Automatic generation of affective 3D virtual environments from 2D images

Automatic Generation of Affective 3D Virtual Environments from 2D Images

Alberto Cannavò[1][a], Arianna D'Alessandro[1], Daniele Maglione[1], Giorgia Marullo[1], Congyi Zhang[2][b] and Fabrizio Lamberti[1][c]

[1]*Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, Italy*

[2]*Department of Computer Science, The University of Hong Kong, Chow Yei Ching Bldg, Pokfulam Road, Hong Kong*

{*f_author, s_author*}*@polito.it, cyzh@hku.hk*

Keywords: Virtual Reality, Image-based Modeling, Scene and Object Modeling, Human-Computer Interaction.

Abstract: Today, a wide range of domains encompassing, e.g., movie and video game production, virtual reality simulations, augmented reality applications, make a massive use of 3D computer generated assets. Although many graphics suites already offer a large set of tools and functionalities to manage the creation of such contents, they are usually characterized by a steep learning curve. This aspect could make it difficult for non-expert users to create 3D scenes for, e.g., sharing their ideas or for prototyping purposes. This paper presents a computer-based system that is able to generate a possible reconstruction of a 3D scene depicted in a 2D image, by inferring objects, materials, textures, lights, and camera required for rendering. The integration of the proposed system into a well-known graphics suite enables further refinements of the generated scene using traditional techniques. Moreover, the system allows the users to explore the scene into an immersive virtual environment for better understanding the current objects' layout, and provides the possibility to convey emotions through specific aspects of the generated scene. The paper also reports the results of a user study that was carried out to evaluate the usability of the proposed system from different perspectives.

1 INTRODUCTION

The use of computer-generated graphics assets is becoming very common in various application domains, ranging from movie and video game production (Schmalstieg and Stork, 2019; Fascione et al., 2018) to virtual reality simulations (Zyda, 2005), augmented reality applications (Ates et al., 2015), etc. Several software suites, like Blender[1], Autodesk Maya[2], and 3ds Max[3] have become the mainstream solutions chosen by experts (modellers and animators) to design and build 3D scenes, since they offer a complete set of tools for managing 3D graphics (Khatri, 2018; Seidler, 2018). However, the high flexibility of these suites and the large number of available functionalities are paid with a very steep learning curve (Lu et al., 2010), which make them not easy to use especially for novice users (Cannavò et al., 2019). Among numerous operations performed with such software, one of the most common is the setup of the objects' layout in the scene. Although this operation requires to manipulate for each object six degrees of freedom (3D position and orientation), standard input devices (mouse and keyboard) can control only two degrees of freedom at a time (Cannavò and Lamberti, 2018), thus increasing the mental effort of the user. The limited dimensionality of these devices does not affect only the input, but also the output. In fact, in order to have a clear idea of the scene setup, users are requested to simultaneously look at multiple views of the scene or use shortcuts to quickly change the current point of view. As a result, the scenes generated by users with limited skills are often unrealistically simple (Xu et al., 2002), and/or require a lot of time to be created. The high number of skills required and the difficulty in using traditional software suites could prevent non-professional users from quickly creating a 3D scene, although this operation can be fundamental also for them (Chang et al., 2017), e.g., to share an idea with more expert users or to create a simple pro-

[a] https://orcid.org/0000-0002-6884-9268

[b] https://orcid.org/0000-0002-4259-2863

[c] https://orcid.org/0000-0001-7703-1372

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

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

[3]3ds Max: https://www.autodesk.com/products/3ds-max/overview

totype of the scene to be later refined by using more accurate methods.

Considering the user input, the research community is devoting more and more attention to new approaches for generating contents automatically by processing, e.g., text (Chang et al., 2014), images (Vouzounaras et al., 2014) and audio clips (Sra et al., 2017). Systems capable to generate not only the layout, but also a convincing visual representation of the scene are getting more and more of interest because of their applicability in real scenarios. Among the different alternatives, one of the most common solutions consists in creating 3D scenes from a text description. Although this possibility has been widely addressed in a number of works (Coyne and Sproat, 2001; Seversky and Yin, 2006; Chang et al., 2014; Chang et al., 2017), it may not represent the best solution for fast prototyping. In fact, using a picture of the scene to be reconstructed could be faster than writing a description of it. For these reasons , we specifically decided to investigate the use of 2D images as input to the step in which contents to be inserted in the scene are defined. Another disadvantage of already existing solutions is the limited integration with traditional graphics suites. Current systems are commonly implemented as standalone applications. For this reason, they still require additional operations (for example the importing/exporting of the generated assets) to enable further modifications of the resulting scenes through more sophisticated methods embedded in professional software.

With respect to system output, the attention of the research community is focused in particular on the new possibilities offered by Virtual Reality (VR) technology. A number of works already demonstrated the capacity of VR to enhance the users' spatial awareness of the environment for generating 3D animations (Vogel et al., 2018; Cannavò and Lamberti, 2018). By exploring the automatically generated scene in an immersive virtual environment, the users could better understand the current objects' layout. This aspect may help non-professional users to, e.g., provide expert developers with more accurate hints on how the scene could be improved. Moreover, the use of VR enables more intuitive techniques based on 3D interactions that may be exploited by non-professional users to apply changes to the layout (Cannavò et al., 2019).

Finally, when the generation of 3D scenes is specifically targeted to the design of immersive experiences, one additional aspect that is often underestimated is the emotional relevance of the virtual environment, which could consistently affect the users' sense of presence (Hoorn et al., 2003). Different aspects (shapes, lights, materials, and textures) have proven to be capable to transfer measurable effects on the humans' mind (Crippa et al., 2012), and the interest in creating affective virtual environments is getting higher and higher, as confirmed by the growing number of works in the literature (Baños et al., 2004; Sra et al., 2017; Naz et al., 2017; Pallavicini et al., 2019).

Moving from these considerations, the present paper proposes a system targeted to non-professional users for the automatic generation of 3D scenes from a single 2D image. The main objective of this work is not to present a tool meant to replace traditional graphics suites, but rather to augment them by allowing users with limited skills to quickly create a first draft of a 3D scene. Non-professional users could take advantage of this system to present a possible setup of the environment to expert developers who may be in charge of applying further modifications to the scene. Once created, the scene can also be explored into an immersive environment, with the aim of getting insights about objects' layout that could be exploited by expert users to improve the final quality. Considering this goal, the proposed system was developed as an add-on of the well-known 3D graphics software named Blender. This integration allows more skilled users to directly manipulate the scenes created by the system, without the need for import/export operations. Using intuitive 3D interaction techniques, users can also modify the objects' layout within the immersive environment. The VR system adopted in this work is the HTC Vive kit. The add-on described in this paper is also released as open source (https://github.com/grainsgroup/AutomaticSceneGeneration).

Finally, this paper takes some steps toward the inclusion in the 3D scene generation process of aspects that characterize affective VR environments.

In order to evaluate the usability of the devised system, both objective and subjective measurements were collected through a user study that was carried out by involving non-expert users. Results confirm the high usability of the system for generating 3D scenes which appear as similar to the input image. They also showed that the system is effective in conveying the intended emotions. Finally, they provided interesting insights on the suitability of VR for enhancing spatial awareness about resulting objects' layout.

2 RELATED WORKS

The problem of supporting the automatic generation of 3D scenes is not new. For instance, (Coyne and

Sproat, 2001) presents a system named WordsEye which allows the users to easily convert a text into a 3D scene by leveraging a database containing 3D models and poses. The workflow of the system encompasses parsing of the input text, semantic analysis, and identification of the low-level descriptors (3D objects, poses, spatial relations, color attributes, etc.) to be added/used to/in the scene. In (Seversky and Yin, 2006), a framework is presented which, given a text description or an audio clip, is able to generate a 3D scene by placing 3D objects retrieved from a database. To control object positioning, the devised algorithm assumes that spatial relations (in, on, under, above, in front of, etc), together with a number of possible modifiers (to the left/left of, towards), have been provided in the input text/audio. The algorithm was implemented in Java by leveraging the 3D graphics named API JView. More recently, in (Chang et al., 2014), another system has been presented which converts a text description into a 3D scene. First, objects to be placed in the 3D scene are extracted from the text. Afterwards, the system, differently than in the previous works, infers the most likely objects' layout based on observations made in previous spatial arrangements. Refinements made by the user to improve the final layout are considered by the system to improve its estimates. Another example is represented by the system named ScenSeer which was proposed in (Chang et al., 2017). The system extracts objects to be placed in the scene by parsing the text provided as input. Then, it exploits a spatial knowledge base (obtained by combining an existing database of 3D models and 3D scenes) to infer the objects' layout and identify possible additional objects to add even though not explicitly mentioned in the input text. The user has the possibility to add, remove, manipulate and replace objects in the scene by issuing other text commands.

In all the works reviewed above, the input provided to the system consists in textual descriptions. However, as said before, for fast prototyping purposes, describing a scene using text could be tedious and time-consuming. Under this hypothesis, this paper investigates the use of images as input to recreate a 3D environment. Regarding this topic, the literature contains several works that tried to faithfully recreate the environment depicted in an image. For example, the approach developed in (Vouzounaras et al., 2014) identifies perspective cues (like perspective lines or distorted planes) in a single 2D image with the aim to create a 3D reconstruction (composed by flat textured planes representing walls, floor, and ceiling) of both indoor and outdoor environments. The work in (Esteban et al., 2011) presents a set of tools built for Matlab to obtain a 3D reconstruction of an environment from a set of calibrated images. With respect to (Vouzounaras et al., 2014), the richness of the scene is improved, by considering also possible objects found in the environment. However, the reconstructed 3D scene is represented through a single mesh with a high number of vertices, making this approach (commonly referred to as photogrammetry) not suitable for fast prototyping. The focus on the object is posed also in the approach described in (Payne et al., 2014), where a neural network is exploited to recognize and reconstruct two classes of objects: boxes and spheres. The input of the network is a 2D image. The corresponding output is a 3D textured VRML, X3D or WebGL file representing the recognized object. The limited set of recognized objects represents a limitation for general-purpose applications.

It is worth observing that all these works developed standalone tools, making the integration in the existing 3D graphics suites difficult to achieve. The literature does not provide examples of systems that combine the advantages of automatic 3D scene generation with standard graphics suites integration, with the exception of the work in (Lu et al., 2010). The framework presented in the latest work is able to convert a text description into a 3D scene which then can be visualized in Autodesk Maya. A knowledge base is exploited to represent the common sense knowledge, i.e, common properties of an object, e.g., its name, canonical position, orientation in the world, etc. Combing information provided by the knowledge base and extracted by the analysis of the input text, it is possible to translate the text into a XML file describing the 3D scene which can then be loaded in the considered graphics suite.

Examples concerning the automatic generation of immersive environments are the works reported in (Sra et al., 2016; Sra et al., 2017). In the first work (Sra et al., 2016), the real world is used as a template for modeling the virtual environment. After the generation of a 3D map (combining depth and color images) representing the surrounding environment, walkable areas and obstacles are detected and replaced with a corresponding virtual counterpart in VR. The virtual counterpart is not the same object identified in the real environment, but rather it represents a different object which occupies the same volume. The virtual objects to be placed in the scene are automatically retrieved from a set of possible objects related to the context (for example scene settled on an island, in a volcano, in the space, etc.) chosen by the users for the virtual environment. In (Sra et al., 2017), a system called Auris is described which automatically generates a VR environment from music

(audio clips and lyrics). In particular, nouns extracted from the lyrics by using the Standford part-of-speech tagger are used to influence the design of the global scene, the objects to be added and their materials. The final output is a psychedelic and surreal environment that the users can explore. Moreover, in this work, emotional aspects have been considered in order to dynamically affect the lighting of the scene and the textures to be applied to objects. In order to identify the general mood to be conveyed by the scene, a neural network was trained to discriminate between two classes (happy and sad moods) by using the Mel-frequency Cepstral Coefficients (MFCC) features extracted from the audio clips.

Moving from the above review, we designed a system able to combine three features: the possibility to automatically generate a 3D scene from its 2D representation, the integration with a well-known graphics suite for editing purpose, and the possibility to use VR for exploring and manipulating the created scene. Moreover, this work tries to take a few steps towards the introduction of aspects that characterize affective environments into the automatic scene generation process.

3 SYSTEM OVERVIEW

This section presents the overall architecture of the system, which is illustrated in Figure 1. The standard workflow begins with the definition of the input, i.e., the source image and the mood that the scene has to convey, by leveraging the graphical user interface provided by the *Scene Creator add-on*. Afterward, the same tool combines the input data and the results of Google Cloud Vision APIs, which are exploited for context and objects extraction. Models to be inserted into the scene are retrieved from a *Models database*. Once created, the 3D scene can be manipulated in a well-known 3D computer graphics software (Blender). Finally, the user can explore the generated 3D scene into an immersive virtual environment and apply changes to the objects' positions, orientation and scale by means of another add-on (the *Virtual Reality add-on*).

3.1 Input

The first step for the generation of 3D scenes is represented by the input definition. The user has to specify the parameters which define the emotion to be conveyed by the created scene and the source image used to infer its contents. The mood can be defined by exploiting the Scene Creator add-on. In particular, this

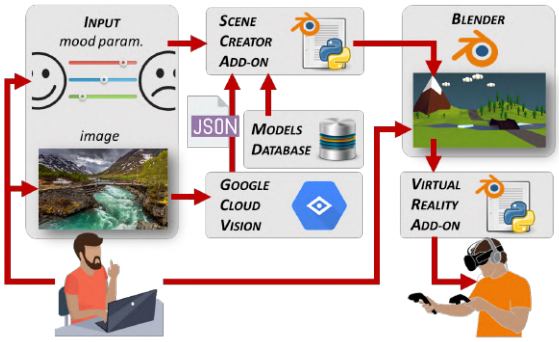


Figure 1: System architecture.

Figure 2: Graphics user interface of the Scene Creator add-on: the Automatic scene generation panel.

add-on is constituted by two parts: the back-end and the front-end. The back-end implements the logic for the scene generation (more details will be provided in Section 3.4). The front-end provides the user with a panel named *Automatic scene generation*, shown in Figure 2.The panel is automatically added to the Blender's Tool shelf, i.e, the set of panels on the left of the Blender's 3D View editor, after the installation of the Scene Creator add-on.

The Automatic scene generation panel includes the following controls:

- *Mood Controls*: a combo-box that allows the user to choose the emotion to be conveyed. Currently, the system supports two opposite emotions: happiness and sadness.
- *Input File*: a text-box for specifying which is the file to be used as source image.
- *Additional Objects*: a text-box for specifying the number of additional objects to be inserted in the 3D scene in addition to those recognized in the source image by the Google Cloud Vision APIs. More details about the additional objects placed into the scene will be provided in Section 3.4.
- *Run Button*: a button that allows the user to start the generation of the 3D scene by activating the functionalities of the back-end.

3.2 Google Cloud Vision

As said before, the source image is used by the system to extract the context of the scene and the objects to be

inserted. In order to obtain this information, the user submits the selected image to the Google Cloud Vision APIs. The APIs provide developers with the possibility to exploit pre-trained machine learning models for image labeling and classification. The APIs are able to identify several elements in the source image, e.g., objects, faces and texts. Moreover, the APIs can reconstruct valuable metedata, built upon the elements recognized in the source image. The APIs are available at the Google Cloud Platform web page (https://cloud.google.com/vision/), section AI & Machine Learning Products. The web page offers the possibility to try for free the basic functionalities of the APIs through a web-based interface, without the need to install and activate the APIs. As described in the online API documentation[4], the result of the Google Cloud Vision processing is a Json file which presents different fields structured as follows:

- *cropHintsAnnotation*: set containing 2D coordinates that specify the corners of possible crops, i.e., regions of the image to be remove since they could represent unwanted objects;
- *imagePropertiesAnnotation*: set of attributes to specify the dominant colors of the source image;
- *labelAnnotations*: list of labels representing the broad categories (objects, locations, activities, products, etc.) to which elements identified in the image belong to;
- *localizedObjectAnnotations*: list containing data which provide general information for each object, for example the name, its position, and the 2D coordinates of the rectangular region that contains it;
- *safeSearchAnnotation*: data to identify possible explicit contents, like adult or violent contents in the image;
- *webDetection*: web references which match with the source image.

The Json file that is returned by the Google Cloud Vision APIs is used in the following steps as a descriptor for the source image. In particular, the two lists *labelAnnotations* and *localizedObjectAnnotations* are considered in the generation of the 3D scene, since they contain data needed for identifying the objects to be synthesized. In the future, information contained in other fields could be considered to improve the results. For instance, the dominant colors could be used to influence the objects materials, whereas web references could be exploited to get more insights about the source image.

[4]Google Vision API documentation: https://cloud.google.com/vision/docs/

3.3 Models Database

To reconstruct the scene, objects are retrieved from an existing database of 3D models. The database contains the 3D mesh of the objects (stored as Filmbox .FBX files), their materials, textures, and metadata. Metadata include the following information (as an example, possible values are reported for an object named "waste bin"):

- alternative names for the object (dustbin, garbage pail, trash bin, wastebasket);
- categories object belongs to (park, garden, backyard);
- information which can be used to establish relations with other objects in the scene (more details will be provided in Section 4).

For the use cases, described in more detail in Section 6, a database containing 28 objects belonging to two specific environments (an indoor and an outdoor setting) was used.

3.4 Scene Creator Add-on

The back-end of the Scene Creator add-on was implemented as a Python script for Blender. The script receives the source image which was converted into the Json file described above by using Google Cloud Vision APIs. Data contained in the Json file and parameters representing the mood selected by the user are combined to generate the 3D scene. An algorithm was developed to determine the objects to be included and their position in the scene. The main steps of the algorithm are reported below.

1. All the objects, materials, textures, and lights already in the scene which may have been generated by a previous run of the algorithm are removed, and all the parameters and variable used by the algorithm are set to the default values.
2. The source image is processed with the Google Cloud Vision APIs to obtain the resulting Json file.
3. The Json file is parsed to extract the information in the lists *labelAnnotations* and *localizedObjectAnnotations*.
4. From the *labelAnnotations* list, the system determines whether the image represents an indoor or outdoor scene. Presently, if the list contains labels like room, bedroom, living room, etc. an indoor setting is assigned. For indoor settings, a predefined setup consisting of four walls, ceiling, and floor is automatically created in the 3D scene. For

outdoor settings (associated with labels, for example, grass, park, garden, etc.), a mesh representing the ground is inserted.

5. The *labelAnnotations* and *localizedObjectAnnotations* lists are navigated to look for a match between objects in the database and objects identified in the source image. The user can increase the number of objects in the scene by setting a value greater than zero in the Additional objects textbox. In this case, the algorithm tries to find additional matches between the object category (defined in the object metadata) and the labels in the *labelAnnotations* list.
6. All the objects to be inserted in the scene are organized in a graph-based structure. Each node represents an object, whereas edges are the spatial relations among objects. Relations are created based on the metadata contained in the Models database (additional details about the methodology used to identify spatial relations will be described in Section 4).
7. For the current node, the algorithm determines its position in the 3D scene by considering a set of rules; these rules avoid objects overlapping, and ensure that spatial relations and physical constraints are satisfied.
8. The current node is marked as explored, and the object is added to the scene.
9. Steps 7 and 8 are repeated for all the linked nodes of the current node still to process.
10. Lights are added to the scene and their parameters, as well as the materials and textures of all the objects placed in the scene are modified according to the mood selected by the user. The influence of the mood will be described in Section 5.
11. The main camera for rendering the scene is added. Its position in the 3D scene is fixed on a specific location for all the scenes, whereas the orientation is automatically adjusted in order to have at least one object in its field of view.

3.5 Blender

The output of the algorithm presented above is a Blender scene containing textured 3D objects, lights, and a camera. The scene does not require any further operations to be used. However, the user can further configure the objects' layout and their appearance using the Blender's native interface. For example, he or she can manipulate existing objects, add new objects, change the materials, set up a different camera, assign new textures, etc.

3.6 Virtual Reality Add-on

Once the user has defined the scene and its contents, he or she can navigate it as an immersive environment by activating the VR mode. This modality is supported by the Virtual Reality add-on, a Python script realized for Blender. This add-on was developed by leveraging the Virtual Reality Viewport library[5] and the Python bindings for Valve's OpenVR SDK[6] named Pyopenvr. The former library was exploited for the visualization of the Blender's viewport in VR through a head-mounted display. The functionalities offered by the latter library were used to retrieve position/orientation of the Vive's controllers and the buttons status. The developed add-on integrates the above data to enable interactions with the virtual contents through the Vive's controllers. To align the coordinate system of the Blender's 3D view and the virtual environment, the center of the generated scene is set as the origin of the virtual environment. Using the tracking data of the VR system, the user can move in the physical real space in order to explore the 3D scene. Currently, a one-to-one mapping is defined which remaps one Blender unit into one meter in the real world. The interactions with 3D objects pass through two stages: selection of the object and manipulation. The first operation is achieved by moving the virtual representation of the Vive's controllers (reconstructed in the VR environment by leveraging the tracking data) close to the object and pressing one of the Grip buttons. Through manipulation, the user can then change the position, orientation, and scale of the selected object by pressing the Trigger button and operating the controllers. Once the user releases the Trigger button, the object maintains the last transformation applied by moving/rotating the controller; the scale transformation is achieved by performing a 3D pinch gesture with both the controllers.

4 SPATIAL RELATIONS

The use of images as input for automatic scene generation poses the challenge of inferring the spatial relations among objects to be included in the scene. Differently than in the works where the object relations were assumed to be already reported in the text provided as input, the Google Cloud Vision APIs do not offer this kind of information.

[5]Virtual Reality Viewport: https://github.com/dfelinto/virtual_reality_viewport

[6]Pyopenvr: https://github.com/cmbruns/pyopenvr

To infer it, we selected 100 images representing indoor and outdoor environments by searching them on the web using several keywords related to the considered environments. Each image was processed with Google Cloud Vision APIs to extract the objects. For each of the objects recognized in the image, at least one spatial relation was manually assigned, by choosing it from a predefined set. This set was the same that was exploited in the text-based scene generation system in (Chang et al., 2014), and includes the following relations: *left, right, above, below, front, back, on top of, next to, near, inside, and outside*. The possibility to reconstruct realistic scenes by using this high-level descriptive information was already demonstrated in (Seversky and Yin, 2006; Li et al., 2009). The result of the above process generates a list of objects that can be found in these environments and their possible relations. These data have been used to build the Models database. In particular, the list of objects determines which objects have to be included in the database. Moreover, for each object in the Models database, a new entry was added to the object's metadata to report the probability to find a specific spatial relation with another object in the database. When the two objects are identified in the input image, the most recurrent relation is set. For example, for the "lamp" and "nightstand" objects, in the 100 annotated images we found a high probability to see the relation "lamp on top of the nightstand" (and viceversa). Thus, if in the source image a lamp and a nightstand are recognized, the above relation is set for the two objects.

The metadata contain also the bounding boxes that represent, for each possible relation, the spatial regions where related objects have to be placed. For example, Figure 3 shows the spatial regions defined for the nightstand object. In order to place the lamp on top of the nightstand (step 7 described in Section 3.4), the algorithm first gets the bounding box of the nightstand representing the "on top of" relation. Afterwards, a random 3D coordinate is generated into this region and assigned to the lamp object. Finally, the z coordinate (the vertical position) of the lamp is adjusted using the algorithm proposed in (Xu et al., 2002) to respect physical constraints and avoid a floating lamp over the nightstand. If the object does not contain any relation with other objects, it is randomly placed in the scene respecting only the physical constraints.

As described in (Xu et al., 2002), the proposed system also exploits hierarchical relations among the objects. For example, for the relation "the lamp is on top of the nightstand", the nightstand represents the parent node whereas the lamp is considered a child node. This hierarchy is also maintained in the Blender's data structure, in order to transfer further transformations possibly applied by the user with the native interface from the parent object to its children, as common in many modeling tools.

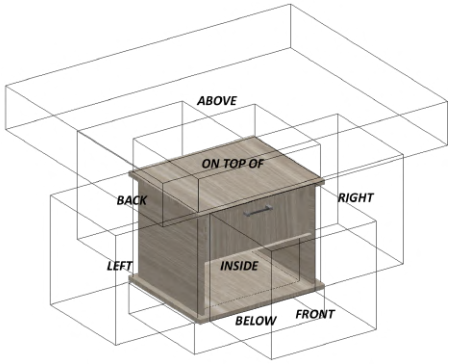


Figure 3: Spatial regions defined for the nightstand object.

If the user runs the algorithm several times by providing the same input for the source image, number of additional objects and selected mood, the system may create different alternatives of the scene due to the randomicity in the placement of objects in the assigned regions. Differences in the generated 3D scenes can be introduced also by setting the number of additional objects to a value greater than zero, as the algorithm randomly chooses the objects to be inserted from a set of possible alternatives generated by the category mapping. As a matter of example, the three scenes in Figure 4 have been generated with the same source image by setting the number of additional objects to three. For this image, the recognized objects (which were included in all the scenes) are the bed, the mirror, the cabinet, the cabinetry and the nightstand. The lamp, the carpet, the library, etc, were automatically added to the scene as additional objects.

5 MOOD INFLUENCE

As said, in this work two opposite emotions are considered: happiness and sadness. In order to define the affective aspects of the virtual environment depending on the mood selected by the user through the Automatic scene generation panel, we leveraged the main findings in (Sra et al., 2017; Krcadinac et al., 2015). According to these works, a scene conveys sad emotion when lights intensity is low; it contains few objects; objects' materials have dark tones with low saturation; textures present dot patterns. Conversely, a happy scene presents very intense lights;materials with very light, pure and saturated colors; textures with curved shapes or patterns with a high number

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

Figure 4: Scenes generated from the same source image (a), including a bed, a mirror, a cabinet, a cabinetry and a nightstand, and by setting the number of additional objects to three. Results of changing the mood for the same source image: happy mood (b,c), sad mood (d).

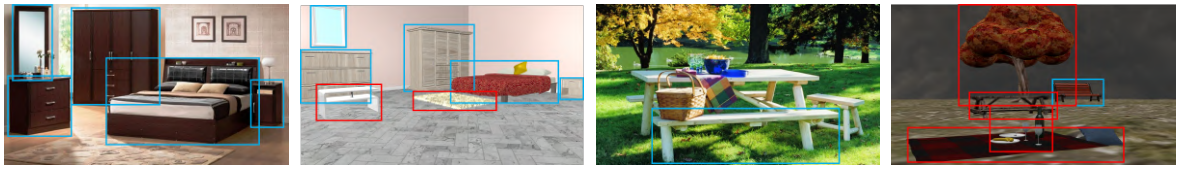
of small rounded elements. Moving from the above considerations, it is possible to create a map between the emotions required by the user and the various parameters that control the scene. Currently, the algorithm presented in Section 3.4 considers only lights, together with objects' materials and textures. In particular, if the happy (or sad) mood is set, specific materials and textures containing the proper features are automatically assigned to each object choosing them from a predefined set of available alternatives. Moreover, the number of lights in the scene and their intensity are adjusted to match the affective state. As said, Figure 4 presents three scenes generated from the same source: it is possible to appreciate the differences in terms of materials, image textures and lights due to the selection of a happy (Figure 4b, 4c) and sad (Figure 4d) mood, respectively.

6 USE CASES

In order to show the results which can be obtained with the algorithm proposed in Section 3.4, two use cases have been considered. The first use case concerns an indoor environment. As shown in Figure 5a, the source image depicts a bedroom with a number of furniture elements. The *localizedObjectAnnotations* list produced by the Google Cloud Vision APIs contains five objects (highlighted with a blue bounding box in Figure 5a): mirror, bed, cabinet, cabinetry and nightstand. These objects are automatically added to the 3D scene as shown in Figure 5b. As said, in order to increase the number of objects in the 3D scene, the *labelAnnotations* list is also considered. According to this list, categories identified in the source image are: bedroom, furniture, room, bed frame, bed sheet, interior design, wood, hardwood, floor. Considering these categories and assuming that the number of additional objects was set to two, two objects (the small table and the carpet) are added to the scene. The new objects are highlighted in Figure 5b with a red bounding box. Given the objects to be placed in the scene, the algorithm identified some spatial relations by retrieving them from the objects' metadata, e.g., the mirror on top of the cabinet, the nightstand to the right of the bed, the carpet near the bed, etc. The mood selected by the user affected materials and textures (the color of the wall, of a very bright hue, and the texture of bedding, which presents circular shapes), as well as the intensity of lights.

In the second use case, an outdoor environment representing a picnic table on a garden is considered (Figure 5c). For this source image, the Google Cloud Vision APIs identified only the bench object (blue bounding box). The *labelAnnotations* list contained ten categories: picnic table, outdoor furniture, outdoor bench, tree, outdoor table, leisure, picnic, recreation, park, sunlounger. From these categories, the objects that are visible in Figure 5d into red bounding boxes (tree, picnic table, carpet, and food) were extracted and added to the scene (for a number of additional objects set to four). The scene was created with a sad mood, as it can be noticed from the low intesity lighting and the textures with dark tones.

The database as well as the images used for the use cases are available for download at https://github.com/grainsgroup/AutomaticSceneGeneration After having analyzed many images representing indoor and outdoor settings, we found that, in indoor images, the number of objects presented and/or recognized by the Google Cloud Vision APIs is generally higher than for outdoor images. For this reason, the use of the labelAnnotation list becomes fundamental for the outdoor settings to obtain a rich scene. It is worth observing that, in order to consider more objects from the labelAnnotation list, the number of additional objects should be higher for outdoor settings than for indoor settings. However, if on the one hand a higher number of the Additional objects parameter can increase the richness of the scene, on



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

Figure 5: Considered use cases: a) indoor source image (source: https://bit.ly/2kW2HHh) containing five objects recognized by the Google Cloud Vision APIs (blue bounding boxes), b) result obtained with the happy mood and Additional objects (red bounding boxes) set to two, c) outdoor source image (source: https://bit.ly/33Owm6O) containing one object (blue bounding box), and d) result obtained with the sad mood and Additional objects (red bounding boxes) set to four.

the other hand the resulting 3D scene differs more from the source image since the new objects may not be contained at all in the source image.

A video showing the use of the system is available for download at https://bit.ly/2moaIVT.

7 EXPERIMENTAL SETUP

In order to assess the usability of the proposed system, a user study was carried out by involving 12 volunteers (6 males and 6 females), aged between 21 and 32 ($\mu = 24.81$ and $\sigma = 3.32$) who were selected among students and academic staff at the authors' university. All the participants could be considered as non-expert users from the perspective of 3D scenes creation, because of their low expertise in the use of computer graphics suites and VR systems.

7.1 Procedure

Before starting the experiments, participants well-informed about the procedure were asked to sign a consent form and to fill in a demographic questionnaire to evaluate their previous experience. Afterwards, each participant was requested to carry out two tasks in order to evaluate different aspects of the proposed system.

The goal of the first task (later referred to as T_1) was the evaluation of the system's usability. Considering this aim, each participant was first introduced to the steps for generating a 3D scene with the devised system. Then, they were requested to use it to create a scene from scratch by choosing a source image from a predefined set of images, setting the number of additional objects and the mood to be conveyed. Participants were instructed to complete this task at their pace, as no time limit was set.

The second task (T_2) was designed to widen the evaluation and consider three different aspects: similarity between the source image and the generated 3D scene, conveyed mood and spatial awareness. In particular, participants were requested to explore in VR four scenes generated by the proposed system using the same number of additional objects and by exploiting various source images and moods. The first two scenario (later referred to as *Indoor setting #1* and *Indoor setting #2*) are shown in Figures 6a and 6b. They represent two settings that were generated with the happy and sad mood, respectively. The other two scenarios, named *Outdoor setting#1* and *Outdoor setting #2* are shown in Figures 6c and 6d. They were generated to convey the sad and happy mood, respectively. As proposed in (Sra et al., 2017), the exploration was considered accomplished when the user had spent at least 30 seconds for each scenario in the VR environment (average time was 57s). Latin Order was used to choose the scenario to start and continue with, in order to reduce possible biases or learning effects. At the end of each task, participants were asked to complete a post-test questionnaire to evaluate specific aspects of interest.

7.2 Metrics

To evaluate the proposed system, both objective and subjective measurements have been considered.

The objective measurements exploited two indicators named *completion time* in T_1, and *object placement* in T_2. The first indicator considers the time needed by a participant to generate a scene, and provides a measure of the effectiveness of the devised system. The object placement indicator, defined in (Suma et al., 2007), evaluates the capacity of the participants to memorize the placement of objects within the scene, thus providing cues about spatial awareness provided by the integrated VR environment. To compute this indicator, participants were asked to show on a map representing the VR environment just experienced which objects they have seen in five specific positions (Figure 7 shows the five positions considered for the Indoor setting #2). The indicator was then computed as the percentage of objects which were correctly recognized in their actual position.

Subjective observations were collected by means of the questionnaire which was filled in by the users

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

Figure 6: Scenarios considered in task T_2: Indoor setting #1 (a), Indoor setting #2 (b), Outdoor setting #1 (c), and Outdoor setting #2 (d).

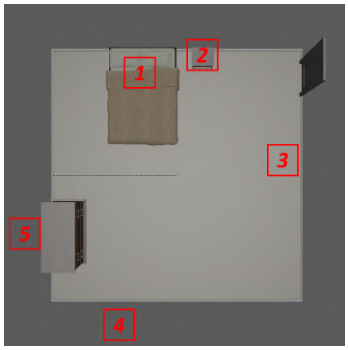


Figure 7: Map of Indoor setting #2 showing the five positions considered for evaluating users' spatial awareness.

at the end of each task. The questionnaire (available for download at https://bit.ly/2lYzFHk) is composed of two sections. The first section (filled in after the execution of T_1) evaluates the usability of the system according to the System Usability Scale (SUS) (Brooke, 1996). Questions were expressed in the form of statements to be evaluated on a 1-to-5 scale (from strongly disagree to strongly agree).

In the second section (administrated after T_2), users were invited to rate the similarity between the source images and the generated 3D scenes by providing a score in a 1-to-5 scale as presented in (Chang et al., 2015). Finally, participants evaluated the perceived mood, as conveyed by each experienced setting. As proposed in (Sra et al., 2017), participants were requested to provide a score in a range from 1 (very sad) to 5 (very happy).

8 RESULTS

Data collected with the demographic questionnaire filled at the beginning of the user study confirmed the low expertise of participants in using graphics suites and VR technology. With respect to the use of graphics suites, 36.4% of the participants said that they never used this software, 45.5% used it sometimes, 18.2% once a month or once a week. Considering VR, 45.5% of the participants never used VR systems, the remaining participants (54.5%) use them sometimes or once a month. In the following, the results of objective and subjective observations collected during the experiments will be presented.

8.1 Objective Measurements

Considering T_1, participants took slightly more than 70s each, on average to complete the generation of a 3D scene from scratch (μ = 71.08s and σ = 35.10). With respect to T_2, the analysis of the object placement indicator used to evaluate the spatial awareness shows that, after having navigated the 3D scene in the immersive environment, the participants were able to correctly remember the objects' layout since, on average, the percentage of objects recognized in the correct position was quite high (μ = 72.5%, σ = 0.15). This finding could be related to the improved spatial awareness that should be guaranteed by the use of VR for exploring the generated scene.

8.2 Subjective Measurements

As said, after T_1 the usability of the devised system was evaluated by exploiting the SUS scale. Overall, participants found the system as characterized by a high usability, since, according to (Brooke, 1996), the obtained score equal to 77.92 corresponds to grade B in the SUS scale (Adjective rating = "Good"). The average scores assigned to each statement are illustrated in Table 1.

Figure 8 shows the average scores (bars heights), medians (black lines) and quartiles (errors bars) obtained for what it concerns scene similarity and mood perception in the four settings (scores of mood perception for the Indoor setting #2 and the Outdoor setting #1 have been inverted, in order to have greatest values representing a higher similarity with the mood conveyed by the generated scene). Participants found that generated scenes were a good reconstruction of the source images, as confirmed by the average value of the four scenarios equal to 3.14 (which represents neutral similarity) Concerning mood perception, scores higher than 4.0 confirmed the capability of the system to convey different emotions in all the considered settings.

Table 1: Subjective results concerning usability based on SUS (Brooke, 1996).

Statements	Score
I think that I would like to use this system frequently.	3.42
I found the system unnecessarily complex.	2.00
I thought the system was easy to use.	3.92
I think that I would need the support of a technical person to be able to use this system.	1.42
I found the various functions in this system were well integrated.	3.75
I thought there was too much inconsistency in this system.	1.58
I would imagine that most people would learn to use this system very quickly.	4.33
I found the system very cumbersome to use.	2.08
I felt very confident using the system.	4.08
I needed to learn a lot of things before I could get going with this system.	1.25

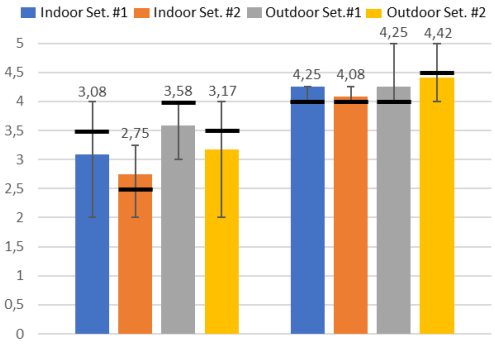


Figure 8: Subjective results: scene similarity and mood perception.

9 CONCLUSIONS AND FUTURE WORK

This paper presented a system that can be used by non-professional users to quickly create a 3D scene for fast prototyping. As said, the main aim was not to replace existing software tools, but rather to make them more accessible to non-expert users. Moving from this consideration, the paper introduced a methodology that combines the advantages of automatic 3D scene generation with VR technology. In particular, users are provided with an add-on developed for a well-known graphics suite, which is able to create a 3D scene starting from a 2D image that represents it. VR, in turn, can help the users to improve their understanding of the objects' layout, as it offers them the possibility to explore the generated scene into an immersive environment and to possibly make modifications to it using intuitive interfaces. The system also takes few steps towards the integration of affective aspects into the automatic generation process. A user study performed by involving users with no expertise in the above technologies produced promising results in terms of usability of the system, scene similarity, mood perception and spatial awareness.

Currently, the system presents limitations in terms of flexibility, i.e., possible scenes that can be created, due to the limited number of objects and related annotations (for defining the spatial relations) in the database. Possible evolutions will consider the introduction of techniques based for example on machine learning, able to directly recognize the spatial relations among objects in the source images. Efforts will be focused also on enhancing the number of 3D objects in the database, as well as on considering different settings. Moreover, future works will be devoted to compare the devised system with related works, as well as traditional software, by including in the evaluation more objective measurements. Finally, other data generated by the Google Cloud Vision APIs will be considered, e.g., the dominant colors of the image or the web references, with the aim to improve the similarity between the source images and the reconstructed 3D scenes.

REFERENCES

Ates, H. C., Fiannaca, A., and Folmer, E. (2015). Immersive simulation of visual impairments using a wearable see-through display. In *Proceedings of the 9th ACM International Conference on Tangible, Embedded, and Embodied Interaction*, pages 225–228.

Baños, R., Botella, C., Liaño, V., Guerrero, B., Rey, B., and Alcañiz, M. (2004). Sense of presence in emotional virtual environments. *Proceedings of Presence*, pages 156–159.

Brooke, J. (1996). Sus - A quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7.

Cannavò, A., Demartini, C., Morra, L., and Lamberti, F. (2019). Immersive virtual reality-based interfaces for character animation. *IEEE Access*, 7:125463–125480.

Cannavò, A. and Lamberti, F. (2018). 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*. The Eurographics Association.

Chang, A., Monroe, W., Savva, M., Potts, C., and Manning,

C. D. (2015). Text to 3d scene generation with rich lexical grounding. *arXiv preprint arXiv:1505.06289*.

Chang, A., Savva, M., and Manning, C. (2014). Interactive learning of spatial knowledge for text to 3D scene generation. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 14–21.

Chang, A. X., Eric, M., Savva, M., and Manning, C. D. (2017). SceneSeer: 3D scene design with natural language. *arXiv preprint arXiv:1703.00050*.

Coyne, B. and Sproat, R. (2001). Wordseye: An automatic text-to-scene conversion system. In *Proceedings of the 28th ACM Annual Conference on Computer Graphics and Interactive Techniques*, pages 487–496.

Crippa, G., Rognoli, V., and Levi, M. (2012). Materials and emotions, A study on the relations between materials and emotions in industrial products. In *Proceedings of the 8th International Conference on Design & Emotion: Out of control*, pages 1–9.

Esteban, I., Dijk, J., and Groen, F. C. (2011). From images to 3D models made easy. In *Proceedings of the 19th ACM International Conference on Multimedia*, pages 695–698.

Fascione, L., Hanika, J., Leone, M., Droske, M., Schwarzhaupt, J., Davidovič, T., Weidlich, A., and Meng, J. (2018). Manuka: A batch-shading architecture for spectral path tracing in movie production. *ACM Transactions on Graphics*, 37(3):31.

Hoorn, J. F., Konijn, E. A., and Van der Veer, G. C. (2003). Virtual reality: Do not augment realism, augment relevance. *Upgrade-Human-Computer Interaction: Overcoming Barriers*, 4(1):18–26.

Khatri, P. (2018). 3D animation: Maya or B33lender. *Global Sci-Tech*, 10(1):40–47.

Krcadinac, U., Jovanovic, J., Devedzic, V., and Pasquier, P. (2015). Textual affect communication and evocation using abstract generative visuals. *IEEE Transactions on Human-Machine Systems*, 46(3):370–379.

Li, C., Yin, C., Lu, J., and Ma, L. (2009). Automatic 3d scene generation based on Maya. In *Proceedings of the 10th IEEE International Conference on Computer-Aided Industrial Design & Conceptual Design*, pages 981–985.

Lu, J., Li, C., Yin, C., and Ma, L. (2010). A new framework for automatic 3d scene construction from text description. In *Proceedings of IEEE International Conference on Progress in Informatics and Computing*, volume 2, pages 964–968.

Naz, A., Kopper, R., McMahan, R. P., and Nadin, M. (2017). Emotional qualities of vr space. In *Proceedings of the IEEE Virtual Reality*, pages 3–11.

Pallavicini, F., Pepe, A., and Minissi, M. E. (2019). Gaming in virtual reality: What changes in terms of usability, emotional response and sense of presence compared to non-immersive video games? *Simulation & Gaming*, 50(2):136–159.

Payne, B. R., Lay, J. F., and Hitz, M. A. (2014). Automatic 3D object reconstruction from a single image. In *Proceedings of the ACM Southeast Regional Conference*, page 31.

Schmalstieg, D. and Stork, A. (2019). Unified patterns for realtime interactive simulation in games and digital storytelling. *IEEE Computer Graphics and Applications*, 39(1):100–106.

Seidler, M. (2018). Blender Eevee render engine in indie production: using Blender's Eevee render engine for art projects.

Seversky, L. M. and Yin, L. (2006). Real-time automatic 3D scene generation from natural language voice and text descriptions. In *Proceedings of the 14th ACM International Conference on Multimedia*, pages 61–64.

Sra, M., Garrido-Jurado, S., Schmandt, C., and Maes, P. (2016). Procedurally generated virtual reality from 3D reconstructed physical space. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, pages 191–200.

Sra, M., Maes, P., Vijayaraghavan, P., and Roy, D. (2017). Auris: creating affective virtual spaces from music. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, page 26.

Suma, E. A., Babu, S., and Hodges, L. F. (2007). Comparison of travel techniques in a complex, multi-level 3D environment. In *Proceedings of the IEEE Symposium on 3D User Interfaces*.

Vogel, D., Lubos, P., and Steinicke, F. (2018). AnimationVR-Interactive controller-based animating in virtual reality. In *Proceedings of the 1st IEEE Workshop on Animation in Virtual and Augmented Environments*, pages 1–6.

Vouzounaras, G., Daras, P., and Strintzis, M. G. (2014). Automatic generation of 3D outdoor and indoor building scenes from a single image. *Multimedia Tools and Applications*, 70(1):361–378.

Xu, K., Stewart, J., and Fiume, E. (2002). Constraint-based automatic placement for scene composition. In *Proceedings of Graphics Interface*, volume 2, pages 25–34.

Zyda, M. (2005). From visual simulation to virtual reality to games. *Computer*, 38(9):25–32.