Multimedia Tools and Applications (2023) 82:11721–11751
https://doi.org/10.1007/s11042-022-13687-1

Application in multimedia: from camera to VR

Aamir Wali[1] · Aliza Lisan[1] · Hammad Ather[1] · Muhammad Qasim[1] · Muhammad Uzair Abid[1]

Received: 12 February 2021 / Revised: 14 April 2022 / Accepted: 15 August 2022 /
Published online: 1 September 2022

Abstract

This work describes a framework that allows children and domestic users to create architectural structures like mazes, houses etc., and navigate them in virtual reality (VR). The user can draw a 2D map of a maze etc. using a simple paper, pen and ruler. The application works by taking as input such a hand drawn image via gallery or camera, and then building a 3D model in VR using Unity3D. The 3D model completely follows the design of the floor plan including the placement of doors and windows. Detecting and then constructing walls, doors and windows on the fly in VR is a challenge. The user can also customize walls, windows, doors and floors. This is done by looking at the object and using the VR controller to select various options. The purpose of the framework is purely entertainment and is specifically designed for children and domestic users. From hand-drawn images on a paper to ready-to-navigate 3-D model in VR in just one click is the novelty of our work. User is completely isolated from the complexities and intricacies of programming and usage of different tools and platforms. We also extend our work from hand drawn designs to professional home blueprints and floor plans. We used fully convolutional network to extract walls from the professional floor plans with mean pixel accuracy of 97.3%. For hand-drawn designs, we were able to detect doors and windows with MAE of 0. Our wall segmentation method reported an MAE of only 1.2.

✉ Aamir Wali
aamir.wali@nu.edu.pk

Aliza Lisan
l145851@lhr.nu.edu.pk

Hammad Ather
l144250@lhr.nu.edu.pk

Muhammad Qasim
l144157@nu.edu.pk

Muhammad Uzair Abid
l144036@nu.edu.pk

[1] Department of Computer Science, National University of Computer and Emerging Sciences, Lahore, Pakistan

Keywords Virtual scenario · User experience · Virtual reality · Paper to VR · Maze design · VR games · 3D modeling

1 Introduction

VR has gained popularity in many fields such as architecture [36], interior design [49], health [28], education [19], games and games-related entertainment. A study shows that VR adds another layer of enjoyment in games and gameflow [41]. This rising interest in VR has led to the birth of amazing VR-based products such as PlayStation VR by Sony, HTC Vive, Oculus Rift, Oculus Quest, head mounted display (HMD) and base stations to track the position of users wearing HMD, etc.

Generally, in VR applications the designing has to be done in advance. Focusing only on architecture and games that involves navigating a map, the buildings and pathways need to be designed in advance. There is no storytelling by the user in the sense that they can create their own design. This is to prevent the users, especially domestic users, from going through this time-consuming effort.

Recently, a new game genre has been proposed that combines user-generated design with game play [13, 14]. In this approach, the user can design a maze in a physical space and play in the VR environment. The model works as follows: The user creates a physical maze on a fixed 4x4 block on a floor using colored tape. Then the user creates the same maze in the application. In [14] this is done by connecting dots on a 4x4 canvas. In [13], the user views this maze through the application using a camera, and manually identifies the endpoints of walls to translate this maze onto a 2D canvas in the application. Then the user renders this 2D image into 3D model. Finally, the user can navigate the maze in VR by actually walking in the physical maze wearing position-tracking HMD.

This model only works on a 4x4 block. This places a limit to the number of design a child can produce. The model itself is constrained by the size of the room. Another issue is creating the physical 4x4 maze. The maze is made with precise measurements. The application is rigidly tied to the actual size of the maze so each of the 16 squared sub-blocks must be of the same size. By design, the 2D to 3D rendering process for VR also does not seem to be scalable. The conversion process *knows* where the walls *can* be. All it does is mask or unmask the walls depending on how the user created the maze in the application. Finally, users would need a HMD and tracking sensors so that the application can track their position.

In this paper, we propose a framework that allows user groups such as children and domestic users who lack technical IT skills to create architectural structures like mazes, houses etc. on paper and navigate them in VR purely for entertainment purposes. The user can draw any maze or map on paper using a pen and a ruler. By taking a picture of the maze by camera and using just one click, the user can view and explore the maze in VR. The images can be anything from hand-drawn lines to professional floor plans. The user can also draw and create doors and windows. The doors can be opened and closed. Most importantly, all this can be done in just 1-click. Also, the user can add textures to walls and floors.

The rest of the paper is organized as follows. Section 2 provides a brief background on VR-powered related work. Section 3 presents the details of our framework. Section 4 highlights the results and discussion. Conclusion follows in Section 5.

2 Background

There are various tools that are used by professionals to model structures in 3D, some of which can be viewed in VR. However, there are hardly any tools or software that can be used by children or domestic users. Some of these software are discussed next.

2.1 3ds Max

3ds Max [3] developed by Autodesk, is used for developing 3D models and animations for games, CGI movies and architectural structures. It does provide a lot of features but overall this tool is for expert users. A user would need to learn the 2D modeling tool before developing 3D models. Also, it does not have the option to show the 3D model of the house in VR or even in first-person View.

2.2 Autodesk maya

Maya [1] is also a 3D modelling software also by Autodesk. It is mainly used for animations. It can also be used for modeling, but like 3ds Max, it is not easy to use. A user has to get familiar with the nuts and bolts of the software before building the 3D model of the house and that can take a lot of time.

2.3 Sketch up

Sketch Up [42] is a very 3D modeling tool developed by google. It provides a canvas on which the user can design anything. It lacks many architectural tools such as the wall tool, one of the basic tools to design a building. It was developed to help out students of architectural engineering who have experience using such tools.

2.4 Revit

Revit [43] provides some very amazing and thrilling features like work-sharing and interoperability. However, it lacks many basic features like texture editing or VR view of the design. Revit has a complicated user interface that is hard for a beginner or computer-illiterate to grasp. It does allow 3D modeling and a 2D view of the design.

2.5 Cinema4D

Cinema 4D [27] is 3D modeling, animation, motion graphic and rendering application developed by MAXON Computer. Cinema 4D provides a built-in house builder tool to build a house. It does not provide features for doors and windows. It only provides the feature to make a cut in the wall for doors and windows. A user has to design the 3D models of doors and windows on their own.

2.6 Blender

Blender [44] is one of the most used software for 3D modeling because it is open source. It has tools to build a 3D model, animations and it allows texture designing of any object. It has a complicated user-interface which is easy for a professional to comprehend. Blender

does provide a VR view of the design but it has some subscription fee for these advanced features.

2.7 Planner5D

Planner 5D [35] is a web application specifically made for modeling houses. It is easy to use. It provides 2D layout and a 3D mode to explore the house. It also provides options to add textures to walls and floors. The user can also view the created house in VR but the rendered house is of poor quality. The user must pay for HD rendering of house. Planner 5D also does not allow the user to input his own blueprint. Also, there is no option to create a single wall, a user must create a whole room. It allows the user to view the house model in 3D but not in first-person view. So, users don't feel like they are walking in a house.

2.8 CAD-to-VR modeling

CAD is another technology that is used by architects and engineers all around the world for the detailed engineering of 2D and 3D design and modeling. It also provides rendering of the objects and is widely used by civil engineers. With the emergence of VR, techniques to export CAD data into VR were developed. This would allow users to view their product designs in real-time at real scale.

Usually, CAD-to-VR transformation is a two-step process. CAD data is first exported to one of the CAD format files such as .jt file extension, and then using a virtual reality editor such as Autodesk Red, etc., the .jt file can be imported and the entire assembly can be view in the VR editor. A number of VR software are now available that can perform CAD-to-VR transformation. A few of these are discussed next.

2.8.1 Exxar CAD-to-VR

Exxar CAD allows the user to create or load entire 3D models. It works for a variety of 3D file formats allowing it to integrate with other CAD software. One of its feature Exxar CAD-to-VR [11] allows the users to convert the entire 3D model into VR in less than an hour. This also has the human interaction component with body tracking capabilities and tools for ergonomics. Exxar CAD in itself is a very powerful tool that is used by various industries such as the automobile to design cars and engines. With VR technology, the designer and customers can get a more immersive experience. The same can be done for architectural structures.

2.8.2 CAD-to-VR for autodesk inventor

The CAD-to-VR for Autodesk Inventor tool converts CAD design to VR for the Inventor CAD [4] software. This tool is not available freely. The Inventor application can create and load models that are then fed to the CAD-to-VR conversion application allowing the model to be viewed in a web browser.

2.8.3 Siemens NX virtual reality

Siemens NX virtual reality [39] is yet another tool that allows users to view their CAD design in real-scale. The only supported VR kits are HTC Vive and Vive Pro.

Other than these tools, we did not find any recent research on conversion from CAD to VR. As far as these tools are concerned, they only provide 3D to 3D conversion. That is, given a CAD design or hefty CAD format file, these applications convert these designs to VR. In our application, a simple picture spanning a few kilobytes taken from any ordinary mobile phone is sufficient to create the VR environment in 3D. The picture in itself can be a hand-drawn design or sketch or an architectural floor plan, so the sketch-to-VR use case also applies to our application. Also, these applications are designed for experts and people with sound knowledge of 3D and 2D modeling, and are not suitable for children.

2.9 Other applications

A number of other paper-to-AR/VR applications have been developed such as 360Proto [29], Pen and Paper [12], etc. However, in all such applications the usage of the word 'paper' is in the sense of paper prototyping. In paper prototyping for VR, the designs and images are created using specific VR sketch sheets and templates [6, 21–23], and prior knowledge and expertise is needed to use these sketch sheets and templates. Furthermore, the designs and sketches are created in perspective and viewed, as they are, in a VR player [20]. There isn't any 3D graphical construction from hand drawn sketches or mazes as proposed in this paper.

2.10 Summary

In this paper, we propose a framework that converts drawings on paper into VR for entertainment. A summary of all the tools discussed in this section is provided only to highlight this novelty of our framework. For this purpose, a feature comparison matrix is also given in Table 1.

3 Research studies

3.1 3D modeling

Besides the tools and applications discussed in the previous section, a lot of research work has been done to convert 2D floor plans to 3D models such as [15, 18, 24, 51]. In each of these studies the work flow is more or less the same: walls are recognized from 2D images and translated to 3D model either by using a third party 3D modeling tool or by extruding edges along the wall height. Horna et al. [18] proposes a method that first detects edges from the image and then combines them to form a closed shape. For example, the walls are combined to form a closed room. The 3D models is constructed using a third-party modeler MOKA [45]. MOKA can generate multi-storey, 3D models of buildings.

Similarly, Gimenez et al. [15] proposes a techniques that also takes as input 2D images of floor plans and translates them to IFC format file which has to be manually opened in DDS-CAD viewer to generate the 3D model. The primary focus of this paper is to recognize walls and openings. This method recognized the walls and openings with an accuracy of 86% and 62% respectively. In both [18] and [15], the different sections of the 3D model can be viewed but they cannot be navigated or explored in first person view, something that is possible in VR.

Zhu et al. [51] and Lewis and Séquin [24] also extract the edges of the walls. However, the 3D model is created by extruding the edges to the wall height. The openings for the doors

Table 1 Feature comparison matrix of our proposed framework and other popular 3D house modeling applications to highlight the novelty of this paper

Reference	Modeling in 2D	Modeling in 3D	VR view	Hand-drawn Image Input	Texture Designing	User friendly	Free
3Ds Max	✓	✓			✓		
Maya	✓	✓			✓		
Sketchup	✓	✓	✓		✓	✓	
Revit	✓	✓					
Cinema4D	✓	✓			✓		
Planner5D	✓	✓	✓		✓	✓	
Blender	✓	✓	✓				
Proposed Framework	✓	✓	✓	✓	✓	✓	✓

and windows are cut off from the model in the next step. A sample 3D model is shown in Fig. 1 (taken from [51]). This model has one fixed view as shown in the figure. Such a model can be compared to 'miniature' models that have an outer view but cannot be navigated from inside. Having said that, the model generated by [24] is Berlekley WALKTHRU [5] compatible that provides an interface to navigate through the building which is by no means comparable to the experience of navigation in VR. The 3D model needs to be integrated into WALKTHRU and requires certain degrees of expertise in the software operations.

3.2 Sketch recognition

In this paper since we attempt to recognize doors and windows in hand-drawn images, it would be worthwhile to review some of the latest studies on sketch recognition.

When drawing or painting, the artists usually start with a sketch [7]. To digitize this process, sketch recognition caught the attention of academics in the field of computer vision and machine learning. Also, with the increase in the use of hand-held touch screen devices, it has become more important in the fields of human-computer interaction and computer vision to understand free-hand sketches by users. In this section, we will review related literature and present some proposed techniques for sketch recognition.

In [30], a pre-trained model ResNet-50 is selected and fine-tuned in order to obtain better classification by reducing the cross-entropy loss. To reduce over-fitting, 1 flatten layer is added to this model along with 1 drop-out layer with a 0.5 probability value. Moreover, the dataset used for training and validation is the TU-Berlin test dataset. 10,000 images were used for training this model to fine-tune. 5000 images for validation and testing each were used. This methodology gave a testing accuracy of 74% which is better than the accuracy of human recognition for this dataset i.e. 73.1%.

Another proposed technique for sketch recognition, known as the Hybrid Convolution Neural Network [52], consists of a two-branch CNN and is a combination of sketch appearance and shape. The appearance features are extracted by the first branch called the A-net,

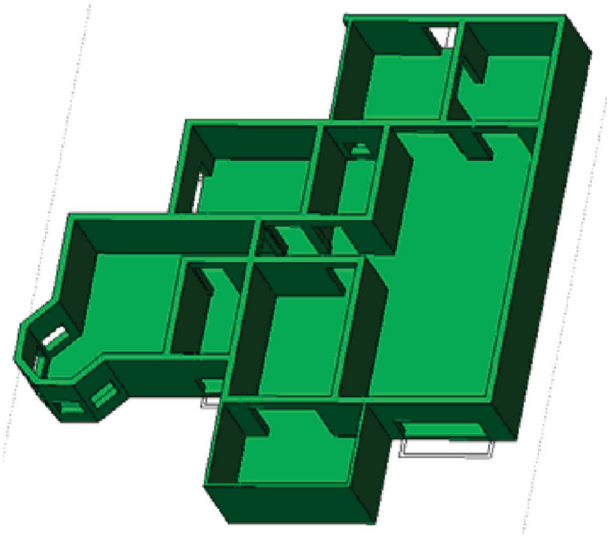

Fig. 1 Sample 3D model taken from [51]

while the second branch S-Net extracts the shape features. A-net is influenced by AlexNet. Both the branches have different input types and provide appearance and shape features. After this, the two feature vectors are normalized and concatenated to achieve a hybrid feature vector. Sketch recognition such as sketch classification is performed by this hybrid representation of features.

In another study, [52], the proposed model on sketch classification is evaluated. The experimental results show that the model performs better than 7 state-of-the-art models consistently on both Tu-Berlin and Sketchy datasets with 84.42% and 82.74% accuracy respectively.

3.3 Floor plan parsing

Finally, we review some studies that use machine learning techniques to recognize different items in professional floor plans. The problem of floor plan parsing has been studied in various forms over the years and can be succinctly described as one aiming to gain meaning from the floor plan. This has produced a set of strict heuristics-based image-processing approaches and the newer image segmentation approaches.

The initial heuristics-based solutions aimed to extract floor plan elements by making assumptions on the floor plan elements such as the stroke thickness of the graphical notation [2] or the shape of the graphical elements for detection [10]. These initial works failed to generalize to real-world floor plans due to the presence of semantic noise such as grid lines, a plethora of symbols used by architects to add further meaning such as the presence of stairs, insulation, load-bearing walls, etc. as well as no restriction on the direction of walls. To extract some sort of semantic meaning from the floor plan rather than blindly following a set of rules, machine learning approaches were explored.

While initial deep learning approaches gave a simple image segmentation treatment of utilizing a large segmentation network to extract a set of binary maps representing floor plan elements [9, 50], the newer approaches aim to utilize some logical cue from the arrangement of the floor plan elements and the structure of the floor plan to better the segmentation results. Wang et al. [25] used an appropriation of a Resnet-152 to segment junctions of various types from the floor pan and then used integer programming to connect compatible junction points forming a vectorized representation of several floorplan elements. However, this approach assumed the principal directions of the floor plan elements, e.g., a wall could not be curved or diagonal.

The more recent approaches have introduced more elegant semantic cues. Wang et al. [47] introduces a multi-task approach of room-boundary element (consisting of walls, doors and windows) segmentation and room-type recognition using a common VGG encoder and a decoder for each task. Room-boundary features in the different layers of the decoder are used with attention to give semantic cues for the room-type prediction decoder. Both the multi-task loss as well as the within-task loss is weighted. This added context as well as the multi-task approach extracts more meaning from the floor plan and grants major segmentation accuracy improvements in both floor plan element mean average precision as well as room type prediction. Xu et al. [48] also tackles the problem of room-boundary segmentation and room-type prediction using a shared VGG featurizer and individual decoders, but it adds an architectural difference by taking inputs from the room-boundary decoder layers and the room-type prediction decoder layers to a separate sequence of boundary attention aggregated model (BAAM) convolutional units. Similar to [47], a weighted loss function is applied to the two prediction losses however, BAAM demonstrates lower overall performance.

Xu et al. [46] tackles the problem of room-type prediction differently by using decision trees with five textual features representing structural information including room area ratio, window ratio, number of edges, doors and windows. These features are extracted from a vectorized representation of the floor plan. This vectorized representation is generated by refining the original raster image. This refinement is done by distinguishing door and window types by first inferring each using a version of the YOLOv3 network. While this special treatment of door and window detection provides an interesting approach, the limitation of generated textual features for room type prediction does not break the state-of-the-art accuracy.

Finally, in [40], a novel wall detection algorithm is proposed. The dataset consisted of the historical floor plans of the Versailles Palace dated between 17^{th} and 18^{th} century. The dataset had no ground truths for the walls so these were generated using multi-directional steerable filters and manual manipulation. In this paper, U-net based convolutional framework is used for wall detection. Mean intersection over union is used as an evaluation metric.

4 Proposed framework

The main idea of our proposed framework is to construct a whole 3D representation of a house or maze using just an amateur hand-drawn sketch in VR. The primary purpose of the proposed framework is entertainment whereby user groups such as children and domestic users would be able to create architectural structures like mazes, houses etc. on paper and navigate them in VR. However, this framework can be used a building block for many other applications in different areas such as Facebook's metaverse that would require large scale creation of buildings in the digital realm. From medical education where a system such as digestive system can be modeled and viewed and learnt by navigating from the inside to a multiplayer game generated from a piece of paper are some of the future applications of this framework.

The user can draw any maze or map on paper using a pen and a ruler. A sample drawing is shown in Fig. 2. The straight horizontal and vertical lines are the walls of the house, whereas the triangles represent the doors. The rectangle with a cross is the window.

The framework comprises of two main steps: image processing and 3D model generation using Unity3D scripting. In the image processing phase, the essential features of the floor plan are isolated such as the walls, windows and doors. The 3D model generation phase further has two modules. In the first module, the image obtained from the image processing step is input into a Unity3D instance to construct an accurate representation of the floor plan. Any required animation is also handled at this stage. The second module is the VR module that handles elements related to navigation in VR such as camera, lights, customization etc. Details for each of these components is discussed in the following subsections.

4.1 Door detection

The doors are represented by right-angled triangles as shown in Fig. 2. Instead of using quarter circles that are traditionally used to represent doors in architectural drawings, we use right-angle triangles as they are much easier to draw. The triangles also capture the door-opening directional information that is provided by quarter circles. The hinge of the door is represented by the corner of the triangle that has the right (90°) angle.

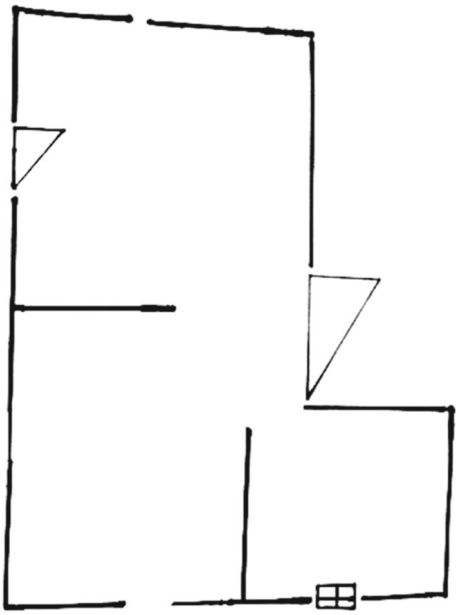

Fig. 2 Sample hand-drawn floor plan. The triangles represent doors, while the rectangle represents a window

The findContours [31] algorithm was used to find the coordinates of all the connected components in the image. A characteristic of the right-angled triangle that sets apart from the other features is that it is a polygon with three vertices. While any polygon detection function could be used, the main challenge was the irregular nature of the hand-drawn triangles as shown in Fig. 3. The irregular shape was a problem since the polygon detection algorithm instead of detecting 3 points for the triangle's connected component, detected a lot more than 3 points.

The problem required a convex hull to be found for the triangle in order to find the smallest convex set that contains all of the points of the triangle contours. Coupled with some image cleaning and blurring, the convex hull was found using [32]. The new hull contours were then used in the approxPolyDP [33] function that first uses the Douglas-Peucker algorithm to approximate a curve or a polygon with a given curve or polygon using a fewer number of points. Once the resulting polygon is identified as a triangle, the function then extracts the vertex coordinates of the triangle. Once the vertex coordinates of each triangle (door) were found, they were rearranged and stored in a file according to the format given below:

Fig. 3 Zoomed in triangle representing a door

< **number of triangles** >:< **Ax1** >< **Ay1** >< **Bx1** >< **By1** >< **Cx1** >< **Cy1** >:< **Ax2** >< **Ay2** >< **Bx2** >< **By2** >< **Cx2** >< **Cy2** >: ... :

In the format, the information of each door is grouped by the colon (:) symbol. The Bs are the coordinates for the hinge of the door and As are the coordinates of the opposite side of the door. The Cs are the coordinates that tells us the point (or direction) in which the door can open.

4.2 Window detection

The windows are represented in Fig. 2 by a rectangle with a cross pattern. The strategy followed to detect the window symbols were quite similar to that for the doors. The windows are distinctive in the floor plan in the fact that they are polygons with four vertices containing four internal contours each having four further vertices. Using the same process of finding the convex hull, the outer contour vertices of a polygon with 4 corners were found. For each such polygon, the internal contours were checked using the contour hierarchy to determine if the whole polygon is a window. The format used to store the windows in the file is as follows:

< **number of windows** >:< **Ax1** >< **Ay1** >< **Bx1** >< **By1** >:< **Ax2** >< **Ay2** >< **Bx2** >< **By2** >: ... :

In the format, As and Bs are the coordinates for the diagonal ends of the windows. If the rectangle does not contain a cross pattern, it would be considered as a small closed room. Coordinates of different windows are separated by a colon (:).

4.3 Wall detection and segmentation

The walls are represented by horizontal, vertical or diagonal hand-drawn, black lines on a plain A4 white paper. Note that once the doors and windows are detected and removed, the only thing that remains are the walls. Due to the irregular nature of hand drawn floor plans, wall detection mechanism and other precautions have been implemented so as to make the application more robust.

The image received by the wall detection step is transformed from grayscale to binary image. Keeping the grayscale-to-binary threshold low, cleanses our image from noise and unnecessary objects such that only white walls remain against a black background. Finally, we detect vertical, horizontal and diagonal walls by using Canny edge detection algorithm.

After this morphological operators such as erode and dilate are applied. This helps to fill-in the rectangles and ensure there are no gaps or missed out pixels. At this point depending on which mask was used, we get 3 images that only contain walls that lie in a particular orientation. The Canny edge detection algorithm gives an image with pixel-wide outlines which is best for contour detection. The contour detection function is used to find the coordinates of each individual wall and stores them in an array. Since these coordinates do not form a perfect rectangular shape, we apply a bound contours function [34] on it which encompasses each component as a perfect rectangle. Coordinates of these bounding rectangles are then forwarded to our Unity instance where they are used to create walls. The format used to store information about the walls is as follows:

< **number of walls** >:< **Ax1** >< **Az1** >< **Bx1** >< **Bz1** >:< **Ax2** >< **Az2** >< **Bx2** >< **Bz2** >: ... :

The first value indicates the number of wall segments followed by the information for each wall segment separated by a colon symbol. The As and Bs are the diagonally opposite corners of each wall segment.

4.4 Construction in unity3D

The Unity module takes the coordinate information as input from the image processing module and builds a complete structure of the house in a Unity instance. For this purpose, the file containing all the coordinates is accessed at this stage. Usually, in Unity3D the entire 3D model is stored in a certain format such as in a .obj extension file. When the .obj file is loaded, the complete 3D structure is produced in Unity3D. Following this idea, we come up with our own specific format for storing the coordinates of the walls, doors and windows. The file first contains information on the number of walls and their coordinates. Then the information on doors and finally the windows. The specific format for storing the information for the wall, doors and windows have already been discussed in previous sections.

First, the walls are loaded and inserted using scripting. The script for loading the coordinates for walls from file is given in Algorithm 1. The size of the walls is scaled using a constant value of 1.6 that is calculated after experimentation.

```
1:  function START
2:      TextAssettxt = (TextAsset)Resources.Load("Tekton_sample600",typeof(TextAsset));
3:      float Ax1 = 0, Az1 = 0, Bx1 = 0, Bz1 = 0
4:      string[] walls = text[current_index].Split(':');
5:      ....
6:      for string word in walls do
7:          int i = 0;
8:          string[] coords = word.Split(' ');
9:          Ax1 = float.Parse(coords[i++]);
10:         Az1 = float.Parse(coords[i++]);
11:         Bx1 = float.Parse(coords[i++]);
12:         Bz1 = float.Parse(coords[i++]);
13:     end for
14:     float diffX = Math.Abs(Ax1 - Bx1);
15:     float diffZ = Math.Abs(Az1 - Bz1);
16:     ....
17: end function
```

Algorithm 1 Script for loading coordinates of walls from file

Doors and windows are also inserted through scripting. The doors can also open and close. This also required scripting. Once the user is in a close proximity of a door, the rotate transform values of the door is changed to produce the animation of an opening or closing door. For this purpose, the *sphere collider* components are used. The position of the hinge, and the direction in which the door can open depends on how the user drew the door triangle. Recall that for each door, we extracted 3 coordinates A, B and C during the door detection phase (see Section 4.1). One of these coordinates, B is the hinge coordinate that tell us the position where the hinge object must be placed. Hinge object is an empty *game object* and specifies the side of the door that is hinged. The hinge object is made the parent of the door object. In order to rotate the door, we simply need to rotate the hinge object.

The A coordinate is the opposite side of the door and this is used to determine both the width of the door, and the orientation of the door, i.e., vertical or horizontal. Once the

hinged-side of the door is determined, the width is calculated and the orientation of the door is known, the third coordinate C specifies whether the door will open away or towards the user. When placing the door, all of these different cases had to be handled. Algorithm 2 specifies the script for one specific case of creating a horizontal door and opening it downwards.

```
1: function START
2:     // CASE 1: Horizontal Door
3:     if Math.Abs(Ax - Bx) > Math.Abs(Ay - By) then
4:         ...
5:         //CASE 1A: opening downwards
6:         if if (Cy < By) then
7:             //CASE 1AR: Hinge on the right
8:             if Bx < Ax then
9:                 pivot.transform.position = new Vector3(–0.5F, –0.5F, 0.5F);
10:                door.transform.parent = pivot.transform;
11:                pivot.transform.position = new Vector3(Bx, 0, By);
12:            else//CASE 1AL: Hinge on the left
13:                pivot.transform.position = new Vector3(–0.5F, –0.5F, 0.5F);
14:                door.transform.parent = pivot.transform;
15:                pivot.transform.position = new Vector3(Ax, 0, Ay);
16:            end if
17:            pivot.transform.localScale = new Vector3(scale, doorHeight,
    doorThickness);
18:            Animator anim = pivot.AddComponent<Animator>();
19:            ...
20:        end if
21:    end if
22:    SphereCollider sphere = pivot.AddComponent<SphereCollider>();
23:    sphere.radius = 1;
24:    sphere.isTrigger = true;
25:    pivot.AddComponent<DoorScript¿();
26:    ...
27: end function
```

Algorithm 2 Excerpt from script for opening horizontal door downwards

The script for creating both horizontal and vertical windows is listed as Algorithm 3. The coordinates A and B are the diagonal ends of the window that are read from the file.

In the hand drawn images, the doors and windows may be separated from the walls by a small gap. If there is a wall very close to the window or door, this gap is filled by extending the wall till it hits the edge of the door or the window.

4.5 VR module

After the Unity module is done making the structure, the VR module comes into play. The VR module is responsible for the movement of the camera and the controller in the Unity instance built by the Unity module, so that the user can move in the structure of the house in

```
function START
    ...
    GameObject myParent = new GameObject();
    myParent.transform.position = new Vector3(-0.5F, -0.5F, 0.5F);
    window.transform.parent = myParent.transform;
    myParent.name = "Window";
    //CASE 1: Horizontal wall
    if Math.Abs(A1x - B1x) > Math.Abs(A1y - B1y) then
        scale = Math.Abs(A1x - B1x);
        myParent.transform.localScale = new Vector3(scale, windowHeight, 1);
        myParent.transform.position = new Vector3(A1x, 0, A1y − 1);
    else
        scale = Math.Abs(A1y - B1y);
        myParent.transform.localScale = new Vector3(scale, windowHeight, 1);
        myParent.transform.rotation = Quaternion.Euler(0, 90, 0);
        myParent.transform.position = new Vector3(B1x, 0, B1y – 1);
    end if
    ...
end function
```

Algorithm 3 Script for creating horizontal and vertical windows

VR. The VR module also adds functionalities like collision detection, rigidness, and visual improvements like changing textures and material of the structure. The user can interact with the VR environment such as customize parts of the house such as the floor, walls, doors and windows using a Bluetooth controller.

4.6 Evaluation

For the evaluation of the hand-drawn designs, we first had to create our own dataset.

Dataset generation For this purpose, three school-going children of the ages 8, 9 and 10 were given 15 A-4 sheets each, a marker and a ruler, and asked to draw different designs. This activity was spread across two days and was done by the children on their own, at their home. This was not done under the supervision of any member of the research team. Prior to this, every child was given a brief introduction and a demo so that they understand what was required of them in the presence of their parent(s). A total of 43 designs were provided. These designs were converted to images by the same children. Finally, three labels were assigned to every image: wall segment count, door count and window count that indicate the number of wall segments, doors and windows in the image respectively. These were entered manually for each image. This completes the dataset.

For evaluation, we used two evaluation metrics: the wall-pixel accuracy and mean absolute error (MAE).

Wall-pixel accuracy Once the door and windows have been detected and removed, we are only left with an image consisting of the walls. This image is called the predicted image. This predicted image is compared with the original hand-drawn image using wall-pixel accuracy. Let n_{ww} be the number of wall pixels predicted as wall pixels, and let t_w be the

total number of wall pixels respectively, in the original image, then wall-pixel accuracy is given by:

$$\text{wall-pixel accuracy} = n_{ww}/t_w \tag{1}$$

Note that during evaluation, the door and windows were also removed from the original image manually so that there is a wall-to-wall comparison between the original and predicted images. For each wall segment, doors and windows, we used another evaluation metrics: MAE for the walls, doors, and windows.

Wall, Door, and Window MAE For every image in the dataset, once the walls have been segmented, and the doors and windows have been extracted, and their individual count is known, MAE is calculated. For image i, let p_{i1}, p_{i2} and p_{i3} respectively be the predicted wall, door and window count and a_{i1}, a_{i2} and a_{i3} be the respective actual wall, door and window count given in the dataset, and N are the total number of images, then the MAE for wall, door, and window count are given by:

$$\text{wall-count MAE} = \frac{1}{N}\sum_{i=1}^{i=N}|p_{i1} - a_{i1}| \tag{2}$$

$$\text{door-count MAE} = \frac{1}{N}\sum_{i=1}^{i=N}|p_{i2} - a_{i2}| \tag{3}$$

$$\text{window-count MAE} = \frac{1}{N}\sum_{i=1}^{i=N}|p_{i3} - a_{i3}| \tag{4}$$

5 Application of the proposed model: parsing walls in a professional floor plan

The proposed model has many applications. In this section, we show how the proposed model can be applied and extended to professional floor blueprints. A sample professional floor plan is shown in Fig. 4. Professional floor plans contain a lot of extra information such as in-house images, text, etc., which would need to be pre-processed and cleaned before passing on to our model. It would be interesting to see how our model responds to this type of data.

The extra text was removed using the Google cloud vision API [16]. This API, developed by Google provides a state-of-the-art solution to recognizing and understanding images. Some of the key features of this API are detection of printed and handwritten text, faces and celebrity recognition, identification of popular places and logos. Google Cloud vision API is already pre-trained and the API was used directly to detect text.

For other information such as measuring lines, doors, etc., we once again employed the machine learning technique. Due to the variation of the symbols representing the walls, local feature descriptors such as SURF and SIFT did not produce good results. Consequently, a machine learning approach was used to extract the walls from the image. The details of the classification model and the dataset used to train the model are discussed next.

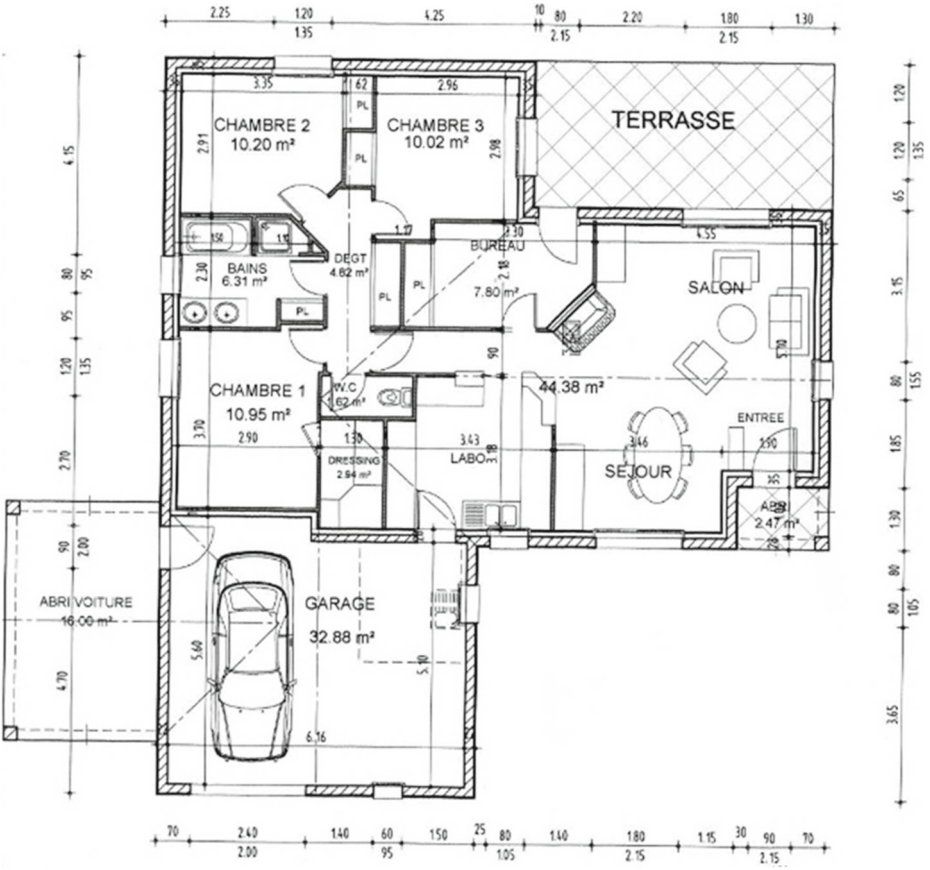

Fig. 4 Sample floor plan (taken from [8])

5.1 Dataset

To train the classifier, we used the public dataset CVC-FP [8] which consists of 122 images of the floor plans. The dataset further has 4 subsets each with a different design and dimensions. One of the subsets had just 4 images which were discarded leaving behind 118 images for training and testing. The dataset contains the actual floor plan and ground truths for the doors, windows, walls, parking and dividers. We only used the ground truths for the walls. Figure 5 shows a sample floor plan with its corresponding ground truth.

Some of the images also contained floor plans for the first floor in addition to the ground floor. In this case, the floor plan for the first floor was removed.

5.2 Fully convolutional neural network

In the proposed framework, fully convolutional networks (FCN) are used to detect and segment the walls. The architecture of FCN is similar to that of VGG-16; in fact the VGG-16 model was adapted for the FCN. As the name suggests, VGG-16 has 16 weight layers (13 convolutional layers and 3 fully connected (FC) layers). In all convolutional layers, the size

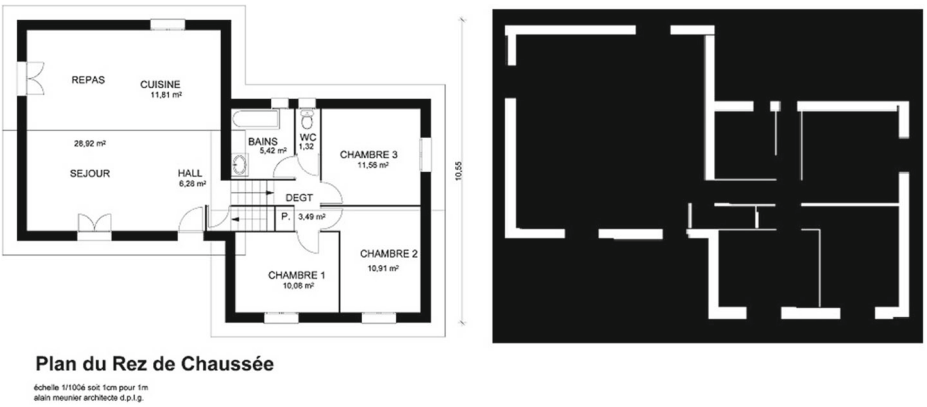

Fig. 5 A sample floor plan along with the ground truth for walls (taken from [8])

of the feature masks is 3x3 with a stride factor of 1. Spatial padding of 1-pixel is done, thus preserving the size. All max-pooling layers have a window size of 2x2 with a stride of 2. The VGG-16 has 3 dense layers including the output layer. The first two layers have 4096 nodes. The last layer that actually depends on the number of classes has 1000 nodes. The dense layers are followed by a softmax layer.

VGG-16 is restricted to images of size 224×224. In our case, the images of the floor plans can be of different sizes. Thus, there is a need of a classifier that is not restricted by the input image size. For this purpose, FCN is used which is similar to VGG-16 however, the two dense layers are replaced by 1×1 convolutional layers. The output from the last convolutional layer is very small and needs to be upsampled which is done by combining the output from shallower layers with the output of the last convolutional layer using skip layers following [26]. Upsampling only the output from the last convolutional layer produces a coarse prediction. Upsampling is done to get an output that has the same size as the input. This allows FCN to classify or infer each pixel of the image. In our case, FCN will take as input a floor plan and infer which pixel is the wall. FCN has been used for image segmentation before such as in [9]. We used FCN with an 8-pixel stride, trained using stochastic gradient descent with a learning rate of 0.01. The training was done for 40 epochs. Such a small number of epochs was a consequence of using a small dataset of only 118 images.

5.3 Evaluation and metrics

To evaluate FCN, 4-fold cross-validation is used. The evaluation metrics are the mean pixel accuracy and mean intersection-over-union (IoU) which have been used earlier for semantic segmentation [26].

Let n_{ww} be the number of wall pixels predicted as wall pixels, n_{bb} be the number of background (non-wall) pixels predicted as background pixels, and let t_w and t_b are the total number of wall and background pixels respectively, in the ground truth, then mean pixel accuracy is given by:

$$\text{mean pixel accuracy} = \frac{1}{2} * (n_{ww}/t_w + n_{bb}/t_b) \tag{5}$$

The mean IoU is given by:

$$\text{mean IoU} = \frac{1}{2} * (\text{wall IoU} + \text{background IoU}) \tag{6}$$

where,

$$\text{wall IoU} = n_{ww} \ / \ \text{union of wall pixels} \tag{7}$$

and,

$$\text{background IoU} = n_{bb} \ / \ \text{union of background pixels} \tag{8}$$

The mean pixel accuracy and mean IoU are calculated for each image and then their mean value is reported.

6 Results - I

In the last section, we discussed the different stages of how both professional floor plans and hand drawn outlines of houses, mazes, etc. can be converted into 3D structures in VR. In this section, we present the results of each stage.

6.1 Doors and windows detection and removal

Professional floor plans contains lot of extra information such as text, furniture etc., as shown in Fig. 4. We therefore use machine learning to recognize walls in the floor plans and extract them from the image. For hand-drawn designs, we use image processing to detect and remove windows and doors leaving us with just the walls. Professional floor plans and hand-drawn designs are therefore discussed separately.

6.1.1 Professional floor plan

FCN is used for extracting/segmenting walls from floor plans. The dataset only had 118 floor plans. Using 4-fold cross validation, we attained a mean-pixel accuracy of 0.97 or 97.1% on wall segmentation using the CVC-FP data set. This is comparable to [9] that has a mean-pixel accuracy of 97.3% but this study uses FCN with different set of hyper parameters. Furthermore, in terms of mean IoU, 94.5% is obtained. Another study that has published the accuracy of their wall detection and segmentation method is [15], that has recognized the walls with a mean-pixel accuracy of 86%.

A sample image inferred using our trained FCN is given in Fig. 6. As seen in Fig. 6 and comparing the results with the original floor plan in Fig. 5, the FCN has quite accurately extracted the walls from the image and removed all extra information such as doors and any remaining text. A threshold is applied to the inferred image to fill out the gray parts of the walls. After that the walls are segmented and a list of points is generated for each wall. This list is later passed to the 3D model generation module.

6.1.2 Hand-drawn designs

In hand-drawn images, the triangles represent walls and rectangles with a cross, represent windows. To extract the walls, first the doors and windows are identified and removed. Then the horizontal and vertical walls are detected whose results are discussed in the next section. Figure 7 shows both a hand-drawn floor plan with two doors and a window, and the

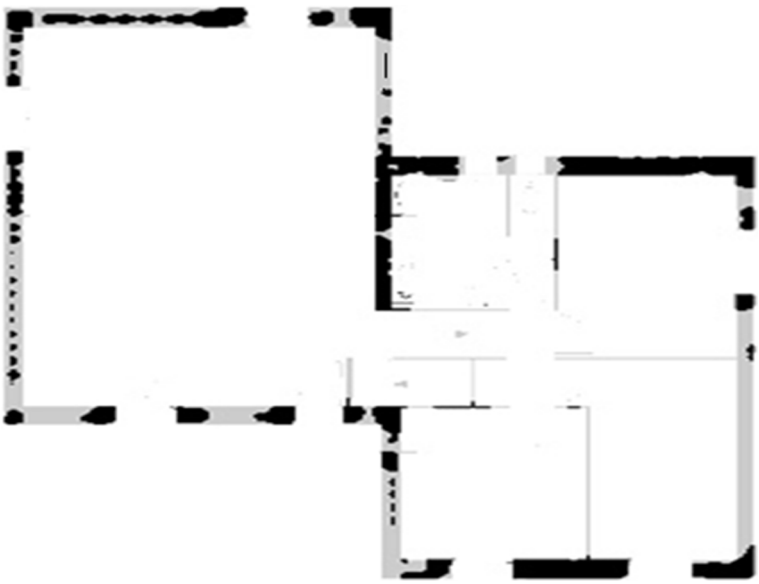

Fig. 6 Results of detecting walls in a professional floor plan using FCN classifier. Original image given in Fig. 5

resulting image. As seen in Fig. 7b, the doors and windows have been correctly detected and removed in the resulting image.

Figure 8 shows another more complicated test map. Figure 8a is the actual hand-drawn design which has multiple doors in close proximity to each other. This image was drawn to test our program for reliability and robustness. It can be seen in Fig. 8b that only the wall structure is left and the doors and a window have been successfully removed. The resulting image will be input to the wall detection and segmentation module.

Figure 8 is interesting since two of the doors have been drawn opposite to each other. Also, one of the door (center-left) has no adjacent walls. The purpose of this is to check if the contour for each door are determined correctly in the output file. As mentioned in the previous section, the first vertex is where the door closes. The second vertex is the hinge and the third vertex shows the direction in which door can be rotated. The coordinates generated for this example in the output file are given below.

4:376 769 495 769 498 683:344 767 341 626 463 624:657 640 661 520 778 524:375 144 485 151 481 291:

These coordinates have been verified by plotting them on a graph. Note the points that correspond to the doors are flipped when compared with the original image (Fig. 9).

Quantitative evaluation For hand-drawn designs, the door and window detection algorithm produced the door-count MAE and window-count MAE of 0. This shows that the image processing component of the proposed framework was able to detect all doors and windows correctly. In the 43 images in the dataset, there were 136 doors and 73 windows. Another evaluation metric that we used was wall-pixel accuracy. Once the doors and windows were removed, we obtained a wall-pixel accuracy of 98.8%. Unlike wall recognition

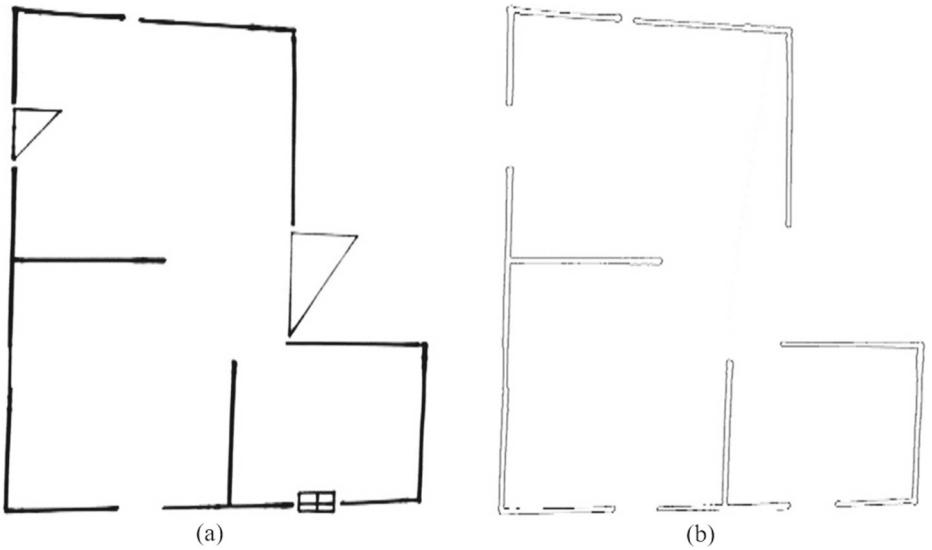

Fig. 7 Detection and Removal of the doors and windows from a hand-drawn floor plan design. **a** is the original image consisting of 2 doors and a window. **b** shows the resulting image with the doors and window removed

algorithms where every pixel in the image is classified as a wall or non-wall pixel, our method recognized walls by subtracting doors and windows. This accounts for such high accuracy.

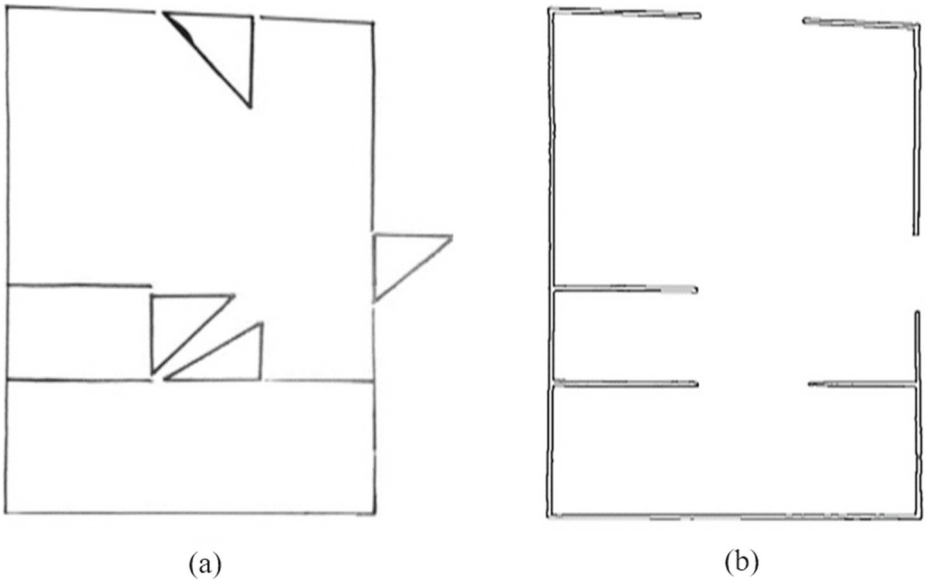

Fig. 8 Another example that demonstrates the door detection and removal module. (**a**) is the original hand-drawn floor plan with 4 doors, while (**b**) shows the resulting image with no door

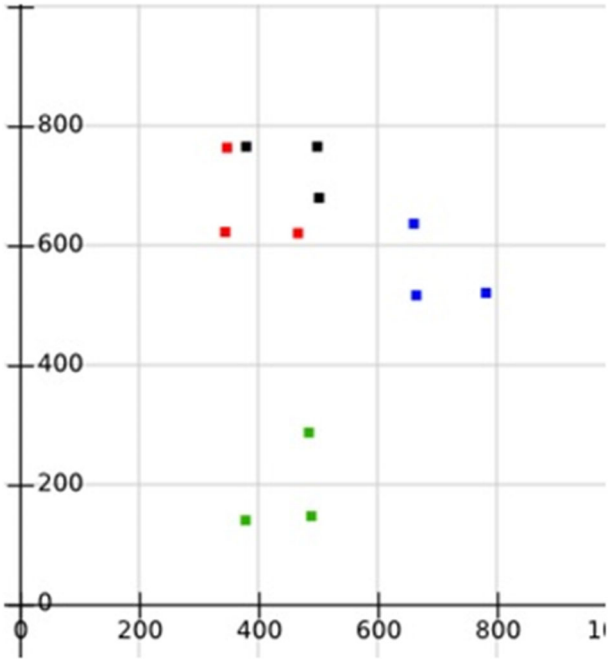

Fig. 9 Graphical user interface - Apply action 'walk'

6.2 Walls segmentation

In professional floor plans, the walls are detected using FCN classifier and the only thing left to do is segmentation. In hand-drawn floor maps, we have so far detected and removed doors/windows. As an extra check, walls will first be extracted and then segmented. In this subsection, we only show the results for hand-drawn images since the same segmentation approach also works for professional floor plans. The output of wall detection and segmentation module is an array of contours for each wall.

Figure 10 shows the results of detecting and segmenting walls of the design given in Fig. 7. The image is not to scale. The vertical walls and horizontal walls have been shown

Fig. 10 Segmentation of horizontal and vertical walls in the hand-drawn map given in Fig. 7a

separately. The output shows that all 7 horizontal walls and 5 verticals walls have been correctly detected and segmented.

Similarly, Fig. 11 show the segmented vertical and horizontal walls of the original hand-drawn design in Fig. 8. The original image has 7 horizontal and 5 vertical walls, and the same number of walls have been detected and segmented. Once again, the results are not shown to scale.

6.2.1 Quantitative evaluation

Our dataset consists of 43 images that has a total of 1013 wall segments. We used the wall-count MAE as the evaluation metric for wall segmentation method. We obtained a wall-count MAE of only 1.2 which is quite promising.

7 Results - II

In the previous section, we demonstrate how both professional floor plans and hand-made designs or maps are processed and get translated into an array of points for doors and windows, and an array of contours for walls. In this section, we examine the results of the VR module with respect to hand-drawn mazes and drawings.

We first begin with a very simple test image. Figure 12 shows the hand drawn image of a long corridor. The resulting model in VR is shown in Fig. 13. In this hand drawn image, and all others that follow, we have inserted a small arrow to indicate the position and direction of the camera with respect to which the VR screen is shown. The arrow is inserted for the sake of readers so that they can better visualize the mapping between the drawings on paper and the 3D model in VR.

A more complex hand drawn maze is shown in Fig. 14. This maze also has a door and a window in addition to different pathways and rooms. Considering the small arrow to indicate the approximate location and direction of the camera, the resulting VR view is shown in Fig. 15.

Finally, Fig. 16 shows another maze with lots of corridors and pathways. A snapshot of one part of the maze in VR is shown in Fig. 17. Note that in this VR model the corridors are very narrow. This is because in the application settings we allow the user to select one of the four styles: huge, wide, middle and narrow. In this example, we test the narrow style. No matter which style is selected the height of the walls remains the same and only the overall spacing of corridors and rooms is scaled.

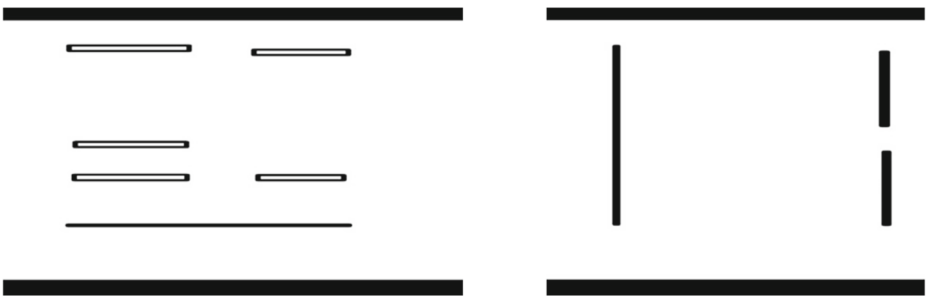

Fig. 11 Segmentation of horizontal and vertical walls in the hand-drawn map given in Fig. 8a

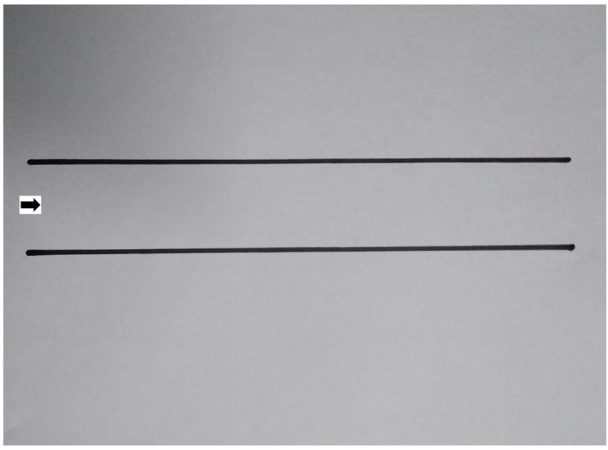

Fig. 12 Hand drawn test image 1. The small arrow indicates the position and direction of view

The user can also customize wall, doors and windows. This is done by looking at the object in focus and using the Bluetooth controller to select various options. Snapshots of various customization options for wall, doors and windows are given in Appendix.

7.1 Subjective evaluation

A small-scale subjective evaluation was required to test how well the framework entertains. For this purpose, an experiment was conducted involving 18 school-going children between the ages of 8-11. Every child was asked to use the application for 20-30 minutes and record their feedback in the form of a questionnaire in the presence of their parent(s), at their home. The parents were mere spectators. Before the experiment, each child was given a brief introduction to VR and its navigational mechanism using a Bluetooth controller. This was followed by instructions on how to convert the test images to VR. None of the children had any prior experience with VR technology.

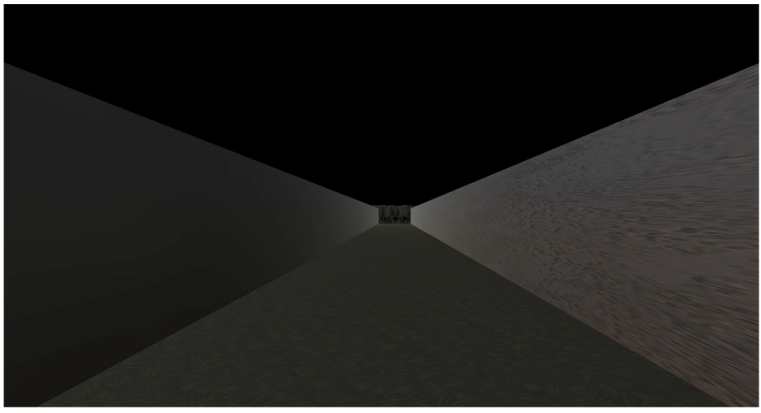

Fig. 13 A look inside the VR model of the hand drawn test image 1

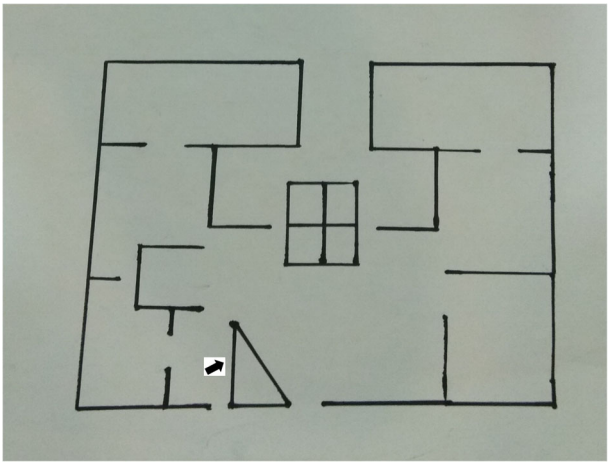

Fig. 14 Hand drawn test image 2

The questionnaire can be divided into two parts. The questions in part one focused on the entertaining aspect. It was an attempt to gather information with regards to how much fun or annoying it was to use the application and their willingness to use it again. Part two of the questionnaire mainly covered the motion sickness aspect as individuals are known to suffer from motion sickness due to VR. The various question/statements given in the questionnaire and their details are given next:

1. How would you rate the application? This question was scored using emotion icons consisting of five scales (excited/pleased, happy, neutral, boredom, frustrated/disgusted) following the pattern given in [37].
2. I am likely to use this application again in the future. This is considered as a reliable way of measuring entertainment according to the PENS model. The response was recorded on a five-point Likert scale where 5 means strongly agree.
3. I enjoyed using the application. Same scale as statement 2 and,

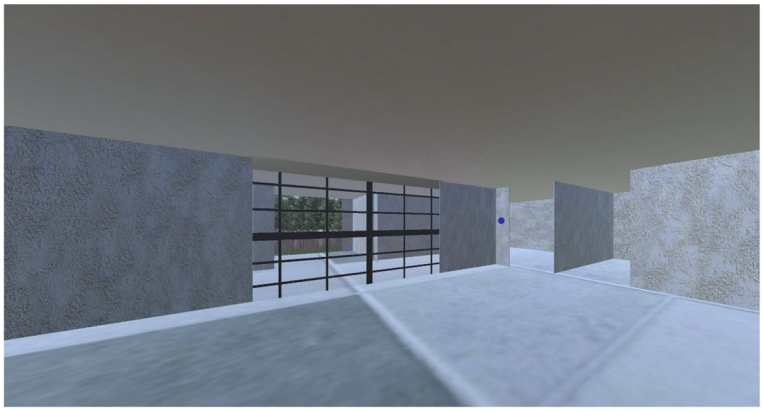

Fig. 15 VR model for test image 2

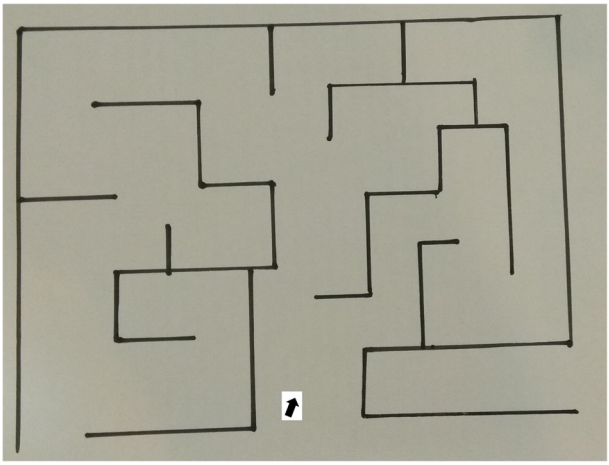

Fig. 16 Hand drawn test image 3

4. I was bored by the application (statement 3 reversed). This strategy was adapted from the intrinsic motivation inventory [38] to analyze the game enjoyment factor. Once again, the statements were scored on a five-point Likert scale where 5 means strongly agree.
5. For the second part: Do you feel any dizziness or nausea or discomfort? A simple yes or no question.
6. To do feel any discomfort in the eyes? Yes/no options only.

The children were free to ask any question in case of ambiguity. The overall rating of the application (question 1) is given in Table 2. It can be seen that almost all the children (14 out of 18) chose excited or pleasure and none of the children chose boredom or frustration. According to Don Norman, one of the best way to measure fun "is observe the people and whether they are enjoying the process". The children were certainly excited and VR was an important factor. Owing to the simplicity of the interfaces, no one was frustrated.

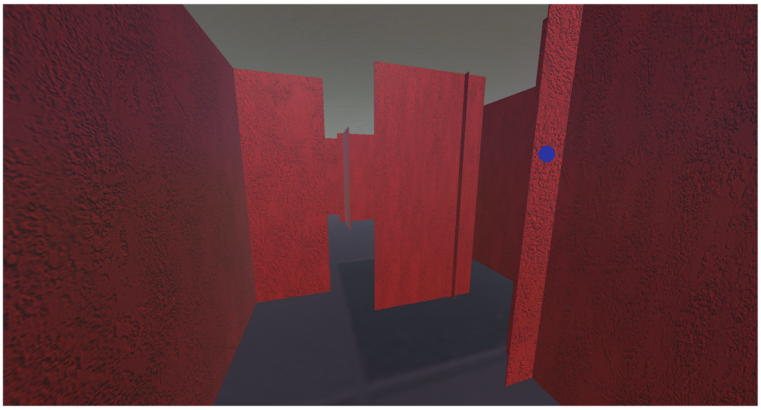

Fig. 17 Peak inside the VR model for test image 3

Table 2 Results for overall rating. Each value indicates the count for a particular mood

	Excited/pleased	Happy	Neutral	Boredom	Frustrated
Rate the application	14	3	1	0	0

Further results on the entertainment and satisfaction are given in Table 3. From Table 3, it can be seen that almost all children expressed their willingness to use the framework again (question 2). In gaming industry, retention is now a major metric for measuring enjoyment [17]. It can also be seen in Table 3 that mostly the children enjoyed the experience (question 3) and they had positive emotions about the application. Finally, the children strongly disagreed with the statement that they were bored (point 4). This is actually the reverse of statement 3, and according to [38], a strategy to analyze the game enjoyment factor. Furthermore, this statement not only showed that the children understood the questions but also showed that they purposefully responded to the questionnaire as they switched from scale of 5 to 1 from question 3 to question 4.

None of the children reported any complaints with respect to motion sickness, headache etc.

7.2 Conclusion

Overall, the results are good. However, there is as issue of extraneous walls. These are extra bit of walls created by the 3D generator and are indicated by blue colored circles in both Figs. 15 and 17. This growth was introduced during the conversion of wall coordinates to the obj file. Besides this issue, all VR models largely captured all the details as shown for the three test maps.

The children expressed that AR itself was also a very pleasing experience. They also enjoyed the process of making their own design. The way how the scene could be zoomed or the view angle could be changed was more natural and easy than using controls on a screen. In conclusion,

8 Conclusion and future direction

In conclusion, considering the rising popularity of VR due to the enjoyment factor that it introduces, we present a framework that allows children and domestic users to create mazes, houses etc., on paper and then navigate them in virtual reality (VR) for entertainment purposes. The user can draw a 2D map of a maze etc. using a simple paper, pen and ruler. The application takes a hand drawn image and then builds a 3D model in VR using Unity3D. The 3D model completely follows the design of the floor plan including the placement of

Table 3 Results for whether the children would like to use the application again and how much of an entertainment was the application

	1(strongly disagree)	2(disagree)	3(neutral)	4(agree)	5(strongly agree)
Use again(Q2)	0	0	3	0	15
Enjoy (Q3)	0	0	3	1	14
Bored (Q4)	15	3	0	0	0

doors and windows. The user can also customize walls, windows, doors and floors. From paper to 3-D model in VR, everything can be done in just one click. The main novelty of our works is constructing a VR model from hand drawn images. We also show how our framework can be extended to professional home blueprints and floor plans. We have shown through various examples and test maps, the working and output of our framework. We used fully convolutional network to extract walls from the professional floor plans with mean pixel accuracy of 97.3%. For hand-drawn designs, we were able to detect doors and windows with MAE of 0. Our wall segmentation method reported an MAE of only 1.2. Based on subjective evaluation, we have shown that framework which was designed for entertainment, is entertaining based on the response to the statement "I enjoyed using the application".

One of the main challenge that we faced was the non-uniform shapes of the doors and windows drawn by the user. All these problems were properly handled to make our framework more robust and open to all kinds of inputs. The other challenge was converting the coordinates in a text file to a 3D model in Unity3D. This was resolved by converting the coordinates of walls, doors and windows in a file and handling them in Unity3D through scripting.

For future work, we will also like to add stairways. This would allow user to create multistory mazes and houses. Another extension to this framework is multiplayer. This would allow 2 or more children to navigate the same maze online and interact with one another. Also, the lines in the scanned image may not always be parallel. This effects the 3D model in the sense that the walls would appear distorted. For future work, we would like to resolve this issue through perspective correction. Finally, we would like to include more complex shapes such as curves and ellipses.

Appendix: Snapshot of customization options for walls, doors and windows

Fig. 18 Customization menu for walls and floor

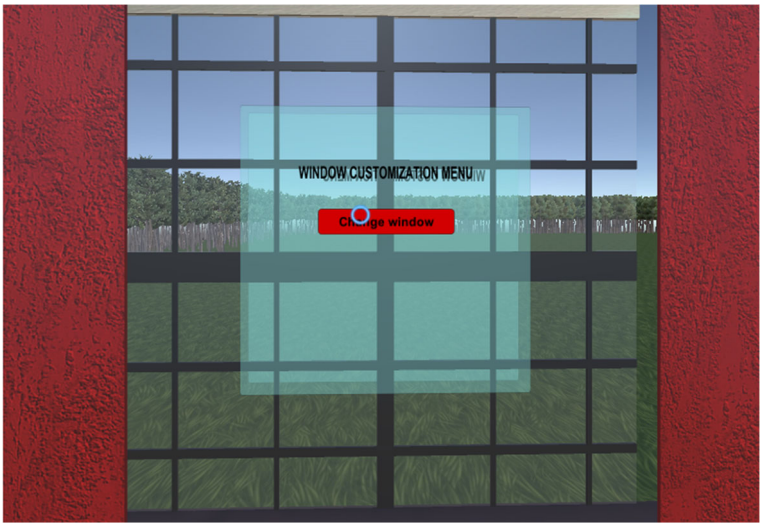

Fig. 19 Customization menu for windows

Fig. 20 Customization menu for doors

Acknowledgements We would like to thank our undergraduate students Hamza Khan, Hafsah Shehzad, Khadija Asim, Arham Fatima for searching and installing 3D modeling software discussed in Table 1 and assisting in their analysis.

Funding No funding was received for conducting this study.

Declarations

Ethics approval and consent to participate This statement is to certify that the author list is correct. The Authors also confirm that this research has not been published previously and that it is not under consideration for publication elsewhere. On behalf of all Co-Authors, the Corresponding Author shall bear full responsibility for the submission.

All procedures performed in studies involving human participants were in accordance with the 1964 Helsinki Declaration and its later amendments or comparable ethical standards. Informed consent was obtained from the parents of all individual participants involved in the study. All subjects gave written informed consent.

Competing interests The authors declare that they have no competing financial interests or personal relationships that could influence the work reported in this paper.

Conflict of Interests The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Alias Systems Corporation (2020) Maya, [Online]. Available: https://www.autodesk.com/products/maya. Accessed 15 Dec 2020
2. Ahmed S, Liwicki M, Weber M, Dengel A (2011) Improved automatic analysis of architectural floor plans. In: 2011 International conference on document analysis and recognition. IEEE, pp 864–869
3. Autodesk, Inc. (2020) 3ds max, [Online]. Available: https://www.autodesk.com/products/3ds-max. Accessed 15 Dec 2020
4. Autodeskcom (2022) Inventor software. https://www.autodesk.com/products/inventor/overview. Accessed 19 Feb 2022
5. Bukowski R (1995) The walkthru editor: towards realistic and e ective interaction with virtual building environments. University of California, Berkeley
6. Chung TM (2018) Strategies for vr prototyping, [Online]. Available: https://medium.com/inborn-experience/strategies-for-vr-prototyping-810e0d3aa21d. Accessed 15 Dec 2020
7. Dalal S, Vishwakarma VP, Kumar S (2020) Feature-based sketch-photo matching for face recognition. Procedia Comput Sci 167:562–570
8. de las Heras L-P, Terrades OR, Robles S, Sánchez G (2015) Cvc-fp and sgt: a new database for structural floor plan analysis and its groundtruthing tool. Int J Doc Anal Recogn (IJDAR) 18(1):15–30
9. Dodge S, Xu J, Stenger B (2017) Parsing floor plan images 05:358–361
10. Dosch P, Tombre K, Ah-Soon C, Masini G (2000) A complete system for the analysis of architectural drawings. IJDAR 3:102–116
11. Exxar (2022) Experience cad to reality. https://exxar.cloud/cad-to-vr/. Accessed 19 Feb 2022
12. Fritz M (2018) Ar/vr prototyping: take designs from paper to glasses, [Online]. Available: https://medium.com/inborn-experience/ar-vr-prototyping-cd765bad650f. Accessed 15 Dec 2020
13. Gai W, Yang C, Bian Y, Dong M, Liu J, Dong Y, Niu C, Lin C, Meng X, Shen C (2016) Ubimaze: a new genre of virtual reality game based on mobile devices. In: Proceedings of the 18th international conference on human-computer interaction with mobile devices and services adjunct, pp 591–593
14. Gai W, Yang C, Bian Y, Shen C, Meng X, Wang L, Liu J, Dong M, Niu C, Lin C (2017) Supporting easy physical-to-virtual creation of mobile vr maze games: a new genre. In: Proceedings of the 2017 CHI conference on human factors in computing systems, pp 5016–5028
15. Gimenez L, Robert S, Suard F, Zreik K (2016) Automatic reconstruction of 3d building models from scanned 2d floor plans. Autom Constr 63:48–56
16. Google Cloud Vision API (2019) Vision ai derive image insights via ml cloud vision api. https://cloud.google.com/vision. Accessed 16 May 2020
17. Halim Z, Baig AR, Mujtaba H (2010) Measuring entertainment and automatic generation of entertaining games. Int J Inf Technol Commun Converg 1(1):92–107
18. Horna S, Damiand G, Meneveaux D, Bertrand Y (2007) Building 3d indoor scenes topology from 2d architectural plans. In: GRAPP, pp 37–44

19. Kamińska D, Sapiński T, Wiak S, Tikk T, Haamer RE, Avots E, Helmi A, Ozcinar C, Anbarjafari G (2019) Virtual reality and its applications in education: survey. Information 10(10):318
20. Kamppari-Miller S (2018) Vr paper prototyping, [Online]. Available: https://blog.prototypr.io/vr-paper-prototyping-9e1cab6a75f3. Accessed 15 Dec 2020
21. Kamppari-Miller S (2018) Vr sketch sheets & paper prototyping, [Online]. Available: https://medium.com/designer-geeking/vr-sketch-sheets-paper-prototyping-859d3b7300e4. Accessed 15 Dec 2020
22. Kurbatov V (2017) Draw sketches for virtual reality like a pro, [Online]. Available: https://medium.com/inborn-experience/vr-sketches-56599f99b357. Accessed 15 Dec 2020
23. Kurbatov V (2019) Templates for ar/vr sketches, [Online]. Available: https://medium.com/inborn-experience/templates-for-ar-vr-sketches-e424dfb60e54. Accessed 15 Dec 2020
24. Lewis R, Séquin C (1998) Generation of 3d building models from 2d architectural plans. Comput Aided Des 30(10):765–779
25. Liu C, Wu J, Kohli P, Furukawa Y (2017) Raster-to-vector: revisiting floorplan transformation, pp 2214–2222
26. Long J, Shelhamer E, Darrell T (2015) Fully convolutional models for semantic segmentation. In: CVPR, vol 3, p 4
27. Maxon Computer (2018) Sketchup, [Online]. Available: https://www.maxon.net/en/cinema-4d
28. McIntosh J, Rodgers M, Marques B, Gibbard A (2019) The use of vr for creating therapeutic environments for the health and wellbeing of military personnel, their families and their communities. J Dig Landsc Archit 1(4):185–195. Accessed 15 Dec 2020
29. Nebeling M, Madier K (2019) 60proto: making interactive virtual reality & augmented reality prototypes from paper. In: Proceedings of the 2019 CHI conference on human factors in computing systems, pp 1–13
30. Noor M, Nazir M, Rehman S, Tariq J (2021) Sketch-recognition using pre-trained model
31. OpenCV (2017) Open contour. https://docs.opencv.org/3.4/df/d0d/tutorialfindcontours.html
32. OpenCV (2017) Convex hull. https://docs.opencv.org/3.4/d7/d1d/tutorialhull.html. Accessed 15 Dec 2020
33. OpenCV (2017) Approx polydp. https://docs.opencv.org/3.4/jscontourfeaturesapproxPolyDP.html. Accessed 15 Dec 2020
34. OpenCV (2017) Bounding rect. https://docs.opencv.org/3.4/da/d0c/tutorialboundingrect.html. Accessed 15 Dec 2020
35. Planner5D (2016) Planner 5d - home and interior design creator. https://planner5d.com/. Accessed 15 Dec 2020
36. Racz A, Zilizi G (2018) Vr aided architecture and interior design. In: 2018 International conference on advances in computing and communication engineering (ICACCE). IEEE, pp 11–16
37. Rieffe C, Oosterveld P, Terwogt MM (2006) An alexithymia questionnaire for children: factorial and concurrent validation results. Personal Individ Differ 40(1):123–133
38. Ryan RM, Mims V, Koestner R (1983) Relation of reward contingency and interpersonal context to intrinsic motivation: a review and test using cognitive evaluation theory. J Person Soc Psychol 45(4):736
39. Software SDI (2022) Nx virtual reality | siemens software. https://www.plm.automation.siemens.com/global/en/products/mechanical-design/nx-virtual-reality.html. Accessed 19 Feb 2022
40. Swaileh W, Kotzinos D, Ghosh S, Jordan M, Vu NS, Qian Y (2021) Versailles-fp dataset: wall detection in ancient floor plans. In: International conference on document analysis and recognition. Springer, pp 34–49
41. Sweetser P, Rogalewicz Z, Li Q (2019) Understanding enjoyment in vr games with gameflow. In: 25th ACM symposium on virtual reality software and technology, pp 1–2
42. Trimble, Inc. (2020) Sketchup, [Online]. Available: https://www.sketchup.com/. Accessed 15 Dec 2020
43. Trimble, Inc. (2000) Sketchup, [Online]. Available: https://www.autodesk.com/products/revit. Accessed 15 Dec 2020
44. Ton Roosendaal (2017) Blender, [Online]. Available: https://www.blender.org/. Accessed 15 Dec 2020
45. Vidil F, Damiand C (2003) Moka, [Online]. Available: www.sic.sp2mi.univpoitiers.fr/moka/. Accessed 15 Dec 2020
46. Wang W, Dong S, Zou K, Li W (2020) Room classification in floor plan recognition. In: 2020 4th International conference on advances in image processing, pp 48–54
47. Wang W, Dong S, Zou K, Li WS (2020) Room classification in floor plan recognition; room classification in floor plan recognition 2020. https://doi.org/10.1145/3441250.3441265
48. Xu Z, Yang C, Alheejawi S, Jha N, Mehadi S, Mandal M (2021) Floor plan semantic segmentation using deep learning with boundary attention aggregated mechanism, pp 346–353

49. Yan R, Masood A, Li P, Ali SG, Sheng B, Ren J (2018) 3d simulation of interior house design in vr using vr3id method. In: 2018 IEEE International conference on progress in informatics and computing (PIC). IEEE, pp 236–240
50. Yamasaki T, Zhang J, Takada Y (2018) Apartment structure estimation using fully convolutional networks and graph model, pp 1–6
51. Zhu J, Zhang H, Wen Y (2014) A new reconstruction method for 3d buildings from 2d vector floor plan. Comput-aided Des Applic 11(6):704–714
52. Zhang X, Huang Y, Zou Q, Pei Y, Zhang R, Wang S (2020) A hybrid convolutional neural network for sketch recognition. Pattern Recogn Lett 130:73–82