are found by iteratively solving a linear system for each of the six vertices required (i.e., four vertices for constructing the plane and the two additional dangling vertices). Once the plane has been automatically reconstructed (Fig. 2c), the Camera Calibration PVR add-on is executed to calibrate the Blender's 3D camera. The operations

[3] Camera Calibration PVR: https://github.com/mrossini-ethz/camera-calibration-pvr.

[4] XiaohuLuVPDetection: https://github.com/rayryeng/XiaohuLuVPDetection.



(a) (b) (c)

Fig. 2. Automatic camera calibration: a) dangling vertices, b) vanishing points, c) reconstructed plane.

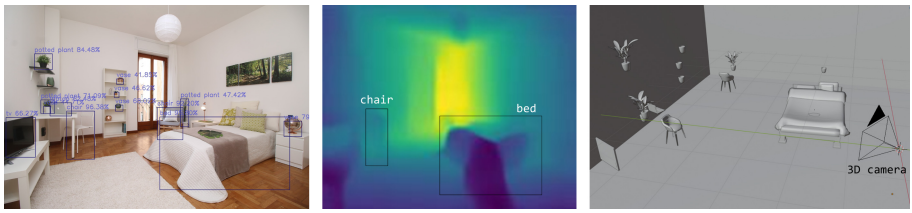
described above are totally transparent to the user, who is only asked to click on the "Calibrate Camera" button Fig. 1 to automatically execute the entire calibration process.

3.2 Object Positioning

By clicking on the "Object Positioning" button, the system automatically extracts objects from the source image and places them into the environment according to a number of rules detailed in the following. A Convolutional Neural Network (CNN) is used to extract from the source image the labels and the bounding boxes (BBs) of the objects detected in the environment. To this aim, the open-source python library Image AI[5] was considered, since it offers a number of reproducible machine learning algorithms for image prediction, object detection, video object tracking, etc. Object detection is supported using three models, i.e., RetinaNet, YOLOv3 and TinyYOLOv3, trained on COCO dataset[6]. The YOLOv3 model was selected for its performance in terms of computation time, however, in the future, more advanced or custom models could be leveraged to improve the quality of the recognition. Moreover, fine-tuning on a specialized custom dataset could be applied to reduce the error-rate and increase the number of labels extracted from the source image. The labels are leveraged to search corresponding 3D models in the mesh database. The mesh database contains 50 models, covering all the possible labels that can be extracted from the source image with the considered CNN. Besides the 3D geometry, the database also memorizes materials, textures, physical properties, and baked animations. A number of alternative models are also provided in the database to avoid repetitions in case of labels recognized multiple times. The BBs are leveraged by the Object Retrieval block to extract from the source image the RGB color picked at the center of the BB. The RGB values are used to edit the default shader of the 3D objects to make it assume a base color similar to that observed in the source image.

Once all the models are retrieved from the database, their positions are adjusted in the 3D environment to match the center of the BB detected in the

[5] Image AI: https://github.com/OlafenwaMoses/ImageAI.
[6] COCO dataset: https://cocodataset.org/#home.



(a) (b) (c)

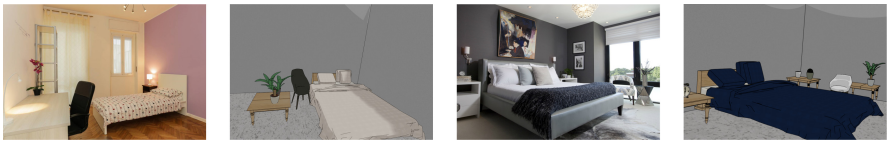
Fig. 3. Objects' depth estimation: a) source image with recognized objects and BBs, b) depth maps with two sample BBs mapped in.

source image with that observed by looking at the object from the calibrated 3D camera. To compute the BB of an object observed by the 3D camera, all the 3D coordinated of the vertices are mapped into a 2D plane and then converted to pixel coordinates in the camera viewport. This operation can take time for high poly meshes. For this reason, the geometry of the mesh is first simplified by using a Blender's modifier that automatically reduces the number of vertices.

From a perspective point of view, bringing an object closer to the camera or scaling it, makes no difference in the projected image. Moreover, the coordinates of the BBs retrieved by the object detection components are 2D and do not contain indications about the depth. To solve this issue, a principal assumption was made, i.e., the proportions among the objects are fixed (according to the sizes defined in the database) and coherent among the objects. Under these hypotheses, it is possible to use a depth prediction algorithm to extrapolate the depth value of the objects in the image. To this aim, the CNN models trained for depth prediction from a single RGB image proposed in [7] and available at https://github.com/iro-cp/FCRN-DepthPrediction are leveraged. Depth prediction network predicts depth map for the source image as shown in Fig. 3b. Pixel coordinates of the BBs detected in the source image are then mapped to the depth map and the depth value at the center of the BB is used to set the distance of the object from the 3D camera. The result of this process is depicted in Fig. 3c. Finally, the orientation of each object is adjusted with a successive approximation method with the goal of minimizing the difference between the aspect ratio of the BB detected in the source image and that observed in the scene from the 3D camera.

3.3 Environment Generation and Physics

This step, executed by clicking on the "Generate room" button, generates a plane with grass in case of outdoor settings or a complete room with walls, floor and ceiling for indoor settings. Then, physics constraints are applied to the scene, i.e., gravity force, to avoid floating elements and correctly place objects on the surface below. Examples of 3D scenes generated by the proposed system are shown in Fig. 4.



(a) (b) (c) (d)

Fig. 4. Considered use cases: a) source image n.1 (source: https://www.aluna.it), b) result obtained from image n.1, c) source image n.2 (source: https://www.wattpad.com), and d) result obtained from image n.2.

4 Experimental Setup

With the aim to assess the performance of the proposed system a user study was conducted by involving 31 volunteers (24 males and 7 females), aged between 22 and 55 ($\mu = 31.45$ and $\sigma = 10.55$). Participants can be considered as non-professional users, since only a limited number of them had experience with graphics suites. Participants' background was evaluated through a demographics questionnaire filled in before the experiment. Collected data revealed the following statistics on the usage of graphics suite: never: 51.6%, sometimes: 25.8%, once a week: 9.7%, everyday: 12.9%. To evaluate the proposed system, both objective and subjective measurements have been collected for generating three representative indoors settings. The objective measurements consider the completion time needed to infer the objects from the image (inference time), the time required to create the scene layout (position time), and the number of objects detected in the image. The subjective metrics have been used to compare the proposed system with the automatic scene generation tool proposed in a previous work [1]. The subjective observations were based on a questionnaire aimed at evaluating the scene similarity. In particular, participants were requested to judge the similarity of the generated scenes in terms of number and types of objects, overall scene layout, as well as applied materials and textures. A within-subject approach was used in this study to compare the two tools. The questionnaire is available at https://bit.ly/3akDFbk.

4.1 Results

The results in terms of inference and position time, as well as the number of detected objects, are reported in Table 1

It is worth noticing that the most time-consuming task is represented by the inference time, which on average covers the 76.14% of the overall process. The flexibility of the proposed architecture makes it possible, in the future, to replace the block in charge of inferring objects in order to speed up this process and increase the number of detectable objects.

Average scores regarding scene similarity are reported in Fig. 5. Paired samples t-test with 5% significance ($p < 0.05$) was used to analyze statistically significant differences. Participants found that the proposed system was able to

Table 1. Objective results concerning: inference time, positioning time and number of objects detected.

Use case	Inf. time	Pos. time	Detected obj
1	22.32 s	4.35 s	4
2	25.20 s	12.59 s	13
3	26.20 s	7.37 s	6

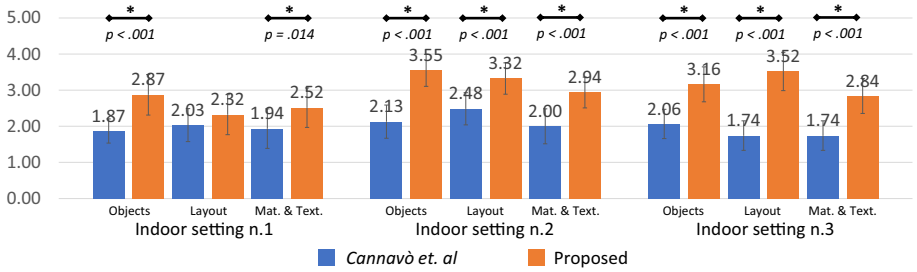


Fig. 5. Subjective results concerning scene similarity: average scores (bars height) and standard deviation (errors bars). Statistically significant differences ($p < 0.05$) are marked with the '*' symbol.

generate a better reconstruction of the source image compared with the previous work in the majority of the considered use cases, as confirmed by the higher values obtained in all the considered metrics (except for the first use case, in which no significant differences were found for the layouts).

5 Conclusions and Future Work

This paper presented a system that allows non-professional users to quickly create 3D scenes for fast prototyping. A methodology was introduced that combines the advantages of automatic 3D scene generation with traditional graphics suites through the development of an add-on target to a well-known graphics suite. The system proposed in this paper was not meant to replace the traditional graphics suites, but rather to support non-professional users in the quick prototyping of 3D scenes by leveraging a single 2D image as input. The scene generated automatically by the proposed system can be considered as a draft that expert developers can take as reference or starting point to obtain the ultimate version. A user study performed by involving participants with limited skills in using graphics suites. Results showed promising improvements in terms of scene similarity with respect to a previous work.

Currently, the system presents limitations in terms of flexibility, i.e., possible scenes that can be created, due to the limited number of objects in the database and labels that can be inferred by the selected library. Besides solving the above limitations, which can be faced with the development of new models

and the introduction of alternative tools for inferring objects from images, possible evolution will consider the introduction of techniques, to further improve the capacity of the system of preserving the relationship among objects.

References

1. Cannavò, A., D'Alessandro, A., Maglione, D., Marullo, G., Zhang, C., Lamberti, F.: Automatic generation of affective 3D virtual environments from 2D images. In: Proceedings of the International Conference on Computer Graphics Theory and Applications, pp. 113–124 (2020)
2. Cannavò, A., Demartini, C., Morra, L., Lamberti, F.: Immersive virtual reality-based interfaces for character animation. IEEE Access **7**, 125463–125480 (2019)
3. Cannavò, A., Lamberti, F.: A virtual character posing system based on reconfigurable tangible user interfaces and immersive virtual reality. In: Proceedings of the Smart Tools and Applications for Graphics - Eurographics Italian Chapter Conference, pp. 1–11 (2018)
4. Chang, A., Savva, M., Manning, C.: Interactive learning of spatial knowledge for text to 3D scene generation. In: Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces, pp. 14–21 (2014)
5. Chang, A.X., Eric, M., Savva, M., Manning, C.D.: SceneSeer: 3D scene design with natural language. arXiv preprint arXiv:1703.00050 (2017)
6. Esteban, I., Dijk, J., Groen, F.C.: From images to 3D models made easy. In: Proceedings of the 19th ACM International Conference on Multimedia, pp. 695–698 (2011)
7. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: 4th International Conference on 3D Vision, pp. 239–248 (2016)
8. Lu, J., Li, C., Yin, C., Ma, L.: A new framework for automatic 3d scene construction from text description. In: Proceedings of IEEE International Conference on Progress in Informatics and Computing, vol. 2, pp. 964–968 (2010)
9. Lu, X., Yaoy, J., Li, H., Liu, Y., Zhang, X.: 2-line exhaustive searching for real-time vanishing point estimation in Manhattan world. In: Proceedings of the IEEE Winter Conference on Applications of Computer Vision, pp. 345–353 (2017)
10. Oliveira, B., Azulay, D., Carvalho, P.: GVRf and blender: a path for android apps and games development. In: Proceedings of the International Conference on Human-Computer Interaction, pp. 329–337 (2019)
11. Payne, B.R., Lay, J.F., Hitz, M.A.: Automatic 3D object reconstruction from a single image. In: Proceedings of the ACM Southeast Regional Conference, p. 31 (2014)
12. Sra, M., Maes, P., Vijayaraghavan, P., Roy, D.: Auris: creating affective virtual spaces from music. In: Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology, p. 26 (2017)
13. Vouzounaras, G., Daras, P., Strintzis, M.G.: Automatic generation of 3D outdoor and indoor building scenes from a single image. Multimedia Tools Appl. **70**(1), 361–378 (2011). https://doi.org/10.1007/s11042-011-0823-0
14. Xu, K., Stewart, J., Fiume, E.: Constraint-based automatic placement for scene composition. Proc. Graph. Interface **2**, 25–34 (2002)