Declarative Image Generation

Anitej Palakodeti

TU Delft

Declarative Image Generation

by

Anitej Palakodeti

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday August 24, 2022 at 10:00 AM.

Student number:	5214068	
Project duration:	November 24, 2021 – August 24, 2022	
Thesis committee:	Prof. dr. G.J.P.M. Houben,	TU Delft, Chair
	Dr. J. Yang,	TU Delft, Supervisor
	Dr. A. Katsifodimos,	TU Delft, Supervisor
	Dr. M. Weinmann,	TU Delft, Graduation Committee

This thesis is confidential and cannot be made public until August 24, 2022.

TU Delft

Preface

I present to you my Master thesis on Declarative Image Generation. This thesis is a culmination of nine months of research, planning and development which has indeed enriched my experience and has given me great insight into academia.

I would like to thank Dr. Jie Yang and Dr. Asterios Katsifodimos for introducing to me this thesis topic and for their guidance and supervision throughout the time I worked on this thesis. I am very grateful to Agathe Balayn and Ziyu Li for their constant advice, ideas and feedback. This helped me stay on the right track while working on my thesis and their assistance ensured that I could produce high-quality work. I would also like to thank Prof. Dr. Geert-Jan Houben for his detailed feedback and guidance, and Dr. Michael Weinmann for agreeing to be part of my thesis committee.

I am thankful to my peers studying Computer Science at TU Delft as our interactions and discussions have helped broaden my views of the possibilities within the field. Lastly, I would like to thank my parents, brother and friends for motivating me during the two years spent pursuing this Master's degree. I appreciate all their support and encouragement.

Anitej Palakodeti
Delft, August 24, 2022

Abstract

Generating synthetic images has wide applications in several fields such as creating datasets for machine learning or using these images to investigate the behaviour of machine learning models. An essential requirement when generating images is to control aspects such as the entities or objects in the image. Controlling this helps in creating custom datasets tailored for the above applications and creates a platform to conduct diverse experiments with the generated images, enabling research in the application's field. Existing methods individually enable controllability over various elements in the image such as selecting the objects, their properties, colour or the relations between objects, etc. but we identify a research gap in this field where no single method allows the user to control all these aspects of the image. An additional research gap identified is that existing methods cannot generate images based on a query with a logic based combination of entities.

In this thesis, we aim to fill this research gap by developing SceneUI - a system that allows the user to specify and control aspects of the scene through a user interface such as the objects, object properties, spatial relations between objects, object colour, extent of contextual objects and the background of the image. Additionally, we include a component where the scene is generated based on an *OR* query specifying the objects as predicates, which serves as a foundation to generating images based on entity combinations. Owing to the limited range of attributes for objects and the lack of objects with attributes in the dataset, we augment the dataset by expanding the attributes of objects in the scene graphs of the dataset and introducing additional objects that have attributes. This was done by identifying recurring objects in the dataset that could be expanded and manually annotating the changes in the dataset. This increases the level of controllability and gives a wider range of an object's properties to choose from.

The goal of the thesis is to design and develop a method that allows the user to declare, specify and manipulate elements of the image and eventually use the images generated for two use cases - *Generating Images for Interpretable Machine Learning* and *Generating Images from Queries as Ground Truth*. We evaluate SceneUI to show its usability and effectiveness for the two use cases through two experiments. In the first experiment, we use SceneUI to create biased datasets where each bias is based on an object's colour or object type and the goal is to train a deep learning model that learns the biases. The results show that the model learns the biases well and thus, SceneUI can be used to control datasets which can be used to benchmark machine learning explainability methods. The second experiment generates a dataset of images using the *OR* query and training machine learning models on the dataset to ensure the suitability of the images for machine learning tasks. As the generated images will be used as ground truth given a query for a specialised machine learning model, the model is expected to identify the predicate objects in the image. The results show SceneUI can generate images based on objects in the query and can also accommodate the objects to have properties. The models identify important features in the SceneUI-generated images and are thus suitable to be used as ground truth. We also discuss the limitations and tradeoffs of SceneUI and present potential future directions for research to improve its scalability.

Contents

1 Introduction **1**
- 1.1 Thesis Goal . . . 2
- 1.2 Research Questions . . . 3
- 1.3 Thesis Challenges . . . 3
- 1.4 Contributions . . . 4
- 1.5 Thesis Outline . . . 4

2 Background **5**
- 2.1 Machine Learning For Image Generation . . . 5
- 2.2 Computer Graphics For Image Generation . . . 6
  - 2.2.1 Generating Synthetic Datasets . . . 6
  - 2.2.2 Dataset Generating Software . . . 6
  - 2.2.3 Domain Randomisation . . . 7
- 2.3 Indoor Scene Synthesis . . . 7
  - 2.3.1 Text to 3D Scene . . . 8
  - 2.3.2 Semantic Scene Graphs . . . 9
  - 2.3.3 Including Contextual Objects . . . 10
- 2.4 Use Cases . . . 10
  - 2.4.1 Generating Images for Interpretable Machine Learning . . . 11
  - 2.4.2 Generating Images from Queries as Ground Truth . . . 11
- 2.5 Literature Analysis & Research Gaps . . . 12

3 SceneUI **15**
- 3.1 System Overview . . . 15
- 3.2 Dataset . . . 17
  - 3.2.1 Semantic Scene Graphs . . . 17
- 3.3 Expanding Attributes . . . 18
  - 3.3.1 Method to Expand Attributes . . . 19
  - 3.3.2 Expanded Attributes and Objects . . . 20
- 3.4 Graph Synthesis . . . 20
- 3.5 Graph Matching . . . 21
  - 3.5.1 Graph Alignment . . . 21
  - 3.5.2 Subgraph Augmentation . . . 22
- 3.6 Scene Creation . . . 23
  - 3.6.1 Generating Subgraph Scene Nodes . . . 23
  - 3.6.2 Rendering Augmented Nodes . . . 24
- 3.7 User Interface . . . 25
  - 3.7.1 Tables . . . 26
  - 3.7.2 Lists . . . 27
  - 3.7.3 Parameters . . . 27
  - 3.7.4 Comparison with Original Work . . . 27
- 3.8 Query Generated Scene . . . 29

4 Experiments **31**
- 4.1 Bias Detection . . . 31
  - 4.1.1 Types of Biases . . . 32
  - 4.1.2 Generating Causal Graphs . . . 32
  - 4.1.3 Generating Images . . . 34
  - 4.1.4 Fine-tuning the Neural Network . . . 36

4.2 Query Generated Images . . . 37
4.2.1 Generating Images . . . 37
4.2.2 Image Classification . . . 38

5 Results **39**
5.1 Bias Detection . . . 39
5.2 Query Generated Images . . . 43

6 Discussions & Conclusions **45**
6.1 Conclusions . . . 45
6.1.1 Summary . . . 45
6.1.2 Discussion and Comparison with Literature . . . 47
6.2 Limitations . . . 47
6.3 Future Work . . . 48

A Appendix - Dataset Training Images **49**
A.1 Biased Dining Room Images . . . 49
A.2 Biased Study Room Images . . . 50
A.3 Biased Living Room Images . . . 51
A.4 Natural Training Images . . . 52
A.5 Images with only Chair . . . 53
A.6 Images with only Table . . . 54
A.7 Images with Chair and Table . . . 55
A.8 Images without Chair and Table . . . 56

List of Figures

1.1 Images generated by altering Attributes of objects in a study room. 2

2.1 Scene Graph example for the description - There are books stacked on the desk. The blue nodes denote objects, green node the spatial relation and yellow node the property [34]. 9
2.2 Scene Graph example for an elaborate description - The round dining table is surrounded by 3 chairs and there is a flower vase on top of the table. A more complicated description results in an intricate scene graph [34]. 9
2.3 Change in scene with more objects such as speakers and chair on increasing the context parameter [34]. 10

3.1 System Overview of the Original Work. The user provides a text description of the scene which is converted to a scene graph. Subsequently, similar graphs are identified and an image is generated. 16
3.2 System Overview of SceneUI. The key difference is the User Interface which allows the user to select the quantity, attributes and relations of objects along with an *OR* query. The interface is supported by a dataset where objects are augmented to have more attributes. SceneUI also enables the image to have a background. 16
3.3 Scene graph for a scene with an office chair in front of desk, with the desk on the right of the queen sized bed. 18
3.4 Generating Tables with different colours selected from the expanded attributes. 20
3.5 Scene graph UI_g synthesised from the input. 22
3.6 Scene Graph D_g from the dataset with aligned nodes that match with UI_g highlighted. . 22
3.7 Augmented Graph S_g with contextual nodes highlighted. 23
3.8 Final image generated from the description and input scene graph. 25
3.9 User Interface designed with tables (1) to display and input information, lists (2) to display objects and select attributes, and parameters (3) to further tune and control the setting. 26
3.10 User Interface describing a scene with an Office Chair, Desk, and a Queen Bed. 28
3.11 Process for generating an image given an *OR* query. Changes are made to the Scene Objects list and the Relations table based on the key value. 29

4.1 Biased Study Room with a speaker and monitor on the desk. 33
4.2 Biased Dining Room with a red dining table. 33
4.3 Biased Living Room with a glass coffee table and a 3 shelf bookcase. 33
4.4 Causal Graph for a Biased Study Room. 33
4.5 Examples of images used for different classes of the three datasets. 35
4.6 Saliency map generated for an image. The highlighted region in the saliency map denotes attributes or areas the image finds important for classification. 36
4.7 Examples of images generated for all scenarios based on objects specified as a query. 38

5.1 Normalised Confusion matrices for Biased and Unbiased models. 40
5.2 Images and respective saliency maps when predicting a Study Room with the biased model. 40
5.3 Images and respective saliency maps when predicting a Dining Room with the biased model. 41
5.4 Images and respective saliency maps when predicting a Living Room with the biased model. 42
5.5 ROC plots for various SVM kernels used in the experiments. 44

A.1 Biased images from the Dining Room class. 49
A.2 Biased images from the Study Room class. 50
A.3 Biased images from the Living Room class. 51
A.4 Unbiased images of all classes also used for training. 52
A.5 Images containing a chair from the *OR* class. 53
A.6 Images containing a table from the *OR* class. 54
A.7 Images containing a chair and table from the *OR* class. 55
A.8 Images from the *Not* class which do not contain a chair or a table. 56

List of Tables

2.1 Overview of literature pertaining to Indoor Scene Synthesis and its features. 12

3.1 Attributes of objects in the scene graphs in the dataset. 18
3.2 Expanded attributes of existing and additional objects in the scene graph dataset. . . . 19

4.1 Categorisation of the biases, biased objects and properties included for each class. . . 33
4.2 Attributes of objects selected for each class with the user interface. 34
4.3 Parameters for SVM kernels using GridSearch. 38

5.1 Overview of the misclassifications cause due to the presence of biased objects or properties. 43
5.2 Performance of different SVM kernels over 4 metrics. 44

1

Introduction

There is an increasing trend of using deep learning models to solve complex problems [30] in research areas ranging from speech translation [47] to using computer vision models for medical image analysis [36]. Though these models give great results with high accuracy, they come with their drawbacks as they can be difficult to debug and comprehend their decision-making procedure [46] which results in hindrances to using them in real life. Thus, as deep learning models are complex black-box models, they require extensive efforts by researchers to understand their behaviour. To design methods that explain a model's behaviour we need to know its true nature of behaviour. Once this is known, the behaviours can be introduced into models by creating biased datasets where the images of a class are biased towards the presence of a particular object, object property or colour. Acquiring relevant datasets with images that contain specific features is tedious and expensive [53]. Thus, generating synthetic images which contain specific objects, or objects with a certain colour is a viable alternative, especially if the features expected in the image can be controlled. These synthetic images can also be used in the training set for various machine learning models where these images will act as the ground truth for a given class.

Image generation has captured the interest of Computer Science researchers for several years and it is now a growing field where the methods for creating realistic images are developing at a rapid pace [42, 41, 13]. The images generated at scale have a wide variety of applications in several fields and the images generated could be based either on an outdoor or indoor setting. Indoor scene generation is a domain within image generation where the scope of the generated content is narrowed down to rooms with appropriate furniture and layouts [60]. The scene is a 3D rendered layout of objects where the camera angle of the view can be changed if needed and an image extracted. Given the constrained space of a room, indoor scene generation methods allow a great extent of controllability in selecting objects based on their type or property, placing them at particular positions with respect to other objects, or selecting the background of the room in which the objects are placed. In this thesis, we restrict the scope of our thesis to indoor image generation due to the limited number of objects and the extent of controllability enabled.

Existing methods in indoor scene generation provide for this controllability through the means of text-to-scene synthesis [5, 7, 8, 34]. These methods make the user declare and specify the objects and the number of instances expected in the scene, the spatial relations between objects, and object properties. These methods achieve the generation of the scene in different ways and each method enables the user to control only certain elements of the scene. Therefore, there is no singular method that allows the user to control all of the following elements - selection of objects, the quantity of objects, positioning of objects with respect to each other, object with functionality, the colour of objects, and the background of the scene. These methods also have a limited range of attributes for an object to select from. Another drawback of the existing methods that request a user to specify the scene is the high cognitive load on the user. Due to the user providing only a text input, they are expected to remember all the possible objects that exist in the scene, the properties that the method allows them to select as well as the relations permitted for image generation.

1.1. Thesis Goal

The goal of the thesis is to develop a method that allows the generation of indoor images based on the specification provided by the user. The method also follows an *object-oriented* approach which is based on designing classes with specific functions and calling objects from these class to process information using the functions. In this way, the the decision-making of the method is traceable and easy to understand. Through this method, the user should also be able to control various aspects of the image by specifying the objects and their quantity, attributes or colour of objects, relations between objects, and the background of the image. The image generation process is also expected to consider these specifications as constraints for the image generation problem.

This is achieved by the means of creating an interactive user interface that allows the user to control all aspects of the scene earlier described. The user interface is merged with an existing work that translates natural language to an indoor scene. The natural language component is replaced with the interface, and further changes are made to increase the level of controllability. We name this method that generates images or scenes via a user interface as **SceneUI**, and our method will be referred to by this in the report. The thesis explores using this controllable image generation method in the context of generating images for the following two use cases:

- ***Generating Images for Interpretable Machine Learning***
  The synthetic images generated will be used to benchmark interpretability and explainability methods in Deep Learning models for Computer Vision. The images will be used to train and evaluate computer vision models to verify whether explainability methods can reveal known model behaviours such as using human interpretable concepts like object shapes, colours, and textures during classification. The known behaviours are introduced in the model by creating biased image datasets and training the models on the biased datasets. Essentially, the images will be used to debug and understand deep learning models by investigating whether models learn certain biases from the dataset.

- ***Generating Images from Queries as Ground Truth***
  Specialised machine learning models are a type of neural network which can be used to identify entities in an image or retrieve images given a query with the entities as predicates. Based on the query, images with appropriate entities are retrieved. Training such models require ground truth images with entities based on the possible queries. Thus, there is a need to generate images based on a query to subsequently use them as ground truth. The specification of the image generated should contain a query with predicates as the entities expected in the images. For example, if the query is a Chair *AND* Stool, both objects should be in the image. Alternatively, for a more complex query - Chair *OR* Stool, there should be an instance of the image where only one of the two is present or both objects are present. The images synthesised will serve as ground truth for training a collection of specialised machine learning models.

(a) Study Room with an Office Chair.

(b) Study Room with a Dining Chair.

Figure 1.1: Images generated by altering Attributes of objects in a study room.

To further elaborate on the concept of controllability that is expected of this method, an example is shown in Figure 1.1 which depicts two scenes of a study room. The common scenario for both images in the scene is to have a desk with a chair in front of it and a trash can on the right of the desk. Additionally, a monitor, keyboard, mouse and two speakers are placed on the table. On observing the main differences between the two figures, we can see that Figure 1.1a constitutes a dark brown study desk with a black office chair in a brown room, and Figure 1.1b depicts a light brown study desk and a light brown dining chair in a green room. With the image generation method developed through the thesis (SceneUI), the user should be able to recreate both scenes. The user can specify all the objects occurring in either image, place them at particular locations, and also select the colours of objects.

In the context of the use cases, the images could be used to explore whether a deep learning model considers the type of chair (Office or Dining) or the colour of the objects in the scene important for classifying a study room. For the second use case, if we provide the query of an office chair *Or* a dining chair, with the remaining objects, we could expect either of these images.

1.2. Research Questions

Given the motivation of generating images for the use cases, we want to develop a method that enables the user to synthesise images. While generating images, the method should also enable the user to control and select the objects occurring in it as well as the object's attributes and positions in the scene. We define the following research question to meet this objective :

- ***RQ:*** How to build a system that generates indoor scene images where the object properties, colour and spatial relations between objects can be controlled?

Answering this research question requires additional research sub-questions to be answered:

- ***RSQ1:*** *What are the current methods of image generation and what features of the synthesised image do they allow control over?*
  Answering this question involves conducting a literature study of image generation methods. The study also narrows down to indoor scene generation and discusses the controllable aspects of the methods.
- ***RSQ2:*** *How can the attributes of objects be expanded to allow controllability over multiple properties?*
  Current methods allow controlling none or only limited properties of an object during image generation. Solving this question would require augmenting the dataset with annotations such that the methods can recognise an object's additional property based on its colour or functionality.
- ***RSQ3:*** *How can the user provide specifications for all the controllable variables in the image?*
  Answering this question involves developing an interactive user interface which enables the user to select the objects and specify all relations and properties of the objects. This interface which takes the user's constraints is be merged well with the image generating algorithm.
- ***RSQ4:*** *How can images be generated given a query with objects as predicate terms?*
  To answer this, the method should include a capability for the user to provide a query with objects as query predicates and synthesise an image based on this.
- ***RSQ5:*** *To what extent can the method be used for the two use cases?*
  Answering this question requires conducting experiments using the images generated for scenarios with both use cases. The goal of the experiments would be to evaluate the suitability of the image generating method for the use cases.

1.3. Thesis Challenges

Answering the research questions and subsequent research sub-questions presented above come with their corresponding challenges.

- ***RSQ2:*** A significant challenge is increasing the level of controllability by making changes to an existing method which is selected as the baseline. As this method allows controllability over only certain features, we would have to include the remaining features of controllability such as colour and selecting a background for the objects.

- ***RSQ3:*** The second challenge relates to having a single method that consolidates all controllable features objects may have. This challenge is tackled by designing the user interface that provides functionalities for the user to select the objects, their properties and relations. Designing this user interface also comes with the additional technical hurdle of ensuring it merges with the main scene generating algorithm. Steps were taken to ensure that the input from the interface was processed and collected in a way that allows conversion of the input to a required graph structure.
- ***RSQ5:*** Another challenge that could be encountered is the evaluation of SceneUI in the context of the use cases. Existing literature evaluates similar methods based on the plausibility and realistic look of the images [5, 7, 34, 8, 17]. In this thesis, we aim to evaluate SceneUI via downstream tasks to investigate the suitability of the methods. The challenge here arises due to a lack of literature that could be referred to evaluate the method in this setting. To perform an accurate evaluation of the method with regards to its suitability, we design experiments that reflect scenarios in which the method would be used for both use cases.

1.4. Contributions

By answering the above research sub-questions and tackling the challenges, we aim to provide the following contributions through the thesis:

- ***C1:*** We contribute in the literature study an analysis of indoor scene generation methods which compares aspects of the scene the methods allow control over.
- ***C2:*** We enhance the level of controllability by expanding and augmenting existing attributes of objects in the scene graphs of the dataset.
- ***C3:*** We design a user interface through which the user can describe a scene by specifying objects, object properties, spatial relations between objects, and selection of the background. This user interface is merged with an existing method that generates scenes.
- ***C4:*** We include a feature in SceneUI where the user can provide a query with objects and the resultant image includes objects based on the query.
- ***C5:*** We evaluate the suitability of the SceneUI and its features in the context of two use cases.

1.5. Thesis Outline

The structure of the thesis report is designed in a way to give the reader sufficient background of the problem and existing work, before diving deep to the main method of SceneUI and the experiments while also describing the motivations behind the choices made.

- In chapter 2 we provide background information about existing methods and techniques for image generation using machine learning and computer graphics. We also discuss methods for indoor scene generation and provide an analysis of the existing methods, thereby answering RSQ1.
- In chapter 3 we discuss the workings of SceneUI by giving an overview of the system and the pipeline for information flow. We also describe the dataset used, and various stages of the pipeline. Additionally, we describe how the dataset was manually augmented, the design and functionality of the user interface, and the generation of images based on queries. This chapter collectively answers RSQ2 in section 3.3, RSQ3 in section 3.7, and RSQ4 in section 3.8.
- In chapter 4 we describe the experimental setup to evaluate SceneUI and its suitability for the two use cases.
- In chapter 5 the results and findings from the experiments conducted in chapter 4 are discussed. The answer to RSQ5 is collectively given in chapter 4 and chapter 5.
- In chapter 6 we summarise the thesis and provide answers to research questions. Addtionally, we compare SceneUI to literature and discuss its limitations and directions for future work.

2

Background

This chapter provides information and context about image generation methods. These methods are classified under **Machine Learning** or **Computer Graphics** as approaches for image generation and are discussed in section 2.1 and section 2.2 respectively. Machine Learning methods learn the features of the input data to generate images, whereas Computer Graphics methods follow a procedure-based technique to synthesise several images. For the purpose of the thesis we also narrow down our scope to image generation of indoor scenes using computer graphics as they allow more controllability over object features and spatial relations. The various methods used within indoor scene synthesis are also presented in section 2.3. These methods are different from traditional computer graphics techniques to generate images as they require modelling spatial relations or using scene graphs to generate images. Furthermore, background information is also provided in section 2.4 on the use cases to provide a better understanding of the existing methods in both fields and how this thesis benefits the fields. Finally, in section 2.5 an analysis of literature on indoor scene generating methods is discussed, and the research gaps are presented with a comparison to the method developed in the thesis.

2.1. Machine Learning For Image Generation

Machine Learning is the field of learning from examples or data by understanding and modelling the features of the available data. By learning the distribution of the data and identifying patterns, statistical algorithms can be applied to make predictions or identify structures in data. Deep learning is a sub-field of machine learning where an artificial neural network is created, similar to the structure of the human nervous system to learn the representation of the data [19]. Deep learning methods are powerful and can be used to synthesise images as the output of the neural network [53]. For deep learning models to generate images similar to the training data, they first need to learn its distribution. Generative Adversarial Networks (GANs) are extensively used in deep learning to generate photorealistic images and are the current state of the art in image generation [43] when compared with other generative models like Variational Autoencoders. A GAN consists of two models - a discriminator network and a generator network. The generator model synthesises an image by sampling from a random noisy distribution while the discriminator classifies the generated image if it is part of the training distribution or not [20]. The resultant loss and feedback are provided to the generator model and it learns to generate more accurate images by finding a mapping from the noise distribution to the training data distribution.

Initial works with GANs did not allow much controllability in the generated images but certain text-to-image generation methods have led to more progress in this area as the text input can specify features of the image that are expected or required [31]. Text-to-image generation uses an attention-based approach which encodes the input text and identifies important keywords that correspond to regions in the generated image that should have certain features. This allows a correlation between semantically meaningful parts with corresponding words and colour descriptions. Using the encoded text description, images are generated by considering a loss function that signifies a relation between the input text and generated image [31, 57]. Imagen, the current state of the art generates realistic and accurate images given a very complicated or implausible textual description [42].

Though there exist several works in deep learning that generate realistic and accurate images given text, they come with their limitations. A major limitation is that deep learning models are very complex and consist of several layers and parameters. They are also black-box in nature thus making them difficult to interpret and debug. Another drawback with these methods is that the neural networks are also very difficult to design due to the high level of complexity, and can take very long to train, thereby requiring high-end computation power. With a procedural algorithm based on computer graphics, it would also be easier to enforce controllability in the synthesised scene by having a rule-based approach. In section 2.2 we discuss ways to scale a particular scene by having several variations of the image. This allows several images to be created whereas most of the works for image generation with machine learning synthesise a single image for an input. Due to the above reasons, we proceed with a computer graphics approach for the thesis.

2.2. Computer Graphics For Image Generation

Computer Graphics is a discipline within Computer Science focusing on creating virtual images, scenes and other content. To synthesise visual content, computer graphics methods study the 3D representation of objects, render 3D models of objects, and create animations with these objects if needed. Computer graphics has several applications in the gaming industry and 3D modelling. In subsection 2.2.1 computer graphics methods are described where 3D objects are added to backgrounds to generate images, subsection 2.2.2 details software and engines that are used to create image datasets, and subsection 2.2.3 introduces the topic of domain randomisation to create multiple variations of the image by varying parameters.

2.2.1. Generating Synthetic Datasets

Computer graphics is extensively used to create synthetic datasets for machine learning purposes. These methods often use a pre-existing background and impose 3D objects in the background in a realistic way. One such method to create synthetic datasets for object detection uses a dataset of scene images such as a kitchen or dining room background [18]. Once the background is selected, potential surfaces in the background are identified using a semantic segmentation algorithm. This allows surfaces in the background similar to a kitchen counter or desk to be identified as regions on which 3D objects are rendered and placed. The scale of the 3D objects placed is also adjusted to make the image more realistic. This process is repeated for every background image where variations of the objects rendered are also made within the same image. Another approach uses a physics engine and computer graphics engine to create a dataset [37]. This method first simulates and renders a background, a surface along with a set of randomly selected 3D models. The objects are then allowed to fall by gravity using a physics simulation and are allowed to rest once they fall. Once the scene is stabilised, images are retrieved by varying the camera angles and lighting of the scene. The synthetic dataset is thereafter used for an object detection task. Synthetic datasets are also created for human pose detection by generating images with synthetic humans [55]. This technique retrieves a 3D model of a human body and allows the selection of a random shape. The pose of the body is also selected, and then a human texture with clothing is imposed on this model. Finally, a background of a scene is added and the lighting and camera angle are decided. The number of images created is scaled by randomising the selection of all possible selections made. However, this method results in unrealistic images often with the synthetic humans being of a very large size. The benefit of using these computer graphics methods is that a wide variety of images can be generated by retrieving realistic 3D models from several datasets that serve different purposes.

2.2.2. Dataset Generating Software

Software and packages that allow 3D modelling and rendering are also applied in this field to create image datasets. Blender is an open-source software based on OpenGL [56] that allows 3D modelling, simulation, rendering, animation and game creation [10]. It also allows users to customise applications and create tools using an API for its rendering engine. Unity Engine is another popular rendering engine widely used in the gaming industry but finds use cases in architecture and filmmaking [54].

BlenderProc
BlenderProc [13] is a procedural data generating pipeline based on Blender which uses objects from the SUNCG dataset [49] to generate images and create datasets for a variety of deep learning tasks. The modular pipeline recreates scenes from the SUNCG dataset and renders the scene as a realistic colour image, a depth image based on an internal depth estimator or an image with a semantic segmentation mask where similar objects are coloured. The BlenderProc sampler allows further variation in the scene by altering parameters for the camera, lights, and distance between objects.

Kubric
Similar to BlenderProc, Kubric is a dataset generating framework also based on Blender's rendering engine. It consists of a highly scalable pipeline which can be run on a local workflow to run large jobs on several machines in the cloud. The dataset generation process allows the selection of the number of objects in the scene from 4 datasets, with the objects, camera and lighting randomly initialised. Using the PyBullet physics engine to account for collisions and overlap between objects, and the Blender rendering engine, a random scene is created using the number of objects, and other initial parameters specified. This process is repeated several times while having random variations in the light, camera angle, and velocity of objects instantiated. In addition to the colour image, the scene's depth maps, instance segmentation, optical flow, surface normals, and object coordinates can also be extracted for use in computer vision tasks.

Unity Perception
Unity Perception is a synthetic data generating package that extends the Unity Editor and Unity Engine. The main contribution of this package compared to other works is that it provides an extensive framework to introduce randomisations in the images, apart from variations in camera angles and lighting [3]. The framework allows this by randomising the location of the objects to be detected with respect to the camera, such that the images contain objects of different sizes. In addition, a wall of random objects is placed behind the objects to be detected to which the object texture is also randomised.

2.2.3. Domain Randomisation
Domain Randomisation is a technique used in creating synthetic image datasets for object detection. The goal of domain randomisation is to help the model generalise well by having a lot of background variations in the images, enabling it to recognise true objects [51]. This also helps to bridge the gap between reality and the synthetic dataset by having diversity in the datasets. This is done by varying the quantity and shape of distracting (background) objects, the texture and position of objects, lighting in the scene, camera angle and random noise. One work that utilises domain randomisation for creating synthetic datasets, performs this randomisation by creating a background layer of 3D objects, and imposes the main objects to be detected in the foreground [23]. The objects in the background layer are randomised along with the light position and light colour. Furthermore, random noise is also added with some objects being blurred. Another approach adds different textures to the objects such as a flat colour, gradient of colours, chess pattern in the colour, and colour with Perlin noise [4]. One work that produced a dataset for car detection uses domain randomisation by placing the 3D object of the car in random scenes as background with an assortment of geometrical objects [52]. Structured domain randomisation is a unique approach that generates distractor objects in the scene based on probabilities by considering the context. This is in contrast to domain randomisation where objects are added to the image based on a uniform probability distribution [39]. For example, in an outdoor image generation setting, based on a parameter for road curvature, road lanes and sidewalks are generated, and vehicles, cyclists, houses and buildings are placed accordingly.

2.3. Indoor Scene Synthesis
For this thesis, the scope is constrained to generating images of indoor rooms with computer graphics methods as working with indoor scenes includes a limited number of realistic objects. This further allows more controllability over aspects such as the object's property or the spatial relations between objects. A computer graphics based approach was also selected for the thesis as it entails a transparent process for image generation. Additionally, the inner details of the method would also be easier to tune and configure, especially when compared with deep learning techniques such as GANs which are black-box and difficult to interpret. The main difference between this section and the methods

described in section 2.2 is the focused domain for image generation and the considerations made when generating images such as modelling spatial relations and considering object attributes. This section explores the field of generating indoor scenes and the methods involved in synthesising realistic scenes. In subsection 2.3.1 methods that convert text to 3D scenes are described, subsection 2.3.2 details methods where semantic scene graphs are used for creating indoor scenes, and subsection 2.3.3 describes a technique to include additional objects in a scene to make the scene more realistic.

2.3.1. Text to 3D Scene

Text to 3D scene methods consist of two main parts in the entire scene synthesis pipeline. The first step is to process the text as natural language and most methods addressing Text to 3D Scene creation follow the same process. The input text is processed either using the Stanford CoreNLP Pipeline [35] or other Natural Language Processing methods, where Part-Of-Speech (POS) tags are assigned to the raw text. These tags label a word as a noun, adjective, pronoun, verb, etc. The POS tags are then provided as input through a universal dependency parser which identifies spatial relations between objects and the object's properties. Using this dependency representation, there are several existing approaches used to create an indoor scene.

Early Works in Text to 3D Scene Synthesis

WordsEye is one of the first methods to create a 3D scene given text that can be applied to indoor and outdoor settings [11]. It considers various spatial relations, between objects, and object properties such as colour, shape, and transparency while also introducing poses and grips for a character interacting with an object. The objects, properties and relations are named as depictors and rules are created for depictors in a scene, with each rule having multiple scenarios based on the entities and relations involved. The scene is finally created by retrieving 3D objects that fit into the rules and thereafter modified. This approach is heavily rule-based and also requires the distance between two objects to be explicitly specified, for example, a table is five feet away from a chair, which makes this method inflexible. Another early work in this field, makes a less rigid computer graphics approach by analysing the voxels (3D pixels) of objects and identifying potential surfaces [44]. From the dependencies identified in the text and the spatial relations specified, the objects are placed accordingly with respect to each other based on the surfaces identified and extracted earlier. This work also allows controllability over selecting the object's colour but the spatial relations are limited and the resultant images are unrealistic.

Modelling priors and Spatial Relations

The limitation of these approaches led to advancements in research in Text to 3D scenes for indoor rooms. A key contribution is the learning of spatial representations from data and relating this to keywords in language that denote the representation [5]. The spatial representations are learned by studying several indoor room scenes and modelling prior probabilities by observing objects, their positions, and their co-occurrence with other objects [16]. The following priors are learned: the probability of object occurrence in different scenes (kitchen, bedroom, dining room); support hierarchy priors - the probability of objects supporting other objects (plate on a dining table); support surface prior - the probability of one of the parent object's surfaces supporting a child object; relative position prior - the probability of an object being placed at a particular region or position with respect to another object given a scene (placing a keyboard on a study desk with a monitor). To relate pre-defined spatial relations with text, the authors crowdsource text descriptions of two objects in a scene. The description provided of the spatial relations between the objects are then mapped with the following features for those objects - the distance between objects, overlap of the objects' bounding boxes and if the objects support each other. Finally, given the objects and spatial relation, the likelihood of an object's position with respect to another is predicted with a machine learning classifier. During the scene creation process, a scene template of the objects is designed from the dependency parser with all the objects, relations and properties. Relevant objects are retrieved from a 3D model database and using the priors and spatial relations learned, objects are instantiated at high likelihood locations, and the scene layout is optimised to minimise collisions.

Follow-up Work to Modelling Priors

Follow-up work that builds on this grounds textual descriptions to objects. This is known as lexical grounding where lexical terms containing the description of an object are mapped to 3D objects [7].

The mapping is created by training a machine learning classifier on a large dataset containing pairs of scenes and textual descriptions. The classifier learns features from the pairs and finds the lexical groundings. For example, it learns the mapping of a red cup, green room, black bed, and yellow round table to appropriate objects respectively. Additionally, this method also helps to learn lexical variants of an object (sofa and chair), making it more flexible and giving it the ability to generalise well during scene creation. This method is finally combined with the spatial relations and priors learned in [5] for indoor scene creation. SceneSeer is another follow-up work that follows the same approach as [5] but also allows interactions with the scene to insert, remove, replace, move or scale objects [8].

2.3.2. Semantic Scene Graphs

A semantic scene graph is a graph used to represent objects in the scene as well as relations between them. The nodes in the graph can represent either the object, the object's attribute or the spatial relation between objects, while the edges connect related nodes. Scene graphs are used for various applications such as visual question answering, image understanding or 3D scene understanding, but here we study its application in indoor scene synthesis [9]. A benefit of using semantic scene graphs is that they encode all the relationships in the scene and also denote semantic information of the scene by explicitly stating the spatial relations and the object's attributes. Using semantic graphs further aids our goal of controllability in scene synthesis as specific nodes in the graph can be altered or selected that would result in the creation of the scene specific to our needs. Figure 2.1 and Figure 2.2 illustrate examples of scene graphs given a text description of a scene.

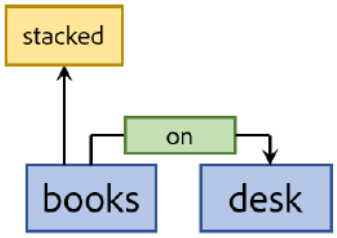


Figure 2.1: Scene Graph example for the description - There are books stacked on the desk. The blue nodes denote objects, green node the spatial relation and yellow node the property [34].

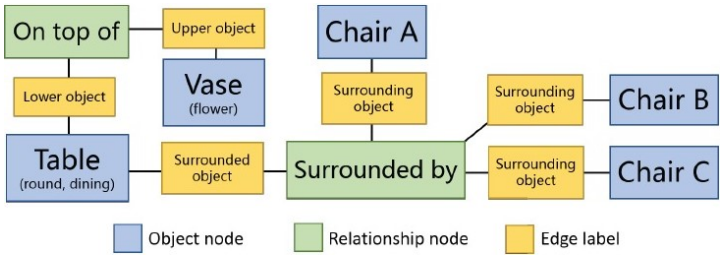


Figure 2.2: Scene Graph example for an elaborate description - The round dining table is surrounded by 3 chairs and there is a flower vase on top of the table. A more complicated description results in an intricate scene graph [34].

Scene Graphs in Text to Scene

Some methods convert raw text to a semantic scene graph via the intermediary step of generating the universal dependency parser from the text. One approach creates a semantic scene graph where the nodes encode information on the object, count of objects, attributes and spatial relations [34]. The scene creation process involves matching the scene graph created from the input with a database of scene graphs and ultimately selecting a subgraph from the matched graph. The scene graphs used in this method also provide a grounding between text and 3D objects as well as spatial relations. This method is described in detail in chapter 3.

Scene Graphs Modelling Object Co-Occurrence and Functionalities

An important requirement when using semantic scene graphs to generate images is to have a database annotated of the objects in the scene and their relations. Some scene graphs do not include nodes containing spatial relations or object properties but just the objects themselves. In this case, the weighted edges connecting the objects encode information about the scene and the relation between objects. One such approach assigns the weight of the edges based on the frequency of objects co-occurring, and activity priors [17]. An activity prior is assigned a value of 1 if the objects are used together by a human (bed, night table and TV) or if both objects would have a functional purpose for a human, and 0 otherwise. During the scene creation process, the user provides the objects and layout of the room. Given this, the graph for the output is created which consists of the input objects and additional objects that have a strong link to the input objects given the room size. Using the objects selected from the graph, a final layout is created which is similar to the database of layouts.

Scene Graphs with Multiple Edges

SceneGen is another framework that uses scene graphs to augment scenes with an additional object by considering its placement and orientation [27]. The nodes in the scene graph represent objects, rooms or object groups (dining chairs and dining tables) while edges represent spatial relations. The graph contains multiple edges between a pair of nodes. The relationships between the objects as edges are represented as follows: Positional Relationships - position of an object in the room, distance between objects and object groups, objects surrounding given object, object support relationships; Orientation relationships - an object facing towards the centre of a room, away from a wall, facing a group of objects, next to an object and facing it. The extensive scene graph generated using these relations are subsequently used in a probabilistic model, to find the appropriate position and orientation to place an object in an existing scene.

2.3.3. Including Contextual Objects

Some scene generation methods also include additional objects in the scene not specified by the user. Often, the addition of these objects makes the scene more realistic and provides more context to the scene. These objects could either be ones that co-occur with the objects specified or supporting objects for the specified object. For example, the co-occurring object for a plate could be a knife and spoon and the supporting object could be a dining table. The number of contextual objects included can be controlled by either specifying the maximum number of objects [5, 8] or by introducing objects that have a probability of occurring above a threshold [34]. Figure 2.3 shows an example of changing the scene by setting a context parameter [34]. The first image of the scene has a low α value and constitutes a scene with only the specified objects. Once the context parameter is increased to 0.5, the scene is augmented by including an office chair, two speakers and a notebook. These additional objects have a high co-occurrence probability with the existing objects of the scene and are thereby selected.

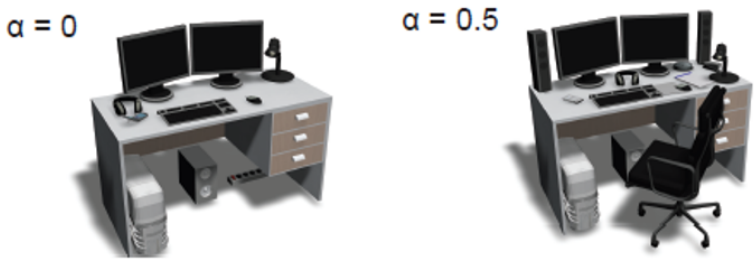


Figure 2.3: Change in scene with more objects such as speakers and chair on increasing the context parameter [34].

2.4. Use Cases

In this section background information and an overview of relevant methods for both use cases is provided in subsection 2.4.1 and subsection 2.4.2.

2.4.1. Generating Images for Interpretable Machine Learning

The black-box nature of complex deep learning models makes them difficult to debug and understand the reasons for failure or accurate results. This makes it a complicated and challenging task to deploy machine learning systems [14]. Additionally, as per European Union regulations and rules, it is also expected of systems to provide an explanation if requested, when a decision is made [22]. Interpretability in machine learning refers to the machine learning model providing an explanation of the functioning of a model in terms easy to understand by humans [2].

Methods to provide explanations in machine learning can be categorised on a local or global level based on the data instances being considered for interpretation [1]. Local explanation methods provide explanations over an instance of an image through saliency maps where the pixels that the model finds important for classification are highlighted [45] or via counterfactual explanations [21], where given a classified image an explanation is provided as to what should change in the image that results in the model giving a different classification. Global methods provide high-level explanations about human-understandable concepts that are important to the model for classification [28]. Interpretability methods can also be categorised as inherent or post-hoc. In inherent explainability methods, the model architecture is constrained and forced to ensure interpretability which could result in changes in model performance [2]. Examples of post-hoc methods are the local methods described above and are easier to implement as they do not require tuning of the models. The purpose of post-hoc methods is to use a trained model and then investigate model explanations.

Existing techniques that evaluate a model's explainability transform the dataset to understand its behaviour based on features of the dataset changed [59]. The BAM framework [59] does this by augmenting an image with an object from another dataset, and training two models trained on the object label, or scene label the object was placed in. The creation of biased datasets helps in evaluating the importance of features. Other works investigate the influence of features by removing important features from the image or by blurring highly attributed features of the image [24]. To further help in understanding the behaviour of the models, SceneUI is used to create biased and unbiased datasets, thereby benchmarking explainability methods. By generating biased datasets and training a biased model, the biases created should reflect in existing explainability methods used to understand the model behaviour.

2.4.2. Generating Images from Queries as Ground Truth

Specialised or proxy neural networks are deep learning models which are based on a reference neural network and mimic their behaviour to an extent [26]. The reference model is trained on a dataset and its outputs are used to train the specialised model. The specialised model consists of fewer layers and thus requires lesser computational resources. It is also unable to generalise and works well only for specific data. Specialised networks are often used in video analysis, where given a query the network identifies frames of the video that contain the query's predicates. The network achieves this by drawing bounding boxes in the frame to identify objects that are specified in the query. For a query - *Bus or Car*, the network should retrieve frames that contain either a bus, a car, or a bus and a car.

BlazeIt is a follow-up work where specialised networks are also able to consider more complicated queries that require the aggregation of objects in the queries such as the sum of objects occurring in the frames [25]. The framework is also capable of limiting the number of frames retrieved that hold the aggregated values. The process of identifying columns and relations from unstructured input like images or videos based on queries is computationally costly. To make this process of inferring machine learning from queries more efficient, Probabilistic Predicates which are binary classifiers are used [33]. The probabilistic predicates use binary classifiers such as Support Vector Machines, Kernel Density Estimators, or a Neural Network to determine whether the input (video or image) contains features that meet the requirements of the query predicate. Essentially, it does not allow the input to reach the inference stage and reduces the load on the query inference by machine learning, thereby making it more efficient.

SceneUI includes a feature to generate images based on a query. In addition to a basic *AND* or *OR* query with two predicates, the queries could also be more complex by selecting multiple (more than two objects as predicates). Furthermore, the queries could also be based on a Conjunctive Normal Form which is a combination of *AND*, *OR* and *NOT* queries. Given the objects in the query, the image generated should accordingly consist of the objects. As the images synthesised correspond to a query and predicates, these images can be used as ground truths for training the specialised models. There-

fore, when the model is required to process a query, it uses the ground truth images to learn which image to retrieve and which objects should be present in the image. In this thesis, we focus only on generating images using the *OR* query with two predicates to show the possibilities of using queries to generate images.

2.5. Literature Analysis & Research Gaps

The previous sections describe various methods for image generation using machine learning or computer graphics. For this thesis, we decide to proceed with indoor room scene synthesis using computer graphics as it allows controllability of objects and their properties to a wide extent, and is also explainable. An analysis of the existing works in indoor scene generation is also discussed as well as the methods and techniques involved. Keeping in mind the use cases of *Interpretable Machine Learning* and *Generating Images from Queries as Ground Truth*, the expectation of SceneUI is that it should allow controllability in the image over a wide variety of factors such as the objects, object's attributes, spatial relations, and should also allow an image to be generated based on an *OR* query containing two objects as a predicate.

Table 2.1: Overview of literature pertaining to Indoor Scene Synthesis and its features.

Paper	Object Direction	Query	Context	Colour	Spatial Relations	Quantity	Background	Scene Graphs
Learning Spatial Knowledge for Text to 3D Scene Generation [5]	X		X	X	X			
Text to 3D scene Generation with Rich Lexical Grounding [7]	X			X	X	X	X	
SceneSeer - 3D Scene Design with Natural Language [8]	X		X	X	X			
Language-Driven Synthesis of 3D Scenes from Scene Databases [34]	X		X		X	X		X
Adaptive Synthesis of Indoor Scenes via Activity-Associated Object Relation Graphs [17]	X		X			X		X
SceneGen: Generative Contextual Scene Augmentation using Scene Graph Priors [27]	X				X			X
WordsEye: An Automatic Text-to-Scene Conversion System [11]				X	X	X	X	
Real-time Automatic 3D Scene Generation from Natural Language Voice and Text Descriptions [44]				X	X			

To get a better understanding of the literature on indoor scene synthesis, and the requirements for SceneUI, a literature analysis was conducted to get an overview of the method's features and controllability factors. This overview is shown in Table 2.1. The rows in the Table denote the papers discussed earlier related to indoor scene synthesis and the columns correspond to the properties that the framework or system allows, in context to the use cases. The columns were selected to compare the methods over features they allow controllability over, and features we expect to have in our method. These features either assist in having more controllability when generating images or make the scene more realistic. The *Object Direction* column denotes whether the objects instantiated in the scene are oriented in a direction by considering other objects around it, such as a chair facing a table, thereby making the scene realistic. The *Query* column denotes whether scenes can be generated by providing a query with predicate terms, and none of the methods satisfies this. This column is crucial as a core requirement of the method is to generate images based on a query. Works are also analysed to check if they allow controllability over scene *Context* to include additional objects that make the scene more plausible; object *Colour* to select objects with different colours; object *Quantity* to specify the number of objects required in the scene; *Spatial Relations* between objects to have a specific arrangement between objects, and selecting whether the scene has a *Background*. Additionally, the table also denotes whether the method uses *Scene Graphs* for scene synthesis as this helps in further controllability over features in the scene.

The method developed through this thesis - SceneUI should ideally have all the properties stated in Table 2.1 and we set these properties as its requirements. This method can be developed by selecting one of the existing works as a baseline and making improvements to it to also fill in the existing gaps for research in indoor scene synthesis. Another limitation of existing works is their method of evaluation. The scene generation methods are usually evaluated by conducting user studies to obtain insights into the plausibility and realistic nature of the scene, or by comparing very similar works. Though this is not a limitation as such, the works do not evaluate the method while considering the downstream tasks the frameworks are developed for, which helps get an idea of the actual usability and utility of the method. With regards to this limitation, through the thesis, we aim to evaluate SceneUI and its images in the context of downstream tasks - the two use cases.

3

SceneUI

In this chapter, the main method - SceneUI is presented which generates images by providing a set of specifications or constraints as input. The core image generation procedure is based on the work by Ma et al. [34]. This framework was selected as the base work, based on the research gaps and literature analysis discussed in section 2.5 and Table 2.1. It satisfies most of the requirements needed to develop a controllable image generation method. The base framework allows controllability over spatial relations, the extent of context in the scene, and the number of objects to be synthesised in the image, and the object's attributes. Furthermore, it uses semantic scene graphs in its image generation algorithm which further helps in enhancing controllability. Another reason for this choice as base line was due to an available and open-source code base[1] that made the modifications easier to apply. To answer the research questions, modifications were made to the original work's code by introducing a user interface which allows the selection of objects to be introduced in the scene, the number of instances for each object, the spatial relations between objects, and the objects' properties, presence of a background room, and context parameter. In addition to selecting the controllable features, the user interface also allows input for objects as query predicates. The existing work had limited attributes about the object's properties in the scene graphs. These attributes were also expanded by augmenting scene graphs in the database with additional functional properties of the object and its colour. An entire overview of the scene generation pipeline by the system is provided in section 3.1 which also illustrates the information flow via scene graphs to the final scene. The various stages of the pipeline of Graph Synthesis, Graph Matching and Scene Creation are described in section 3.4, section 3.5, and section 3.6 respectively. In section 3.2 details about the dataset are described which contains the 3D models used for scene synthesis details of the scene graphs. In section 3.3 the procedure to expand the attributes is described with an overview of the incremented attributes. The details of the user interface through which the user provides specifications of the scene are provided in section 3.7 and the method to generate a scene by providing a query with objects is detailed in section 3.8.

3.1. System Overview

The original scene generating framework consists of four main stages as shown in Figure 3.1. The first stage requires the input that contains the description of the scene. In the original work, the user provides the input as raw text via a *Textbox*. With the help of Natural Language Processing tools such as Part-of-Speech tagging and a universal dependency parser, the raw unstructured text is converted to a more structured form. This structure holds information about the objects, attributes the objects have, and their relations with other objects. This is followed by the second stage in the pipeline of *Graph Synthesis* where the semi-structured conversion of the raw text is translated to a scene graph. From the text, appropriate object nodes, corresponding attributes and relation nodes are identified with connecting edges, thereby creating a scene graph from the input. In Figure 3.1, *O* denotes an Object node, *R* denotes a Relation node and *A* denotes an Attribute node. The third stage of the process is to find the most similar graph in the scene graph dataset, to the input graph. This process of *Graph Matching* identifies graphs that are closest in terms of objects, objects with attributes and spatial

[1] https://manyili12345.github.io/Publication/2018/T2S/index.html

relations. The most similar graph is modified by altering the nodes to meet the input specifications and is finally selected for the final stage in the pipeline for scene creation. The *Scene Creation* step retrieves the objects that match the properties in the selected scene graph, and the layout is optimised to avoid collisions and overhanging between objects. An example of the input, resultant synthesised graph, graph with aligned nodes, and final scene generated is also provided below each corresponding stage to represent it.

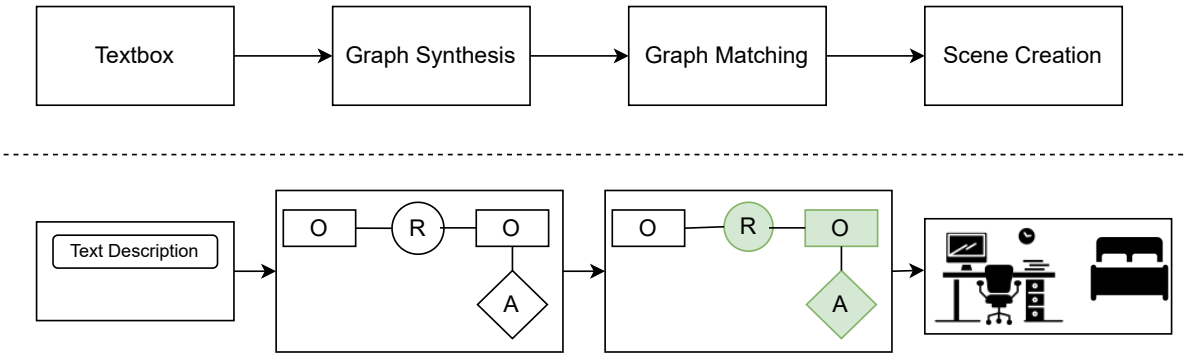


Figure 3.1: System Overview of the Original Work. The user provides a text description of the scene which is converted to a scene graph. Subsequently, similar graphs are identified and an image is generated.

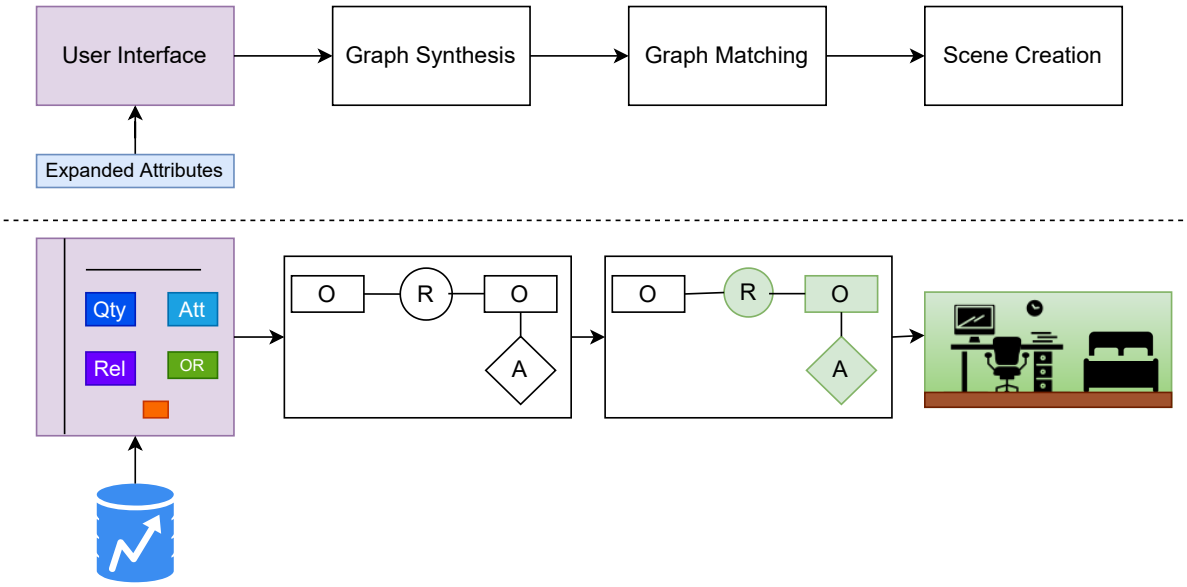


Figure 3.2: System Overview of SceneUI. The key difference is the User Interface which allows the user to select the quantity, attributes and relations of objects along with an *OR* query. The interface is supported by a dataset where objects are augmented to have more attributes. SceneUI also enables the image to have a background.

A key contribution of the thesis and SceneUI is the inclusion of the user interface that allows the user to provide an input with specifications that are required in the scene. This way of providing input not only allows the constraints and specifications of the objects to be directly provided but also removes the dependency of processing the input with NLP tools. The original work required the scene generating software to be connected to a server that processes the raw input to a structured form by a universal dependency parser. In this way, an unnecessary dependency is removed from the system. The user interface has additional benefits which are described in more detail in section 3.7. The updated pipeline with the user interface is shown in Figure 3.2. Similar to Figure 3.1, an example for each stage is provided below the pipeline to better visualise the stages of the pipeline. The main difference in the pipeline compared to Figure 3.1 is the user interface to describe the required scene and the augmented dataset of scene graphs. The rest of the steps in the pipeline are the same and follow the same process to synthesise a scene using semantic scene graphs. Additionally, the final image can be synthesised

with a background as a room. The differences are also highlighted in Figure 3.2. Similar to Figure 3.1, in Figure 3.2 *O* denotes an Object node, *R* denotes a Relation node and *A* denotes an Attribute node.

3.2. Dataset

Owing to the computer graphics approach, the dataset should comprise several 3D objects with the dimensions of their bounding boxes and orientations annotated. SceneUI uses the SceneSynth dataset which consists of 133 realistic scenes [16]. This dataset was used in the original baseline work and thus we select it for our implementation and experiments for the thesis. The scenes contain a variety of objects portraying different categories of rooms such as a dining room, living room, study room, bedroom, bathroom, etc. In total, there are 371 categories of objects with 1742 unique 3D objects in the dataset used in the scenes. This shows that there are several different types of objects based on colour, shape, or size for each category of object. The presence of different objects within a category aids our goal of controllability as different objects can be selected as per the method's needs.

The SceneSynth dataset originally consists of the 3D object file to be rendered along with a file describing the object's material. The authors of [34] augment the SceneSynth dataset by including additional information about the size, dimension and orientation of the objects. The size of the 3D object is denoted and constrained by a box known as a bounding box. The dimensions of the bounding box are used to model spatial relations between objects. Each 3D object is associated with the following files -

- Material Template Library (*.mtl*) - This describes the surface and appearance properties of a 3D object. It also provides information about the colour the object may radiate, the object's colour itself or the transparency and reflective nature of the object.
- Object Bounding Box (*.obb*) - This describes the centroid of the bounding box, the orientation of the three axes, and the size of the bounding box
- Support Plane (*.supp*) - The support plane is extracted for the model which allows other objects to rest on the plane
- Top Plane (*.bbtop*) - This details the dimensions of the top plane of the model's bounding box

The dimensions of the bounding boxes and supporting planes are used especially to model relations between objects by considering the relative positions and distances between objects via the bounding boxes. For a particular spatial relation, the orientation and distance between the bounding boxes are considered. For example, for a spatial relation that denotes whether an object is to be placed on top of another object (plate on table), the support plane of the supporting object is considered to appropriately place the object. Furthermore, the size of the bounding boxes is also considered while optimising the scene layout to avoid collisions between objects.

3.2.1. Semantic Scene Graphs

Semantic scene graphs, described in subsection 2.3.2, are a core component for image generation in this method. The graph consists of three types of nodes: an *object* node for each object in the scene, an *attribute* node describing the object's properties, and a *relation* node that signifies a spatial relation between two objects. The spatial relations between objects in the scene graph define the relative position of one object with respect to another object's frame of reference. The relations defined are *left*, *right*, *on*, *front*, *back*, *near*, and *under*. An overview of the attributes defined for every object in the scene graph is provided in Table 3.1. The attributes are limited to the functional use of the object, shape or size, and the attributes are provided only for chairs, tables or a bed. In the scene graph, the *object* node is assigned a label of the object, the *attribute* node has a label of the property the object holds, and the *relation* node is assigned a label denoting the spatial relationship between objects. Finally, edges connect the attribute node to its corresponding object or connect two object nodes via a relation node. A visual of a semantic scene graph for a scene containing a chair, desk and bed is illustrated in Figure 3.3. The chair has the attribute of an office chair, and the bed is queen-sized. Additionally, the chair is in front of a desk and the desk is to the right of the bed. These spatial relations are denoted in the graph accordingly.

In this way, a scene graph is created. Using this process to generate a scene graph, 133 scene graphs are synthesised by the authors of the original work by considering all the scenes in the SceneSynth dataset. In every scene, the object nodes and labels are identified, followed by the object's attribute nodes based on visual observations of the shape, or appearance. The scene is augmented with spatial relation nodes, whose labels are determined based on the position of objects around a reference object. Edges are then assigned from one node to other relevant nodes. The graph of each scene is stored as a file along with the IDs of the 3D models to be retrieved that correspond to the objects in the scene, and information about the object's bounding boxes and position in the scene.

Table 3.1: Attributes of objects in the scene graphs in the dataset.

Object	Attribute
Chair	Dining
	Office
	Sofa
Table	Dining
	Coffee
	Round
	Rectangular
Bed	Queen

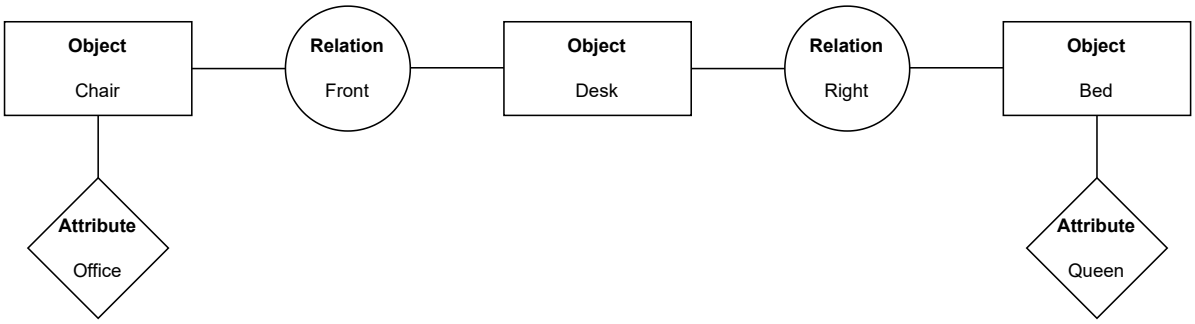


Figure 3.3: Scene graph for a scene with an office chair in front of desk, with the desk on the right of the queen sized bed.

3.3. Expanding Attributes

The scene graph dataset annotated and created by the authors of the baseline work covers only 3 objects with a total of 7 attributes as shown in Table 3.1. These properties cover mostly the functional use of the object such as whether it is a dining chair, office chair, or coffee table. Other attributes refer to the shape or size of the object such as a round or rectangular table, or a queen-sized bed. Thus, the existing attributes in the dataset are quite limited and none of the properties also describe the colour of the object. To further aid our goal of having controllability in image generation, we decided to expand the attributes in the scene graphs of the dataset by annotating the graphs in the dataset. Having more attributes to choose from gives us more flexibility in selecting the controllable aspects of the objects in the image. This also makes using SceneUI to generate images more relevant for the use cases. Having more attributes for an object helps the user to create multiple images of the same objects with different colours, materials and designs apart from the existing attributes. In addition to augmenting an object's attributes, attributes are also created for objects not present in Table 3.1. The objects introduced with attributes are: Desk, Bookcase, Couch and Stool. An overview of the set of objects and their attributes is shown in Table 3.2. The augmentation of objects and attributes was done by manually making changes to the files in the dataset by creating new nodes and connecting relevant edges. This is described in more detail in subsection 3.3.1. In subsection 3.3.2 a description is given of the attributes for corresponding objects.

Table 3.2: Expanded attributes of existing and additional objects in the scene graph dataset.

Object	Colour	Functionality	Material	Shape
Desk	Light Brown, Dark Brown, Brown, White, Black-Brown, Light Brown-White, Brown-Grey, Yellow, Grey	Study		
	Black, Dark Brown, Brown, Light Brown, Blue	Lab		
Chair	Light Brown, Dark Brown, Grey, White Brown, Light Grey, Dark Grey, Brown-White	Dining		
	Brown, White	Dining, Armrest		
	Black, Reddish Brown, Yellow-Green, Grey, Red, Blue, Light Brown, Brown, Grey-Black, Black-Brown	Office		
	Yellow, White	Sofa		
Bed	Grey, Purple, Black-White	Queen		
	Grey, White	Single		
Table	Dark Brown, Brown, White, Design, Yellow	Dining		Round
	Green		Glass	
	Grey, Brown, Red, Dark Brown, Light Brown			Rectangular
	Brown		Glass	
	White	Coffee		Round
	Brown		Glass	
	Black, White			Rectangular
	Transparent		Glass	
Bookcase	Brown	2 shelves		
	Brown, Grey, Black	3 shelves		
	Dark Brown	4 shelves		
	Black	5 shelves		
	Brown	6 shelves		
Couch	Brown, Grey, Red, White, Light Grey, Light Brown			
Stool	Blue, Brown, Black, White	Backrest		
	Grey, Blue, Brown, Black	No Backrest		

3.3.1. Method to Expand Attributes

The selection of new objects in the dataset which have attributes was determined by examining all the scenes in the dataset. During this examination, observations were made on recurring categories of objects and the variations in the object's colour, material or functional type. These common objects were identified which had several different properties and were also typically found in certain types of rooms. Once the objects to be introduced were identified, the colours exhibited in every instance by the object were also noted as properties that category may have. The remaining attributes of the object were identified by examining occurrences of the object and categorising them into their different properties. To have an additional reference of properties the object may have, the Visual Genome project [29] was referred. The Visual Genome project is a dataset widely used for visual question answering

which has thousands of images with descriptions of several regions in the image, often representing various parts of an object. Using a combination of visual observations from the dataset and examining the objects in the Visual Genome project, the attributes were determined as shown in Table 3.2.

To introduce these additions, 102 of the 133 existing scene graphs in the dataset were modified by making changes manually. These 102 scene graphs were the ones that contained the objects in the scene to which new attributes were to be added. The attributes were expanded by identifying the files that required changes, as well as the attribute nodes to be added for the object in the scene. There was a possibility for multiple attribute nodes for a single object or attributes for different objects. In every file, the new attribute nodes were manually created based on the observations made earlier. Once the attribute nodes were created, edges were synthesised in the file to connect the attribute nodes to their relevant object nodes. Finally, the count of the number of nodes and edges was updated.

3.3.2. Expanded Attributes and Objects

As the existing attributes for objects do not include the object's colour, a common theme was to include the colour for all the objects shown in Table 3.2. Some colours for objects like the chair or desk consist of a combination of colours: Black-Brown, Grey-Black, etc. This combination of colours was selected as the object's colour exhibited a blend of the two colours. In addition to the object's colour, the augmented attributes are based on the object's functionality, material, or structural design of the object. For example, the desk is classified as a typical study desk or a lab desk. A dining chair is further divided into one that either has or does not have an armrest, depicting controllability over the item's structure. The functional attributes of the stool also describe the structural design of the object, as the user would be allowed to select a stool with or without a backrest. Furthermore, the attributes of a bed consist of a smaller single-person sized bed in addition to a larger queen sized bed. The dining and coffee tables also allow a further categorisation based on the material, by allowing the table to be selected as one that's made of glass. Finally, with the additional attributes, properties to the bookcase are included so that various bookcases can be selected by determining the number of shelves from a range of two to six. The couch is a frequently occurring object but does not have much variability in its design due to all of them having a similar shape and size. Thus the attributes of a couch are only limited to its colours. In conclusion, the prior attributes of the objects are incremented manifold, and various attributes were added ranging from colour, shape, size, purpose and design. Having these graphs with vast attributes aids our goal of having controllability over a wide range of object properties. Figure 3.4 illustrates two dining tables with different colours. These images were generated using the method by selecting the dining table with specific colours, included as a result of attribute expansion.

(a) Black Dining Table.

(b) White Dining Table.

Figure 3.4: Generating Tables with different colours selected from the expanded attributes.

3.4. Graph Synthesis

The description of the scene provided by the user is stored in a C++ data struct. A separate structure is created for a scene description which is composed of a structure of entities where an entity is an object that is required to be present in the scene. Furthermore, each entity contains a structure describing

the type of spatial relations as well as the other entity it has a relation with, and another structure containing the type of attribute the entity should possess. Additional information is encoded in the entity about the number of attributes, relations, and the count of instances of the entity required in the scene. The original work initialises the structures with the scene description by referring to the NLP processed form of the input. In SceneUI, we read the input provided by the user interface and translate the input to the structures.

Once the structure containing the scene description is created with entities and their objects, and relations, a scene graph is synthesised from this input. Every entity in the scene is parsed and for each entity, the relevant relation and attribute nodes are generated. As every entity denotes an object, every entity has an *Object* node created representing it. If the entity has multiple instances, as many *Object* nodes are created. The relation information in the entity structure is considered a reference. Every relation that the entity holds is parsed and a new *Relation* node is created with a label denoting the spatial relation, and an edge is added from the current entity to the entity it should be connected to. This is further repeated for all the instances of the entity. A similar process is followed to create nodes for an entity's properties. For every attribute of the entity, a new *Attribute* node is created and assigned a label for the property the object should hold. A connecting edge is then created from this node to the entity's *Object* node. In this way, the entire scene graph is created by generating entities as *Object* nodes along with their respective *Relation* and *Attribute* nodes. Every *Object* node in the graph is instantiated in the scene. This input graph is then used for the subsequent step of *Graph Matching*. Object nodes along with their respective Relation and Attribute nodes. Every Object node in the graph is instantiated in the scene. This input graph is then used for the subsequent step of Graph Matching.

3.5. Graph Matching

With the input graph synthesised, the next step is to find a scene graph from the dataset that is most similar to the input graph. This process of matching the input graph with existing scene graphs is graph alignment. By identifying the similar graphs, the method knows the specific locations to render the matched objects. These locations are based on the SceneSynth dataset where objects in the scenes are placed in realistic positions. Once the most suitable scene graph from the dataset is selected, it is altered by removing or increasing nodes to meet the explicit requirements from the input graph. By removing nodes from the selected graph, we can prevent unspecified objects from occurring in the scene. The graph alignment process is described in subsection 3.5.1 along with a metric to score similar scene graphs. The method to modify the highest scoring scene graph is detailed in subsection 3.5.2.

3.5.1. Graph Alignment

The input graph obtained from the graph synthesis stage is denoted as UI_g. The object nodes in the graph are explicitly specified to be in the scene and are also required to have certain properties as denoted by their attribute nodes. The procedure for graph alignment requires the input graph to be compared with all the annotated scene graphs in the database (D_g). Each comparison is scored by a metric, and the highest scoring graph is ultimately selected in the scene.

In the graph alignment phase, the object node in UI_g is aligned when the object node with the same category name is aligned with an object node from D_g. Furthermore, if the object node from UI_g is associated with attribute nodes, it is aligned when the object node from D_g also possesses the same attributes. A pairwise relation node from UI_g is aligned when the type of relation matches with the relation node of D_g along with the object nodes the relation is connected to. This graph alignment procedure is repeated for all remaining 132 scene graphs in the dataset to obtain candidate graphs from the highest scoring scene. The metric to score the graph alignment is shown in Equation 3.1, where for every graph comparison, the number of nodes matched is counted and summed. N_i denotes the node from UI_g and N_j is a node that belongs to the D_g graph. For the pair of nodes being compared, $M(N_i, N_j)$ achieves the value of 1 if the nodes match and 0 otherwise. The match scores from all the node matches are summed for every scene graph and assigned to the scene graph to obtain the highest scoring scene graphs.

$$M(UI_g, D_g) = \sum_{N_i \in UI_g, N_j \in D_g} M(N_i, N_j) \tag{3.1}$$

Figure 3.5 depicts the scene graph UI_g synthesised from the user input and Figure 3.6 illustrates a scene graph D_g from the dataset. The two figures are compared for the graph matching and alignment phase. The nodes highlighted in Figure 3.6 are the aligned nodes, identified after comparing the two graphs. The *Object* node with a dining chair is not aligned as the input graph explicitly specifies an office chair. This underlines the requirement for aligning an object node where both the object category and its associated attribute should be matched. Additionally, the Front *Relation* node is not matched due to the different objects it is associated with.

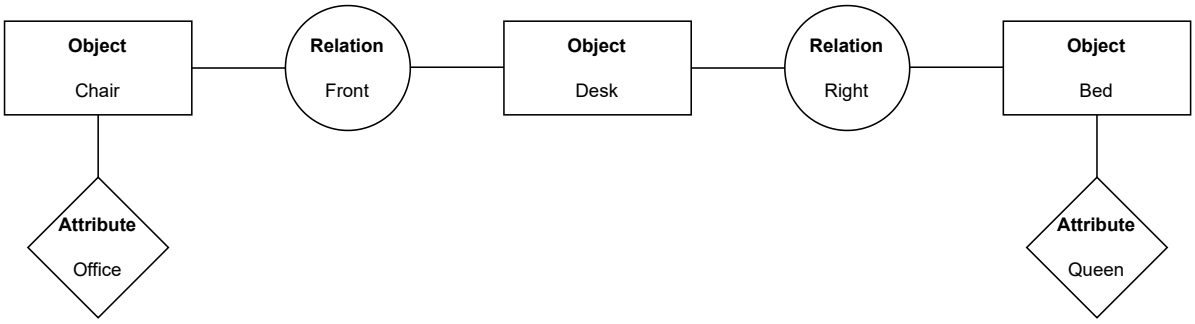


Figure 3.5: Scene graph UI_g synthesised from the input.

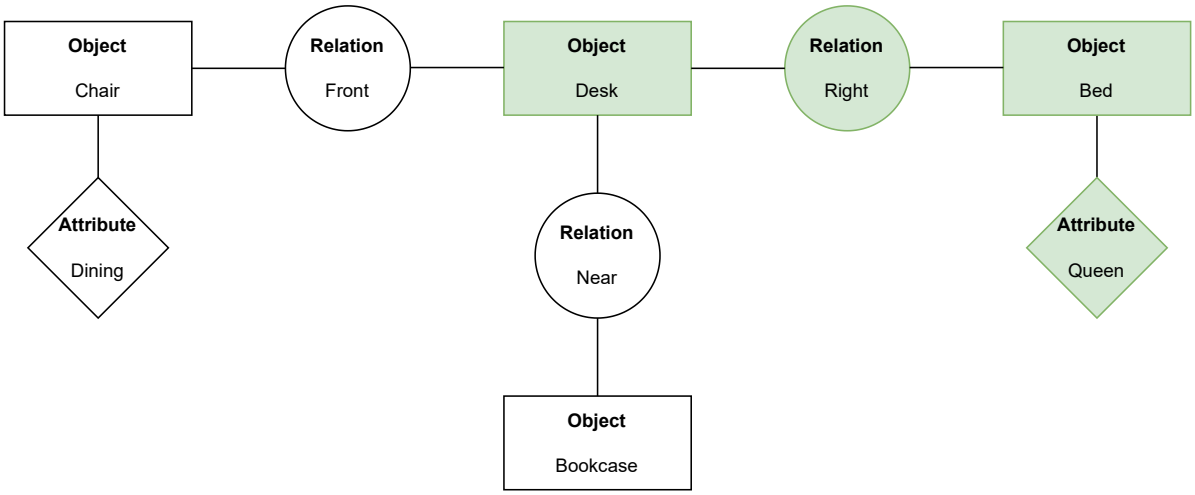


Figure 3.6: Scene Graph D_g from the dataset with aligned nodes that match with UI_g highlighted.

3.5.2. Subgraph Augmentation

The highest scoring scene graphs are stored in a list where the graphs are denoted as S_g. It is quite likely that the high scoring candidate graph does not have all the object nodes explicitly specified in UI_g. This requires augmenting the selected graph S_g to create the final scene. S_g is also named a subgraph as it is only a part of the final scene graph synthesised. To augment this graph, the absent nodes specified in UI_g are generated and included in S_g. The new nodes created are added to the scene such that they match UI_g. *Object* nodes are created and added with their attributes if any, and a *Relation* node is also synthesised and linked to the object nodes as indicated in the input graph UI_g. In this way, the candidate scene graph S_g is augmented with additional nodes to ensure that the final representation contains all the requirements specified in UI_g. There are also cases where the subgraph S_g has many more nodes than UI_g and has all the nodes from UI_g aligned. In this case, S_g is pruned by removing the extra nodes not specified in UI_g. For example, if UI_g specifies just a chair and desk, and

S_g contains a desk and chair with a monitor, computer, mouse and keyboard on the desk, the nodes from S_g are removed such that only the nodes corresponding to the chair and desk mentioned in UI_g remain.

Once S_g is altered by removing or adding nodes, there is a possibility to further enhance the scene by including contextual objects to it. Using an approach defined in subsection 2.3.3, these objects are added. The contextual objects included here are those that co-occur frequently with a particular entity. The system provides a controllable parameter α where objects that have a co-occurrence probability larger than α are added to the scene. Thus, a low context parameter will result in several often, irrelevant objects added to the scene. An example of introducing contextual objects in the scene graph based on object co-occurrence is illustrated in Figure 3.7. The monitor and speaker objects frequently occur with a desk and are typically placed on the desk. The input graph from Figure 3.5 is augmented with relevant objects by adding additional nodes and connecting them to existing nodes, based on co-occurrence probabilities learnt. Additionally, the support hierarchy probabilities are also learned to see which type of objects support the other. Therefore, if an object is specified in UI_g without its supporting object, the final S_g graph is expanded by adding relevant nodes to the graph such that the unsupported object is now supported, according to the support hierarchy learned. For example, if UI_g describes a scene with a plate and knife without a table, with the help of context enhancement, the scene graph is updated to include a table on which the plate and knife will be placed.

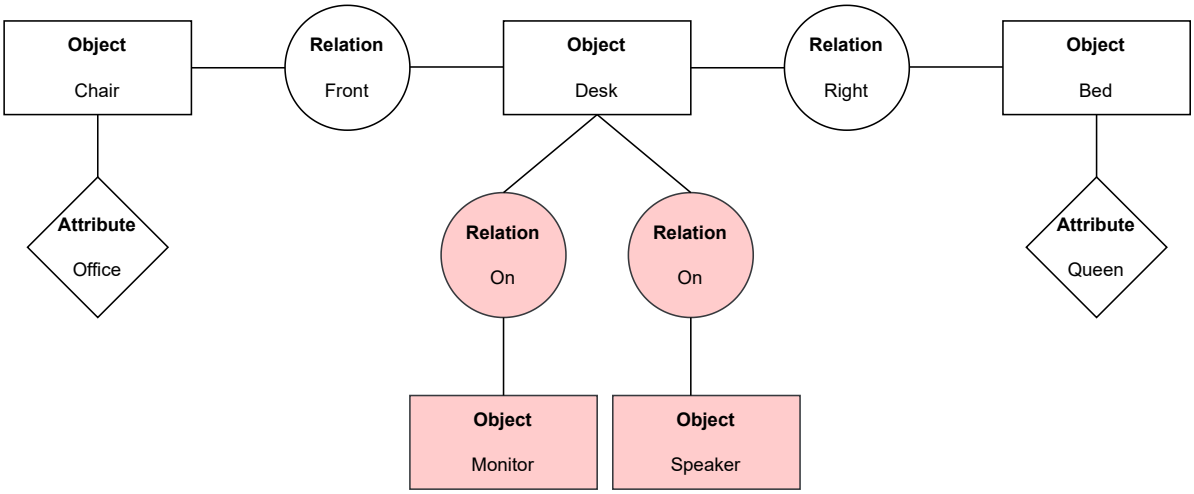


Figure 3.7: Augmented Graph S_g with contextual nodes highlighted.

3.6. Scene Creation

The graph S_g is retrieved from the *Graph Matching* phase after subgraph augmentation, and context enrichment is selected for the final step of scene creation. The process of scene creation requires rendering relevant objects in appropriate positions in the scene environment that conform to the requirements specified in UI_g. The original aligned nodes in S_g before the node augmentation phase originate directly from the database graph D_g. The scene graph file in the dataset contains information on the positions of these objects in the scene. Thus, the positions in the scene of the originally aligned object nodes from S_g are known. In addition to the positions of the objects, the 3D models that correspond to the object nodes and their attributes are also referenced, to render them accordingly.

3.6.1. Generating Subgraph Scene Nodes

The aligned objects from S_g that are based on the dataset scene graph D_g are first introduced in the scene. The positions of these objects in the scene are already known as they are aligned objects. Thus, the objects are placed based on the size of their bounding boxes or their level in the support hierarchy. The bounding box dimensions are obtained by reading the dimensions of the boxes from the dataset file as well as selecting the 3D models that match the object's properties. The larger objects are placed

first followed by the supporting objects such as tables or desks. This is proceeded by the remaining objects that satisfy the constraints or requirements. If the placement of an object does not meet the layout score, the layout generating algorithm is reverted to a previous state. The object's position is then modified by placing it at an alternate location. If after several rollbacks, an appropriate location for the object cannot be found, a failure message is returned and the object is skipped. In this way, a scene is generated and rendered for objects in the subgraph S_g that completely match and align with the input graph, UI_g.

3.6.2. Rendering Augmented Nodes

For scenarios where the subgraph needs augmenting with additional nodes, the placement of the new objects needs to be considered based on their spatial relationship. This is primarily because the positions of the objects in the original S_g graph are known. The authors of [34] formulate a score for a new object o added to the scene with a spatial relation as shown in Equation 3.2, to ensure its placement is physically plausible.

$$Score(o) = L(o) \cdot H(o) \cdot R(o) \tag{3.2}$$

$L(o)$ is the term that indicates a collision penalty that holds a value of 1 if there is no collision between the objects and 0 otherwise. A collision is when the bounding box of the object o intersects with another object in the scene, which would result in an unrealistic scene of the objects appearing to be merged. The term $H(o)$ denotes an overhang penalty score which is defined in [16]. The overhang penalty is defined to avoid the object being placed on the edge of its supporting surface which would result in an implausible scene as the object would not realistically rest in such a position. The final relation score $R(o)$ is defined in Equation 3.3 [34].

$$R(o) = \omega \sum_{o_i \in O_E, r \in E} A(o, o_i, r) + (1 - \omega) \sum_{o_i \in O_U} A_I(o, o_i) \tag{3.3}$$

The term $R(o)$ considers the implicit and explicit relations for the object o when introducing it to the scene. E is the set of relations specified from the user interface input and O_E is the set of objects that are a part of the relations in E. The term $A(o, o_i, r)$ is a score based on the arrangement of two objects having a particular relation. The score maps the probability distribution of the relative position of the two objects by modelling their respective coordinates and the orientation between the objects, for all scenes when they occur together. O_U is the set of objects already present in the scene and $A_I(o, o_i)$ is the implicit relation priors between the object o and the other object between the scene o_i. ω denotes whether a higher priority should be given to the object's explicit relations or the implicit relations for the object placement.

A layout quality threshold is determined to allow objects to be placed at a location only if they satisfy a minimum layout score. If the object placed in the scene results in a collision, overhang, or a low layout score, the layout is then optimised by using a hill-climbing strategy. This strategy involves finding new possible locations to place the model based on the distribution of relations between objects previously modelled. If the layout score result is insufficient or 0 again, the location is not considered for future iterations. If after several iterations, the object cannot be placed at a location that achieves a sufficient layout score given the constraints, the object is placed at a random location.

By following the above process, the objects that belong to augmented nodes in the subgraph are placed at appropriate positions in the scene with the existing objects whose positions are pre-defined. The objects are also placed such that collisions and overhanging are avoided, thereby making the scene created plausible and realistic. The scene rendered in the 3D environment can then be used for various purposes which are presented in chapter 4. The system allows the user to rotate the camera angle and zoom of the 3D environment which helps to get a view of the scene from all possible viewpoints. A screenshot function allows the user to extract an image of the scene. An image from the scene generated is illustrated in Figure 3.8, which is based on the input scene graph depicted in Figure 3.5

Figure 3.8: Final image generated from the description and input scene graph.

3.7. User Interface

The most important contribution and addition to the existing work is the addition of the User Interface. The interactive User Interface was designed using QT: a C++ based platform to develop mobile and software applications [40]. QT Design and QT Creator were extensively used to create the layout of the interface, and various widgets, tables and lists to take input from the user or display information. These tools also allowed the implementation of back-end logic that enables the interaction between the layout widgets and adds functionalities for interacting with various elements in the layout. In this section, we describe the user interface designed for describing the scene for indoor scene generation. In subsection 3.7.1, we describe how table widgets are used in the interface to take input from the user, subsection 3.7.2 details the use of lists to display and select information and subsection 3.7.3 explains how additional parameters are considered for scene synthesis. The last subsection 3.7.4 highlights the importance of the interface and the benefits of using the interface when compared to the original work. The user provides their specification for the scene by following these steps by providing information in the user interface:

- Select the Object from the Object List and place it in the Quantity Table.
- Enter the number of instances for the object which adds the objects to the Object Scene list.
- To specify the attributes of an object, select the object in the Object Scene List.
- Select the checkbox attributes and colours from the list based on the object selected.
- Add Objects in the Relation Table and specify the spatial relations between the objects.
- Determine whether the scene should have a background, contextual objects and value of the context parameter.
- Click on the *Generate* button to synthesise the scene and extract the image.

The user interface designed is illustrated in Figure 3.9. The tables outlined in red (1) are a core component where the user is allowed to define relations between objects, attributes of an object, or the number of object instances by adding or removing rows from the table. The widgets on the left of the layout outlined in blue (2) are the lists used to display information such as the relations, objects in the dataset, objects in the scene, and the object's colours and attributes. Additionally, the interface also consists of parameters to control the level of contextual objects and whether the scene should be generated in a room as the background. These parameters are outlined in the figure in green (3). The figure is also annotated with a legend.

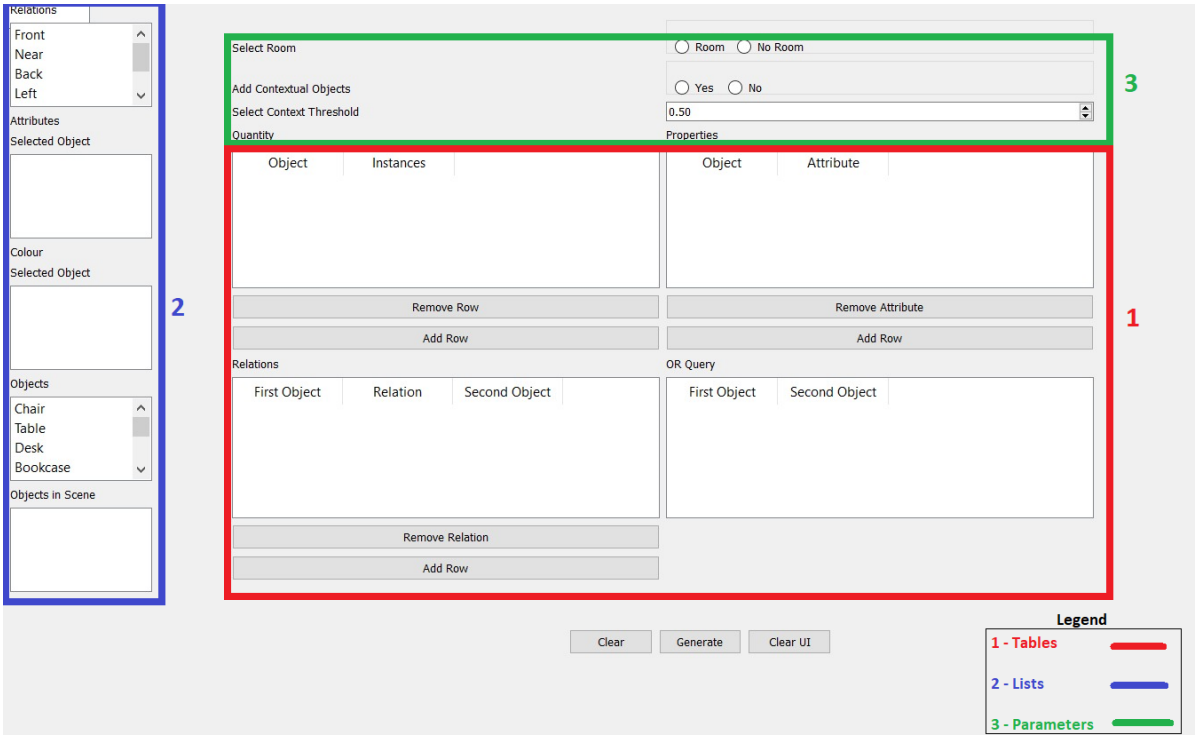


Figure 3.9: User Interface designed with tables (1) to display and input information, lists (2) to display objects and select attributes, and parameters (3) to further tune and control the setting.

3.7.1. Tables

Table elements in QT - QTableWidgets are a core component of the interface to provide specifications of the scene. The interface designed consists of four QTableWidgets - the *Quantity Table*, *Attribute Table*, the *Relations Table*, and the *Or Query Table*. These tables are enclosed within the red box in Figure 3.9. The Quantity table is the first table that the user interacts with when providing the scene's specifications. The user selects the object in the scene by dragging the item from the Objects list and placing it in the table, or by typing the object in the table cell. When the column denoting the number of instances of the object is determined, the object is directly added to the Scene Objects list. If multiple instances of an object that has attributes are specified in the table, the object category is indexed by adding the object to the list as separate entities with each having a unique suffix attached to the object name. By segregating and naming the objects uniquely, the user can specify different attributes for each object instance of the same category. In addition to specifying each object's attributes, their corresponding spatial relations can also be defined. For example, if the user specifies 4 chairs in the *Quantity Table*, the Scene Object list is updated by including the objects *Chair-1*, *Chair-2*, *Chair-3*, and *Chair-4*. The user can then choose any two of these chairs to be dining, and set the other chairs as an office chair or a sofa chair. The office chair could be placed near the desk while the sofa chair can be located behind the dining table.

The *Attribute Table* allows the user to describe the properties that an object is expected to have. The table provides a column for the object and a second column to determine the corresponding object's attribute. The attribute of the objects can be manually entered by the user or by clicking on the object in the scene list and then selecting the attribute or colour from their respective lists. When the checkbox from the attribute or colour list is selected, the property is added to the table along with the object.

The *Relation Table* allows the user to specify the spatial relations between the objects. The first column is for the object in whose reference the spatial relation is specified. The spatial relation is selected from the *Relation List* as well as the second object with which the relation is defined. The cells in the table are mutable and also allow the user to type in the object or spatial relation instead of dragging

and dropping elements from the object or relation lists.

The *Or Query Table* takes as input the two objects which are the predicates to the *OR* query. An explanation of scene generation based on the query is provided in section 3.8

3.7.2. Lists

Similar to QTableWidgets, the User Interface includes QListWidgets which are lists used to display information or provide functionality. The QListWidgets used are for displaying the possible *Spatial Relations* between objects, the allowed categories of *Objects* in scene generation, the *Scene Objects* which are the objects required in the scene as specified by the user, and *Attribute* and *Colour* lists to dynamically display the available properties or colours for a selected object in the scene. The lists displaying this information are highlighted within the blue box in Figure 3.9.

The *Spatial Relations* list as implied consists of the seven possible spatial relations between two objects. These relations are provided as a reference to the user and the relation in the *Relation Table* is selected from this list.

The *Objects* list provides the names of all the 371 object categories that can be used for selection in the scene. This gives the user an overview of the possible objects that can be instantiated and can be used as a reference to customise the scene by selecting relevant objects.

The *Scene Objects* list is a dynamic list which comprises the objects that are required in the scene as stipulated by the user. All the objects in the list are generated in the scene. The list gets incremented whenever the user adds an object to the *Quantity Table* and determines the number of instances. When an item in this list is selected, the *Attribute* and *Colour* lists get filled with items as checkboxes if the item consists of attributes. The *Colour* list is updated with values corresponding to the object based on the colour column in Table 3.2. Similarly, the other attributes of the selected object are displayed in the *Attribute* list. The *Colour* list is also dynamically updated based on the attribute selected. This is done to ensure that the user can only select the colours exhibited by an object with a given property. On checking the checkboxes of the items in the *Attribute* list or the *Colour* list, the property and the corresponding object are added to the *Attribute Table*.

3.7.3. Parameters

In addition to the lists and tables in the user interface, extra parameters are also included in the interface, to allow for more controllability in the scene. These are highlighted as elements in the green box in Figure 3.9. The first parameter is to determine whether the objects specified in the interface should be present with a background. The background is provided as a room in which the objects are located and the room is also a 3D object in which the objects are placed. The user selects whether the scene should contain a room by selecting the appropriate radio button. If a background room is required, a new *Object* node for the room is created, and an *On Relation* node is created, connecting the remaining scene object nodes to the room (object) node using this relation. Including this relation ensures that the objects are constrained to be positioned inside the room. Allowing the selection of the room along with the other objects and the relations they have is an important contribution when compared with the earlier work [34]. The original work which takes only text input does not allow the scene described to have a background. The backgrounds are only loaded as a stand-alone room or when an existing scene graph from the dataset is loaded.

The user interface also enables the user to select whether contextual objects are to be added in the scene. If the user wishes to include them, the appropriate option is selected. The user can also select the context parameter α, where a low value would result in many and often irrelevant objects included in the scene, while setting a high value for the parameter results in the inclusion of only very relevant objects.

3.7.4. Comparison with Original Work

The original work in [34] consists only of a textbox where the description of the scene is given as raw natural language. The natural language input requires additional processing to convert it to a scene

graph. With the user interface described in this section, the additional steps of converting text through a dependency parser are skipped, as the input provided via the interface is all gathered to create the input scene graph. Every object in the scene list is considered as a separate entity, and its relation nodes are created by parsing through the *Relation Table* to identify instances where the entity is the first object of reference. Similarly, the *Attribute* nodes are created from the *Attribute Table* where a property of the object is specified. By explicitly specifying the object's relations, any scope of ambiguity due to text is also eliminated. In the original work, the scene is generated by describing it one sentence at a time which could be time-consuming and cumbersome for the user. The interface also provides a solution to this issue by enabling the description of all components at the scene at once.

By indexing the object added to the scene list, properties for every instance can also be customised, which was more complicated to ensure in the earlier method. Furthermore, the interface is designed in a way to reduce the cognitive load on the user by providing the possible objects in the scene to select from, the object's possible properties, and the spatial relations. With this information provided in the interface and easily accessible, the user would not have to remember these details. An additional example of this is the updating of the *Colour* list based on the attribute selected of the object. The updated list shows the user the available colours for the object that can be chosen given an object's attribute selected. The tables in the interface also serve the additional purpose of providing an overview of all the attributes and relations in the scene, giving the user a better understanding of what to expect in the synthesised result. Another improvement of using the interface, compared to the textbox is the controllability provided over the scene having or not having a background, determining the addition of contextual objects, and the extent of relevant contextual objects. Lastly, the interface also enables the user to generate a scene by providing objects as predicates to an *Or* query. An example of using the user interface to describe the specifications of the scene is depicted in Figure 3.10. The figure shows the objects in the *Quantity Table* with the number of objects. The *Relation Table* specifies the spatial relations between the chair and desk, and the desk and bed. Furthermore, the attributes of the table are also visible in the *Properties Table*. The attributes were added by selecting the object from the *Object Scene* list and checking the required item from the *Attribute* list. An instance of this is also visible where the chair's *Office* attribute is selected. Additionally, the parameters of the scene are also determined to not have contextual objects or a room as a background.

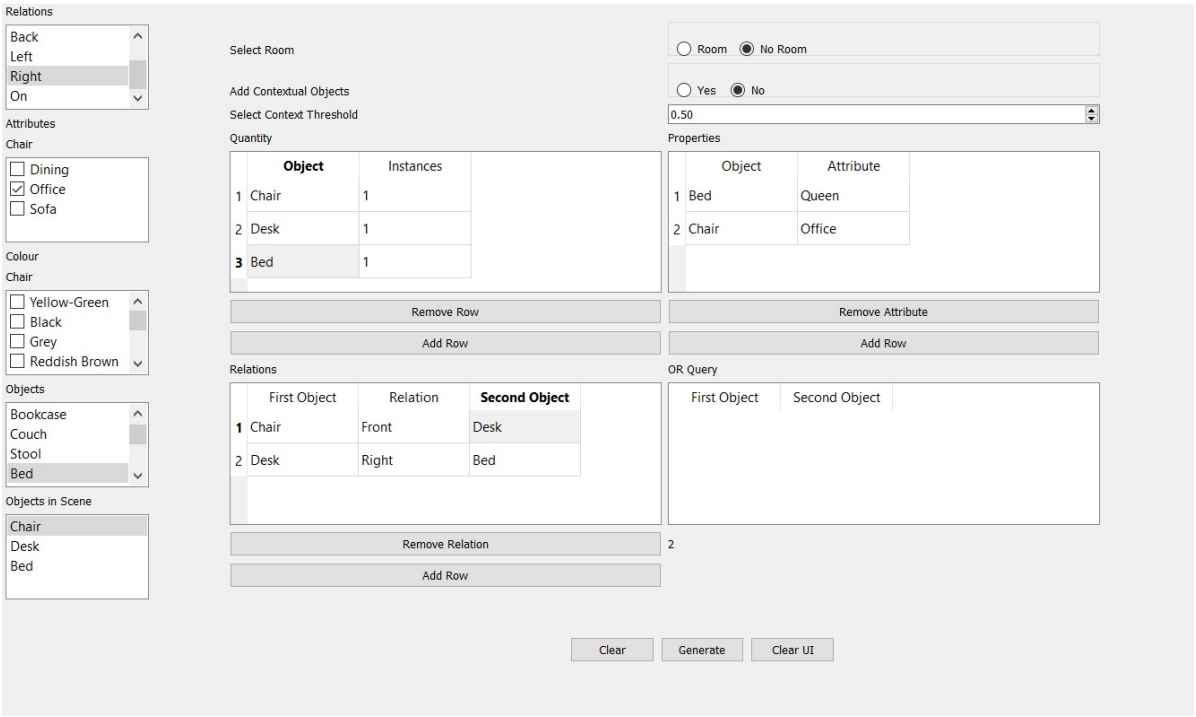


Figure 3.10: User Interface describing a scene with an Office Chair, Desk, and a Queen Bed.

3.8. Query Generated Scene

An important requirement in the development of SceneUI is to have a component where a query is specified, containing objects as predicates. The image generated from this query would be used as ground truth for specialised machine learning models. As earlier works do not explore generating indoor scene images from queries, we limit the scope of our thesis to using *OR* queries with two predicates. In this way, we create a baseline for generating images based on a query and also show that it is possible to synthesise images from a query. We leave the scope of including multiple predicates in the queries and more complex queries with conjunctive normal form such as - Table *AND* (Chair *OR* Stool) for future work. Based on the *OR* query, the scene synthesised should consist of either one or both the objects and the image extracted is used for the second use case as ground truth for specialised machine learning models.

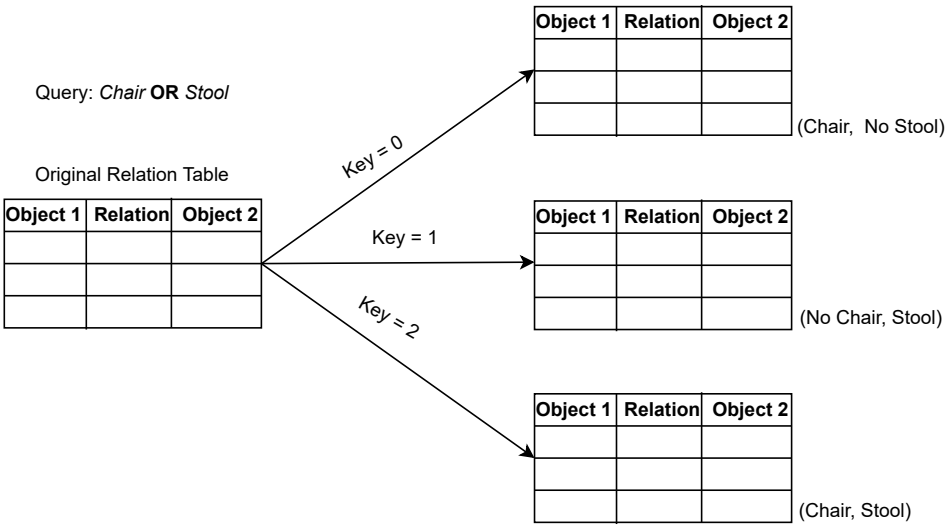


Figure 3.11: Process for generating an image given an *OR* query. Changes are made to the Scene Objects list and the Relations table based on the key value.

The input for the query is provided via the *OR Query Table* in the user interface: where objects required as query predicates are specified in both columns of the table. For generating the scene based on the query, we have three scenarios or possibilities of the image. The basic requirement for all scenarios is that at least one of these objects is present in the generated scene. Thus, given a query of a *Chair* ***OR*** *Stool*, the scene could have either a chair but no stool, a stool and no chair, or both a chair and a stool. When generating the scene based on the query, there should also be an equal probability of either of these scenarios occurring. Therefore, we generate a random number between 0 and 2, with this value denoting a key value. If the key generated is 0, the object in the first column of the *OR Query Table* is selected to be present in the scene while the second is excluded. This is ensured by iterating through the *Relation Table* and the *Scene Objects* list and removing occurrences of the second object as we do not need it in the scene. Furthermore, the *Scene Objects* list is searched to check if the first object is present in the scene. If it is not in the list, the list is incremented by adding the object. Conversely, when the key is 1, the object in the second column is selected in the scene and all instances of the first object occurring are excluded. A similar process is followed as the previous scenario to remove all occurrences of the first object and augment the *Scene Objects* list with the second object. When the generated key holds the value 2, both objects are present in the scene by checking if the *Scene Objects* list contains both objects. If any object is not present, it is then added to the list. Figure 3.11 illustrates the process described above where the changes are made to the relation table based on the key generated.

The procedure to generate a scene based on the query only requires modifications to relevant tables and lists to remove or include objects. Once the modifications are made, the rest of the pipeline is followed to create the input scene graph, find and modify similar scene graphs, and finally generate a scene containing at least one of the two predicate objects.

4

Experiments

This chapter describes the experimental setup conducted to evaluate SceneUI in the context of downstream tasks. Further details are provided on the settings of both use cases, their main requirements, and how images are synthesised for the use cases. The goal of the experiments is to show the usability of SceneUI itself for either task and how its features can be used to customise scenes created. The purpose of the experiment described in section 4.1 is to create biased and unbiased image datasets using SceneUI to observe how the inclusion of various biases in the training set influences the classification process of a neural network. Furthermore, it should allow controllability over the biases induced. An additional goal is to investigate if the computer vision model can learn the biases using the dataset of images created by analysing the saliency maps on the test dataset. In section 4.2 we detail the experiment to evaluate SceneUI by providing a query as an additional input in the user interface. Classes are assigned to the images based on the query and a machine learning classification task is performed. An additional goal is to show that the images generated from the framework can be used to train machine learning models and that the model can learn the features and perform with reasonably good accuracy.

4.1. Bias Detection

This experiment evaluates SceneUI by utilising its features for the use case of explaining deep learning models in the context of bias detection. Using the system, two training datasets are created - one biased and one unbiased with the test dataset also being unbiased. The extent of the biases learned by the model is analysed by comparing classification statistics and explanations of what the model focuses on via saliency maps on the test set, with the unbiased model. An unbiased dataset is also selected for training as this learns an unskewed and fair representation of the scene and assists with the comparison of the biased model to understand the depth of the biases picked up. Additionally, the test dataset is also unbiased as we are interested in observing the biased model's performance on a realistic and unskewed dataset. The description of the types of biases introduced in the experiment is provided in subsection 4.1.1, subsection 4.1.2 details how causal graphs are used to create an overview of objects to be included in each scene, and subsection 4.1.3 explains the image generation process. Finally, subsection 4.1.4 provides details about fine-tuning the neural network with the datasets which is a key part of the experiment.

Dataset Classes

The datasets consists of 3 classes - ***Study Room***, ***Dining Room***, and ***Living Room***. These classes were selected based on observations made on the existing scenes in the database as well as the object categories. Furthermore, on studying the variations within an object category such as different types of bookcases, dining tables or study desks, the three classes were decided as it would be possible to create many variations of a realistic and representative scene. Additionally, 3 classes were decided to create diversity in biases introduced and also increase the complexity of the classification task by avoiding a simpler binary classification. In the biased dataset, each class has its own unique bias based on an object's property or co-occurrence with other objects. On the other hand, the unskewed or natural

dataset contains a more uniform distribution of objects with also an overlap of objects present that typically belong to other classes. The images for each class whether biased or natural were ensured to be a depiction of a scene representative of that class and where a human could unambiguously identify the scene or class the image belongs to. An example of this is the study room always has a study desk, a dining room will have some type or shape of a dining table and a living room should have furniture to sit on, such as a couch or a sofa chair.

4.1.1. Types of Biases

There are three biases introduced in the training set with each class having their custom bias. These biases were selected for the experiment to show that SceneUI allows the user to introduce a variety of biases in the dataset by fixing attributes or spatial relations of objects via the interface, thereby generating images where the objects hold these attributes. An overview of the biases found in each class as well as the biased objects are depicted in Table 4.1.

- The Study Room class comprises images that have an **object co-occurrence bias** where in all the images a speaker or a monitor always exists on a study desk. This means that there are instances of either object present on the desk or both objects. Among all the other objects in the scene, either of the two or both objects is always present. The purpose here is to see whether the model learns a study room as a scene where a speaker or monitor is always present on the study desk. SceneUI is used to constrain the speaker and monitor to be placed on the study desk. An example of a biased study room is shown in Figure 4.1.
  Additional objects included only in this class of the biased dataset are those that are typically found in study rooms: office chair, keyboard, computer mouse, mousepad, desk lamp, printer, pencil, paper.

- The Dining Room class contains images that depict a **colour bias**. The recurring theme of images in this class is that a **red** object should always be present in this class. The dataset of object models has only a single type of a red dining table. Steps were taken to ensure that the model learns the colour red and not just the same table or chair. This was done either by changing the colour or types of chairs in the dining room but keeping a red table present or by having red chairs with a different dining table. Additionally, images from scenes were also selected that had smaller red objects on the table such as a red apple or a red plate. Using SceneUI, attributes of objects were selected that exhibit a red colour. A sample image is illustrated in Figure 4.2.
  Additional objects included only in this class of the biased dataset are those that are commonly found on a dining table: bowl, mug, plate, knife, cup, food, bottle, fruit, pizza, donut.

- The Living Room class consists of images with an **object bias** and a **part-of relation bias**. The images in this class always include scenes with a glass coffee table and on other occasions with a bookcase having 3 shelves. The part-of relation bias mentioned above is a bias where the model learns of a bookcase as an object that contains only 3 shelves. It would further be interesting to see if the model still identifies a bookcase with a different number of shelves as a representative of the living room. SceneUI allows the selection of the number of the bookcase's shelves and the type of table. Figure 4.3 depicts a biased scene from this class.
  Additional objects included only in this class of the biased dataset are those that are typically found on a dining table: couch, sofa chair, plant, knife, TV, TV stand, rug, cabinet.

4.1.2. Generating Causal Graphs

To create varying images with diverse objects in the scene, we need a distribution of objects for each class. Given the number of samples for a class, the distribution of objects for each sample is produced using causal graphs. The causal graph creates a representation of a class, the objects that occur in this class as well as the properties of the class. The class, objects and their properties are all denoted as nodes, and the edges connect the objects to the class and the properties nodes are also connected to their respective objects by edges. The edges are weighted with a value that denotes the probability of the object or property occurring. Overall, it is an intuitive way to represent the objects and their properties that should occur in a class. This helps in generating the skews and biases we need for the biased datasets, such as how often an object should occur in the dataset and how often should it

Table 4.1: Categorisation of the biases, biased objects and properties included for each class.

Class	Biased Objects	Object Property	Type of Bias
Study Room	Speaker	-	Object Co-occurrence Bias
	Monitor	-	
	Desk	Study	
Dining Room	Table	Dining Red	Colour Bias
	Chair	Red	
Living Room	Table	Coffee Glass	Object Bias
	Bookcase	3 shelves	Part-of Relation Bias

Figure 4.1: Biased Study Room with a speaker and monitor on the desk.

Figure 4.2: Biased Dining Room with a red dining table.

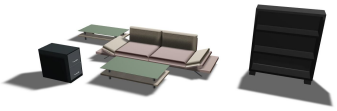


Figure 4.3: Biased Living Room with a glass coffee table and a 3 shelf bookcase.

exhibit the given property. An example of a causal graph for a study room is depicted in Figure 4.4.

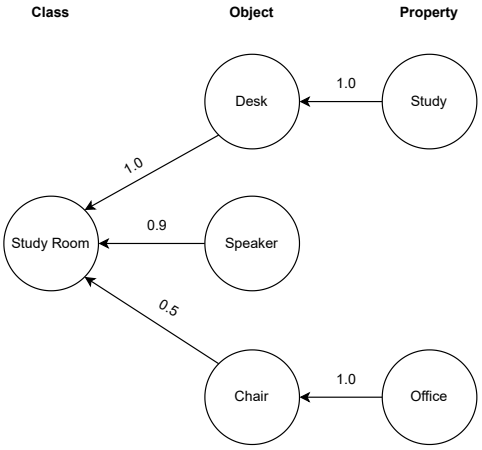


Figure 4.4: Causal Graph for a Biased Study Room.

For example, in the Study Room, the probability of a desk occurring is 1.0 and also very high for a speaker or monitor as shown in Figure 4.4. Similarly, for the dining room, the probability of the chair or table to have the colour *Red* is set to be very high. The other objects occurring in the scenes of the biased datasets are assigned a probability of 0.5. On the other hand, for the unskewed dataset, the probabilities of various objects occurring in respective scenes are assigned based on how one would expect them to be, given real-world observations. The distribution of objects produced for instances of each class is generated using the causal graphs from MirrorDataGenerator [1] [58].

[1] https://github.com/DataResponsibly/MirrorDataGenerator

4.1.3. Generating Images

Given the distribution of objects for each biased class and the natural datasets, the next step was to make use of SceneUI to create images for the biased and both natural datasets. To ensure that the model learns the features of each class well, 50 images were generated for each class in the dataset, thereby also having a uniform distribution. This resulted in 150 images each for the biased and natural datasets which also limits the computational time for the experiments. SceneUI was used to select specific attributes of the objects in the scene, and their quantity and constrain them to particular spatial relations. This was carried out by using the interface described in section 3.7.

Selecting Attributes from the Interface

By selecting attributes for the objects in the scene from the user interface, the classes can be created such that the objects in it can conform to the biases described earlier as well as create a realistic visual of a scene that is representative of an actual room corresponding to the class. The list of objects and their corresponding attributes selected via the user interface for each class can be found in Table 4.2.

Table 4.2: Attributes of objects selected for each class with the user interface.

Class	Object	Attribute
Study Room	Desk	Study
	Chair	Office
Dining Room	Table	Dining
		Red
	Chair	Red
Living Room	Bookcase	3 shelves
	Table	Coffee
		Glass
	Chair	Sofa

Selecting Spatial Relations from the Interface

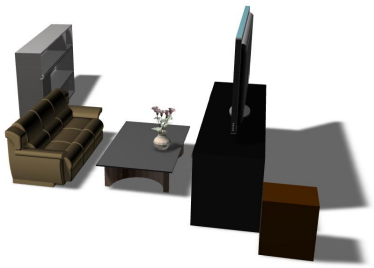
The interface is also utilised to impose spatial relations between objects. This helps make the scene more realistic and avoids unnatural cases such as objects floating in mid-air. To ensure this, smaller objects that are expected to be on a surface are constrained with the *On* relation to being on top of the larger surface. For example, the Study Room class would require the monitor, speaker, mouse, keyboard, mousepad, etc. to be placed on the desk. Similarly, images depicting a Dining Room are expected to have food items and cutlery on the table. Additionally, spatial relations were applied between objects to place objects at a specific position with respect to the other. An instance of this, is the office chair placed in front of the desk in the study room, and in some images of the living room, the TV is located in front of a coffee table which is also placed in front of the couch.

Image Generation Procedure

Every instance of the result obtained from the causal graph provides a distribution of the objects that should be present in every scene. Given the objects to be instantiated in the scene and their properties, the attributes for objects and spatial relations between objects (if needed) are selected and provided as input in the user interface. To create multiple images of the same scene, variations of the scene are made by camera rotations. The scene is also created without a background which further enables obtaining multiple images of a scene from varying angles. The camera rotation function takes user input via a key press, rotates the view of the scene and takes a screenshot of the window which serves as the image. By repeating this action five times, five images of the scene are generated from various angles, till a full rotation is complete. In this way, the process is scaled by providing various inputs of different scenes and obtaining five images of the scene. Finally, 450 images are created, with 150 each for the biased dataset for training, one natural set for training and another natural dataset for testing. Each class in the dataset comprises 50 images. Figure 4.5 illustrates examples of various images used

(a) Biased Dining Room.

(b) Biased Study Room.



(c) Biased Living Room.

(d) Biased Living Room.

(e) Natural Living Room.

(f) Natural Study Room.

Figure 4.5: Examples of images used for different classes of the three datasets.

for different classes of the train and test datasets. Figure 4.5a, Figure 4.5b, Figure 4.5c and Figure 4.5d are examples of images from the biased dataset. Figure 4.5a depicts a dining room with a red table and red chairs, with a knife cup, fruit and bottle on the table. Figure 4.5b depicts a biased study room with a monitor, speaker, keyboard, printer and mousepad on the desk, with an office chair in front of it.

Figure 4.5c and Figure 4.5d are images from the same scene of a biased living room but with different camera angles. The images constitute a couch adjacent to a bookcase with 3 shelves, and in front of a coffee table with a plant on it, a TV on a TV stand and a cabinet. Figure 4.5e and Figure 4.5f are both images from unskewed datasets belonging respectively to the Living Room and Study Room classes. Figure 4.5e is an instance of the natural training dataset and Figure 4.5f belongs to the test set.

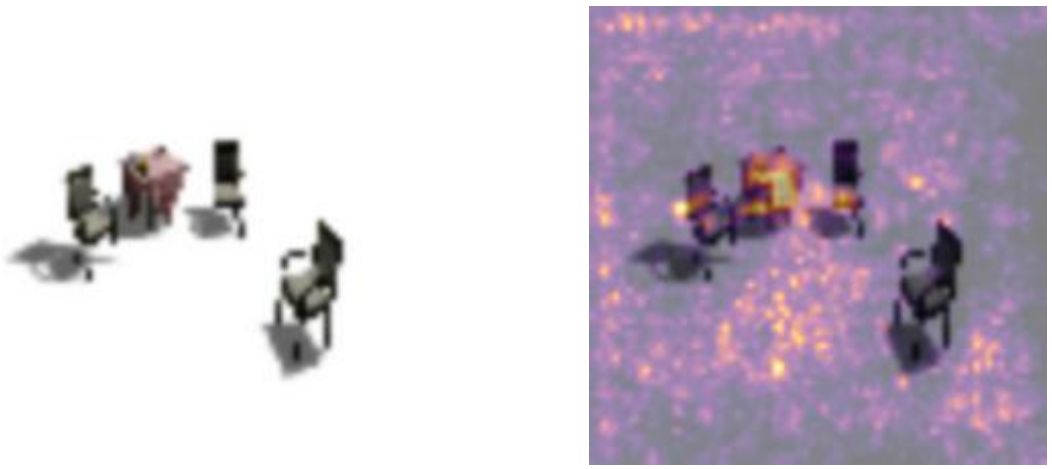
4.1.4. Fine-tuning the Neural Network

Following the creation of the datasets, the next step is to fine-tune a deep learning model on both the datasets assigned for training and observe the results on the test dataset. Fine-tuning a neural network is the process of making a pre-trained model learn the features of a new dataset that does not greatly differ from the data it is pre-trained on. Learning new features of the dataset and reducing the number of classes to predict helps in customising the network for our classification task. For this experiment, the ResNet50 neural network is selected which is already pre-trained on the ImageNet dataset: a database with millions of images that have 1000 categories [12].

Training Parameters

The setting and parameters of the ResNet50 were the same for training both datasets. Furthermore, to limit computation time the images were resized to a height and width of 75 pixels respectively before training. The following parameters were selected:

- Epochs - 75
- Batch Size - 125
- Optimizer - Adam
- Loss - Binary Cross Entropy
- Early Stopping - Patience value of 5 on validation accuracy



(a) Dining Room. (b) Saliency Map for a Dining Room.

Figure 4.6: Saliency map generated for an image. The highlighted region in the saliency map denotes attributes or areas the image finds important for classification.

Generating Saliency Maps

The last step is to generate saliency maps of images from the test dataset. The saliency map of the image highlights the regions in the image that the fine-tuned network focuses on and finds important for classification. The saliency maps on the test set are generated for both the biased and unbiased models trained earlier. For every image in the test set, the trained model highlights the pixels in the image it considers important for classification using a SmoothGrad mask [48], while also considering the predicted class for the image. This mask of highlighted pixels is converted to a heatmap and overlayed

on a grayscale conversion of the image. This combination of layers results in the saliency map for a given image that shows the objects or regions the model considers essential for classification. An example of a saliency map generated by the biased model for a *Dining Room* is shown in Figure 4.6b. The model highlights the region of the dining table that it considers important for it to belong to the *Dining Room* class.

4.2. Query Generated Images

In this experiment, the *OR* query feature of the interface is used to generate images. A supervised classification setup is designed to predict two classes given an image - ***OR*** and ***Not***. Images in the *OR* class should have either a dining chair, a dining table or both objects together, amongst other objects in the scene. In contrast, images from the *Not* class never contain a dining chair or dining table. The purpose of the machine learning classification task is to discriminate between scenes or images that have either of these objects or none of them. To ensure that at least one of the objects is present the query input feature is used. This also helps to show that SceneUI can generate images when given a complex query with objects specified. The query predicates - dining table and dining chair are selected due to the variety of objects present in the database holding these attributes. Another reason for selecting this category of objects is that both are present in the COCO dataset, a dataset with common objects in context [32].

The dataset for the experiment is created in a way that there is an equal distribution of images for all four cases - scenes with only a chair, scenes with only a table, scenes with both a chair and table and scenes without a chair or table. However, as the images without a chair or table constitute a class of their own and the former three make up the *OR* class together, the overall dataset is skewed towards *OR* class. This section is further divided into two key subsections describing the experiment. In subsection 4.2.1 the entire image generation process to create the dataset is described, and subsection 4.2.2 details the machine learning classification task.

4.2.1. Generating Images

The image generation process is to an extent similar to the experiment described in section 4.1. The main attributes to be selected here are the chairs and tables to both be dining when they occur. Additionally, to increase the variety of images, tables were specified to be a rectangular or round shape. Small objects were also constrained to spatial relations and were placed on top of objects like a bookcase, TV stand, desk or table by specifying this through the user interface. A notable difference from section 4.1 in the image generation process is that the objects in the scene are in a room and thus have a background. The presence of the room is also selected via the user interface. Another difference is the variation of images of a scene is not generated by having different camera angles, but by often having the same object in different backgrounds. This difference also shows that SceneUI can be used to create a variety of images, either with a variety of backgrounds or without.

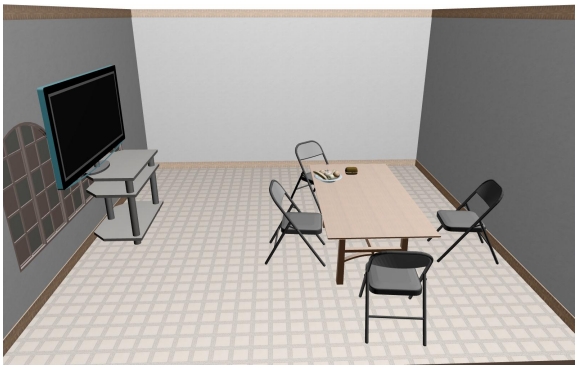
Image Generation Procedure

The images are created by providing the query through the interface, with the query predicates being **Chair** and **Table**. In the attributes section of the interface, both objects are specified to be of the property *Dining* when they occur. To generate images for the positive class (*OR*), the predicates were provided as input and the scene was created. The next step was to manually adjust the camera angle and view of the scene to ensure that the image extracted had the objects appropriately with the room as a background. A set of 150 images for the positive class was generated in this way, with an equal distribution of 50 images each for scenarios with only a chair, only a table, and both chair and table. Apart from the chair and table, other objects were also introduced in the scene such as a desk, bed, bookcase, couch, TV, TV stand, cabinet, nightstand, dresser, vanity, bathtub, speaker, monitor, desk lamp and food items. Images for the negative class *Not* were collected using a similar process, but without any object provided in the *OR* query specification. Using a combination of the additional objects previously mentioned, scenes were created and 50 images corresponding to the *Not* class were generated. To generate images for this class, it was important to make considerations to never instantiate a dining chair or a dining table in the scene. 50 images were selected for each scenario to give

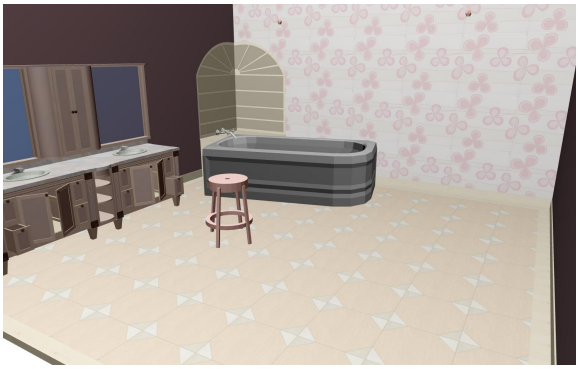
the machine learning model sufficient examples to learn the features for each class. Figure 4.7 provides examples of images generated through the queries for both classes for all 4 scenarios and also illustrates the possible backgrounds for the room that are possible to have. Figure 4.7a, Figure 4.7b, Figure 4.7c belong to the positive *OR* class and Figure 4.7c belongs to the negative *Not* class.

(a) Scene with only a chair.

(b) Scene with only a table.



(c) Scene with a chair and table.



(d) Scene without a chair or table.

Figure 4.7: Examples of images generated for all scenarios based on objects specified as a query.

4.2.2. Image Classification

With the 200 images generated, the next task was to perform a supervised classification task. The classification task is a simple binary class prediction and for this purpose, a simple, fast but powerful machine learning model - the Support Vector Machine (SVM) classifier was used. The dataset is split by assigning 120 images to the training set and 80 images to the test set (60-40 split), in a stratified way to maintain an equal proportion of class distribution due to the imbalanced nature of the dataset. The classification task of the experiment was further conducted by training the SVM classifier with 4 different kernels and comparing the results. This comparison also indicates which kernel performs the best for images generated using SceneUI. Using a GridSearch, the regularisation and gamma parameters for every classifier were tuned to optimise the highest accuracy. By selecting the best parameters for a kernel for the task, we can compare the best case results for each SVM kernel. Table 4.3 shows the SVM kernels used in the experiments for classification and the resulting best parameters after the GridSearch.

Table 4.3: Parameters for SVM kernels using GridSearch.

Kernel	**C**	**Gamma**
RBF	100	0.0001
Poly	0.1	0.001
Linear	0.1	0.0001
Sigmoid	0.1	0.0001

5
Results

This chapter presents the results obtained after conducting both the experiments detailed in chapter 4.

5.1. Bias Detection

The results from the experiment conducted in section 4.1 are presented in this section. The ResNet50 network fine-tuned on the biased dataset provides an accuracy of **58.0%** with the test dataset, and the network fine-tuned on the natural dataset resulted in an accuracy of **66.67%** with the test dataset. Analysing the confusion matrices obtained as a result of the classifier's performance on the test set, gives insights into the proportion of misclassification when trying to classify each class. The confusion matrix summarises and provides information on the classification performance of the classifier [50]. It also gives information on the number of times the classifier predicts correctly and incorrectly when trying to classify a particular label.

Analysing Confusion Matrices

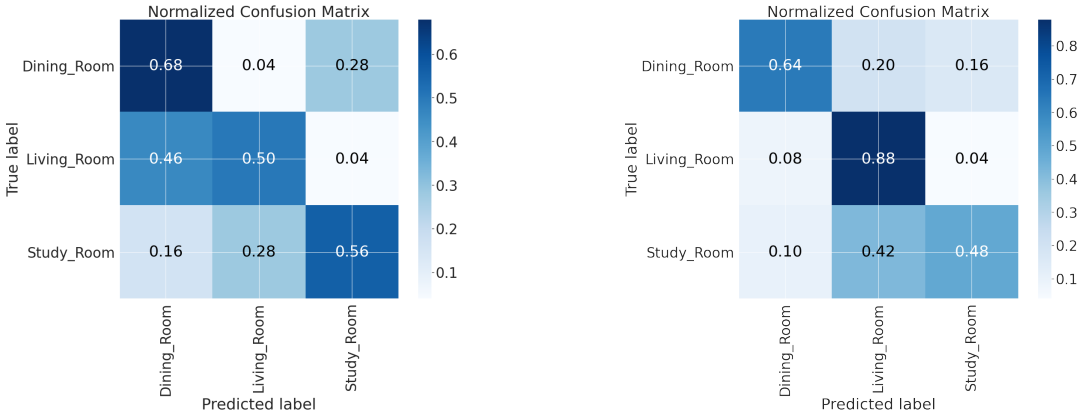
Figure 5.1 illustrates the normalised confusion matrix of the classifier's performance when trained on the biased set and natural set respectively. A notable difference between the two confusion matrices, is the disparity in misclassification rates. This difference indicates that the biased model in most cases is inclined to misclassify one class over another due to the biases learned. When compared class by class, the following differences were found :

- Dining Room - Figure 5.1a shows a difference between incorrectly classified classes as 0.24 and 0.04 in Figure 5.1b
- Living Room - Figure 5.1a shows a difference between incorrectly classified classes as 0.42 and 0.04 in Figure 5.1b
- Study Room - Figure 5.1a shows a difference between incorrectly classified classes as 0.12 and 0.32 in Figure 5.1b. This is the only scenario where the discrepancy in incorrect classification between classes is higher for the classifier trained on the unskewed dataset.

From the two trained models, two sets of saliency maps were generated on the test set which also contains images resized to (75,75). The saliency maps from the test set were analysed to see whether the highlighted regions of the image correspond to the biases introduced for the class, while also checking the class prediction or misclassification for the image highlighted with saliency maps.

Misclassifying Study Room

Figure 5.2 depicts examples of a *Study Room* being misclassified due to induced biases in the training set. Figure 5.2a contains a desk with a mousepad and office objects on it, an office chair in front of it and a couch behind the chair. The saliency map in Figure 5.2b strongly highlights regions of the couch, which is what it strongly focuses on during prediction. Figure 5.2a was incorrectly classified as a *Living Room* due to the presence of the couch. Similarly, Figure 5.2c consists of a study desk with a chair,



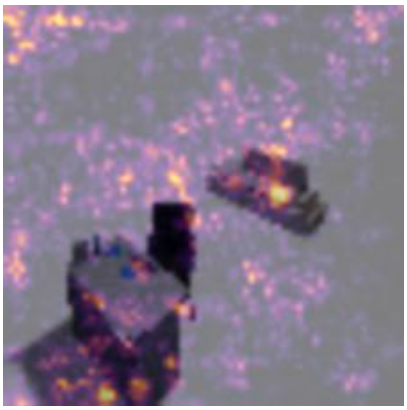
(a) Normalised Confusion Matrix - Biased Classifier.

(b) Normalised Confusion Matrix - Unbiased Classifier.

Figure 5.1: Normalised Confusion matrices for Biased and Unbiased models.

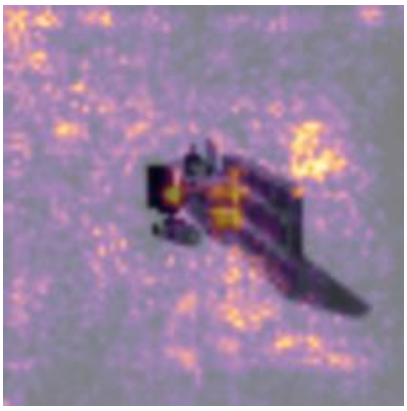
monitor and speaker, adjacent to a bookcase with 3 shelves. Due to the presence of this bookcase only in the *Living Room* class, the image is classified as a *Living Room*. Figure 5.2d gives an indication of the model highlighting the shelves of the bookcase.

(a) Study Room with a couch.



(b) Saliency Map of the Study Room scene.

(c) Study Room with a 3 shelf bookcase.



(d) Saliency Map of the Study Room scene.

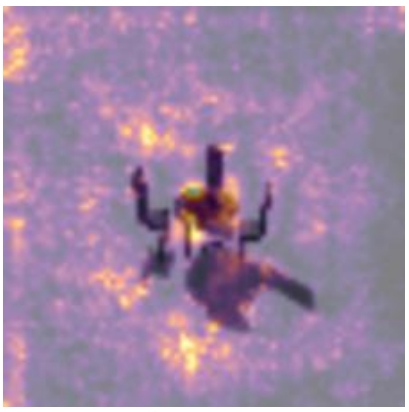
Figure 5.2: Images and respective saliency maps when predicting a Study Room with the biased model.

Misclassifying Dining Room

Figure 5.3 illustrates examples of the dining room being misclassified due to biases originating from the *Study Room*. Figure 5.3a is an image with a dining table, 3 dining chairs and a monitor on it. This

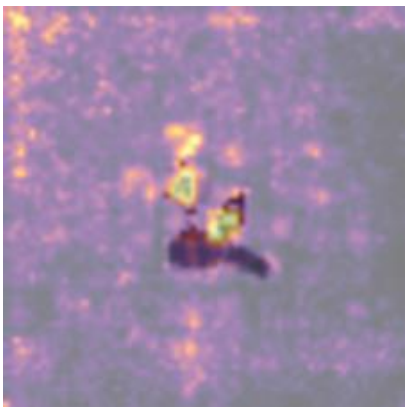
image was classified as a *Study Room* due to the presence of the monitor on the table. The saliency map in Figure 5.3b highlights the monitor as well as the regions on the table it is located. Another example shown is Figure 5.3c which is also classified as a *Study Room* since the desk lamp occurs only in study rooms in the training set. An additional explanation that contributes to this classification could be that the image contains only a single chair, which is very often the case in the study rooms in the training set. Figure 5.3d provides evidence of this by highlighting the desk lamp as well as the dining chair.

(a) Monitor on a dining table.



(b) Saliency Map of the Dining Room scene.

(c) Desk lamp on a dining table.



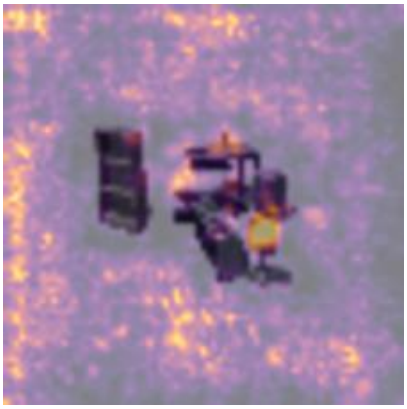
(d) Saliency Map of the Dining Room scene.

Figure 5.3: Images and respective saliency maps when predicting a Dining Room with the biased model.

Misclassifying Living Room
Finally, in Figure 5.4 images of a *Living Room* are shown which are misclassified as a *Dining Room* owing to the presence of red objects. Figure 5.4a is classified as a *Dining Room* due to the red chair in the scene, even though other objects typically found in the *Living Room* class of the training set are present, which is supported by the saliency map shown in Figure 5.4b. Similarly, Figure 5.4c comprises a red table and a red sofa chair amongst other objects. On observing Figure 5.4d, it can be seen that the model focuses on the red aspects of the dining table as well as the sofa chair. Therefore, it is classified as a *Dining Room*. An interesting insight here is that the red sofa chair is not present in the training set, and by highlighting the red portion of the chair in the saliency map, it shows that the model has truly learned the colour bias introduced in the *Dining Room* class.

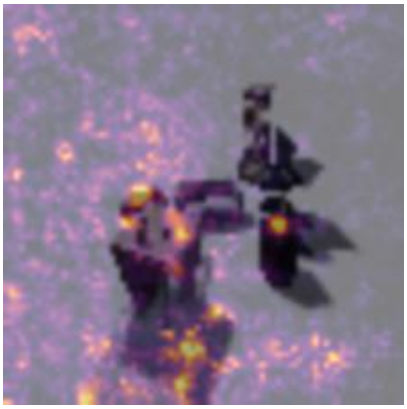
Analysing the Unbiased Model
An analysis conducted of the neural network trained on the natural dataset shows that the model can learn the features of the objects which is also supported by the saliency maps. The saliency maps highlight object properties such as the table's surface when predicting a room, and also focus more on a chair's features as well as notices a bookcase's shelves. This is not always the case for the biased model which identifies biased objects and categorises the image accordingly. For instance, the biased

(a) Living Room with a red chair.



(b) Saliency Map of the Living Room scene.

(c) Living Room with a red table and red couch.



(d) Saliency Map of the Living Room scene.

Figure 5.4: Images and respective saliency maps when predicting a Living Room with the biased model.

model classifies almost every image with a red object as a *Dining Room* while the unbiased model does not consider the colour but the objects present in the scene, based on the scenes in its training set. The presence of biased objects sometimes helps the biased model to correctly predict images from the class the biased objects originate from: a red table in a dining room assists the model in classifying it as a dining room. As the unbiased model does not know these biases, it focuses on the objects present or the combination of objects, which could also explain the slightly lower true positives corresponding to the *Dining Room* and *Study Room* classes when comparing the confusion matrices shown in Figure 5.1a and Figure 5.1b.

Bias Induced Misclassifications
Of the three biases introduced, the colour bias is learned best by the model. From the test dataset itself, 8 *Study Room* images were predicted as a *Dining Room* from which 7 images had a red object. Similarly, of the 23 *Living Room* images misclassified as a dining room, 16 had a red object. 14 *Dining Room* images were incorrectly predicted as a *Study Room* where 4 of these had a monitor on the table, 1 had a speaker, and the other misclassifications were due to the table's colour being similar to that of the desk, or the presence of a desk lamp. Due to a slightly more complicated bias introduced, of the 14 *Study Rooms* classified as a *Living Room* 6 included images with a bookcase having 3 shelves, while the rest were misclassified due to the presence of a couch in the scene that was only found in the *Living Room* class of the training set. An overview of these numbers can be found in Table 5.1.

Conclusion of the Experiment
From an analysis of the results, a suggestion would be to create biased datasets such that the biases introduced should be based either on a colour or the absence or presence of large objects similar to a couch or a bookcase. SceneUI allows the selection of these attributes (colour) as well as instantiating large objects in the scene, as also shown in this experiment. Thus, it can be concluded from the experi-

Table 5.1: Overview of the misclassifications cause due to the presence of biased objects or properties.

Bias	Misclassified Class	True Class	Misclassified Images	Misclassified Biased Images
Red Object	Dining Room	Study Room	8	7
Red Object	Dining Room	Living Room	23	16
Monitor	Study Room	Dining Room	14	4
Speaker	Study Room	Dining Room	14	1
Bookcase	Living Room	Study Room	14	6

ment that the neural network can learn biases deliberately introduced in the training set using SceneUI. However, the biases learned vary to different extents based on the type of bias introduced. Additionally, by generating an unbiased, natural dataset using SceneUI, and comparing the explanations of the model trained on this dataset with the biased model, explainability methods can be benchmarked. Therefore, the results from the experiment show that SceneUI can be used to create both biased and unbiased datasets. Its features also allow the selection of various objects and their properties such as their functionality or colour. The neural network is able to learn the biases introduced via SceneUI and can also be compared with an unbiased neural network.

5.2. Query Generated Images

The results from the experiment conducted in section 4.2 are presented in this section. The analysis involves a comparison of four different SVM models, trained on different kernels. To distinguish the performance of different models and to identify the best performing kernel for this setting, the comparison was made by considering the following classification metrics:

- Accuracy - Accuracy is the ratio of a classifier's correct predictions, to the total number of predictions made. Overall, it denotes how good a classifier is in predicting either class.
- Recall - Recall is the ratio of the correct predictions of the class to the total number of instances of that class [38]. A high recall score signifies that the classifier has a low probability of predicting the other (negative) class.
- Precision - Precision is the ratio of the correct predictions of a class, divided by the total number of times that class is predicted [38]. It indicates the performance when the classifier is required to predict the specific class.
- ROC - Receiver Operating Characteristic (ROC) is a plot of a classifier's True Positive Rate (ratio of correct predictions of positive class to the number of instances of that class) against the False Positive Rate (ratio of incorrect predictions of a positive class to the number of instances of the negative class) [15]. It denotes how well a classifier can distinguish the two classes.
- AUC - The Area Under the Curve (AUC) of the ROC curve plot is extracted as a singular value metric to compare the performance of different models.

These metrics are selected as the scores help quantify the model's performance over various aspects. A model that performs well in most if not all of these metrics would be judged as the best performing model for the task.

Classification Results

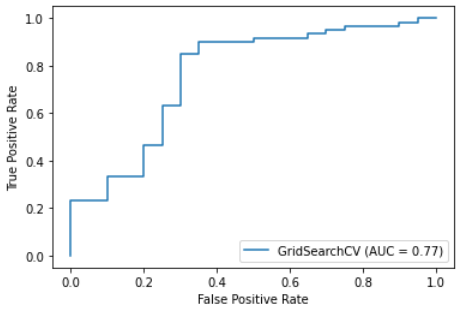
The scores from these metrics are presented in Table 5.2 with the columns containing values in bold denoting the kernel that achieves the highest score for that metric. The table also contains precision and recall scores for both classes, which helps us understand the model's performance when predicting both classes. 0 indicates the negative class, that is *Not* while 1 corresponds to the positive *OR* class.

From the table, it can be observed that when the SVM is trained with the RBF kernel, it outperforms the other kernels over all metrics except recall for the positive class. The ROC plots for all the kernels

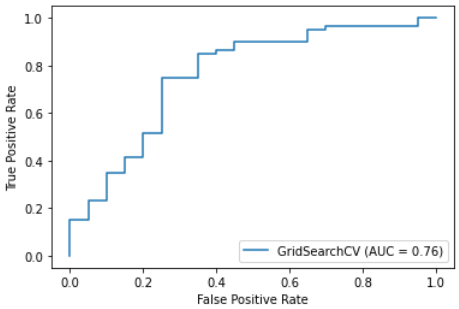
are also illustrated in Figure 5.5. Though there is a minor difference between the AUC values and the plots of Figure 5.5a, Figure 5.5b, and Figure 5.5c is minor, the RBF kernel gives the best performance. A reason why the Sigmoid kernel achieves a low AUC score, recall and precision values for the negative class is because it predicts all instances as the positive *OR* class. Due to the imbalanced proportion of the dataset, it still has a decent accuracy of 75% but very low scores for precision when predicting the negative class. Due to the limited sample size and the imbalanced skew of the dataset, the recall and precision scores for predicting the negative class are low, especially for recall. This indicates that the classifier often incorrectly predicts the image to be of the *OR* class whereas its true class is *Not*.

Table 5.2: Performance of different SVM kernels over 4 metrics.

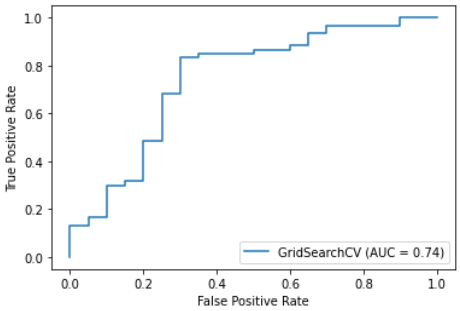
	RBF	**Poly**	**Linear**	**Sigmoid**
Accuracy	**0.7875**	0.7625	0.7625	0.75
Precision	0 - **0.64** 1 - **0.81**	0 - 0.54 1 - **0.81**	0 - 0.54 1 - **0.81**	0 - 0.00 1 - 0.75
Recall	0 - **0.45** 1 - 0.87	0 - 0.35 1 - 0.90	0 - 0.35 1 - 0.90	0 - 0.00 1 - **1.0**
AUC	**0.77**	0.76	0.74	0.48



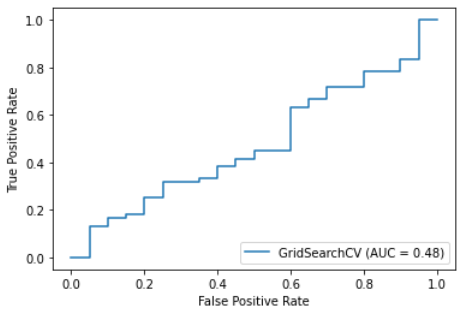
(a) ROC curve for RBF kernel.



(b) ROC curve for Poly kernel.



(c) ROC curve for Linear kernel.



(d) ROC curve for Sigmoid kernel.

Figure 5.5: ROC plots for various SVM kernels used in the experiments.

Conclusion of the Experiment

Thus, from the comparison of metrics, it is observed that the RBF kernel performs the best and can perform reasonably well and can distinguish between the two classes. With more images in the dataset and a more balanced class distribution, the model could also perform better. From the experiment, it can also be concluded that a variety of images can be generated using the query input feature of SceneUI. The scenes are also synthesised in different backgrounds, giving further diversity in features to be learned and also making the scene more realistic. The performance of all the SVM kernels on the dataset also shows that machine learning models can learn the features of the images created as a result of SceneUI and that the images generated are suitable for machine learning tasks and therefore can be used as ground truth.

6

Discussions & Conclusions

In this chapter, we provide the conclusions of the thesis in section 6.1 with a summary of the thesis and answers to the research questions. Additionally, we analyse and discuss SceneUI by comparing it with relevant literature in light of the use cases. We review the limitations of SceneUI in section 6.2 in the context of the method and experiments conducted. Future research directions for development are also presented in section 6.3 to improve the system and ensure its scalability to generate numerous images.

6.1. Conclusions

In this section, we provide the conclusions of the thesis by summarizing our approach, experiments and results in subsection 6.1.1 and answering the research questions defined. Additionally, we discuss aspects of SceneUI and compare it with the literature reviewed.

6.1.1. Summary

In this thesis, we present SceneUI, a system to generate indoor scenes and extract images, where the user describes a scene by specifying the objects and their properties, spatial relations between objects, and the presence of a background. The system was developed by modifying an existing similar work based on scene graphs by merging it with a user interface. The features of SceneUI were designed based on the requirements for two use cases the images will be used for - *Generating Images for Interpretable Machine Learning* and *Generating Images from Queries as Ground Truth*. In this context, we increase the level of controllability provided in the existing work by expanding the attributes in the scene graph of the dataset so that they cover more properties of an object as well as its colours, and also introducing more objects that hold properties. Additionally, we design an interface that allows the user to select and control several aspects of the image including contextual elements and in this way combines various controllable image generation methods. The interface also consists of a feature where the scene is generated based on an *OR* query provided with objects as query predicates. Additional advantages of using SceneUI over the base work are that it reduces the cognitive load on the user by providing an overview of all the relations in the scene, possible objects to choose from, and also allows the entire description of the scene at once.

We showcase the suitability of SceneUI and the usability of the generated images for both use cases by conducting two sets of experiments. The features introduced in the interface and the expanded attributes of objects were also used in the experiments to generate images, thereby showing our work's benefits and usability. In the first experiment, we create biased and unbiased datasets, with each class of the dataset having its own bias created using the interface based on the colour of objects or the presence of objects. Thereafter, a neural network was trained on the datasets and saliency maps of the biased model were investigated to see how well it learns the biases in the images. The analysis indicated that several of the model's misclassified images are due to the presence of biases in the images. The experiment, therefore, supports our goal to use SceneUI to create biased and unbiased datasets and benchmark explainability methods. The second experiment generates images using the query feature of SceneUI and also places the objects with a background. The performance of four

SVM kernels was compared by training them on a dataset of two imbalanced classes, where one class contains the objects specified in the query and the other class contains neither object. The performance metrics of the models highlight that the images generated from SceneUI using the query are suitable for machine learning tasks and so the images can be used as ground truth for proxy models.

Answers to Research Questions
As a result of the research, development and design of experiments in the thesis, we provide the following answers to the research sub-questions described in section 1.2:

- ***RSQ1:*** *What are the current methods of image generation and what features of the synthesised image do they allow control over?*
  A: The literature study in the Background chapter gives an overview and insights into Machine Learning and Computer Graphics based methods for image generation. Additionally, for indoor scene generation Table 2.1 gives an overview of the existing techniques in Computer Graphics and various controllability features allowed such as object direction, query, context, colour, spatial relations, quantity, background and use of scene graphs. Based on our analysis we identify a research gap where none of the methods allow the user to manipulate all these features. Thus, in this thesis, we select a base work [34] that already provides some of these features and can be improved to provide more controllability.

- ***RSQ2:*** *How can the attributes of objects be expanded to allow controllability over multiple properties?*
  A: To augment the dataset, observations need to be made on the existing scenes to identify potential objects and their attributes. This was achieved by observing recurring categories of objects and the variations in the object's colour, material or functionality. Once the objects are identified and categorised, the colours that they have in every scene are also noted. Finally, the scene graph of the dataset is augmented by creating a new attribute node with the given attribute or colour and connecting it with the relevant object. This gives the user an increased range of attributes for an object to choose from.

- ***RSQ3:*** *How can the user provide specifications for all the controllable variables in the image?*
  A: This question is answered by developing the user interface of SceneUI. It provides an overview of all the possible objects and relations and allows the user to select the objects, their properties and the spatial relations between images. Table and List widgets are extensively used to provide various functionalities in this interface. Additionally, we enable the selection of a room as a background in the image and provide the selection of a parameter and its value for the presence of contextual objects in the scene. Thus, an overview is provided of all the controllable elements that the user describes in the image.

- ***RSQ4:*** *How can images be generated given a query with objects as predicate terms?*
  A: Generating an image based on the *OR* query requires the consideration of 3 scenarios. A random key value between 0 to 2 is generated and based on this value, modifications are made to the relation table and the list containing objects expected in the scene. The modification ensures that either one or both objects are selected in the scene, given the key value. With this feature in the interface, images can be generated based on a query specified.

- ***RSQ5:*** *To what extent can the method be used for the two use cases?*
  A: To evaluate SceneUI and its suitability to generate images, we evaluate it in the context of the two downstream tasks. The experiments for both use cases were designed based on detailed consultation and planning to simulate two scenarios for which the generated images would be used. In addition to showing how the images will be used for the use cases, we also discuss and present the advantages of using SceneUI and its features to synthesise the images. Our experiments show how SceneUI can create biased datasets by ensuring that the objects have a fixed property or a certain colour. Spatial relations are also enforced to ensure a specific arrangement of objects. Additionally, we generate images based on a query where the objects also have specific attributes (dining chairs and tables) and the objects are placed in a room as a background. In this way, a diverse range of images is synthesised.

6.1.2. Discussion and Comparison with Literature

A comparison of SceneUI with the relevant works in indoor scene generation reviewed in Table 2.1 shows that it satisfies the requirements stated in the table. As SceneUI is based on [34], it already meets certain requirements of controllability. Through this thesis, SceneUI fills in all its remaining requirements of controllability such as *Query*, *Colour*, and to an extent *Background*. Thus, SceneUI meets our initial objectives of controllable image generation, making it valuable for our use cases. However, one aspect of controllability missing in SceneUI is the ability to manipulate the scale of objects. SceneSeer [8] and WordsEye [11] enable the user to specify the size of an object and a corresponding large or small object is generated. Selecting the size of objects could be useful for the use cases by generating biased datasets where objects have a certain size or by creating ground truth images where objects have a given size. Another aspect where SceneUI falls short in comparison with SceneGen [27] from the Table 2.1 is the dataset of 3D models used by SceneGen. SceneGen uses the Matterport3D dataset [6] which contains more realistic models in texture, colour and shape when compared to the models from the SceneSynth dataset used in SceneUI. There is a strong preference for having realistic images as this data would be easier to use for complex deep learning models which are pre-trained on real-world data. For our use cases, the models for explainability or proxy models would be able to learn the features well of such realistic data, yielding better results. In contrast to SceneUI, BlenderProc generates numerous indoor images using realistic objects from the SUNCG dataset [49] and gives the user the possibility to generate images with various masks. However, BlenderProc offers a basic level of controllability in synthesising these images as it only allows the user to select the objects, image lighting and the size of images, thereby not being suitable for the use cases. This further underlines the importance of SceneUI's controllability features that generate diverse images to meet the objectives of our use cases.

6.2. Limitations

There are some limitations within SceneUI that we would like to address. Addressing these limitations gives the reader some directions where the system can be improved and considerations that would need to be made when developing similar systems. The limitations are described as follows :

- As the image generating procedure is based on a prior work [34], some of its limitations are inherited with SceneUI, one of which being rudimentary pairwise relations described in subsection 3.2.1. The seven pairwise relations can be expanded by modelling additional relationships such as *adjacent* or *inside*. Another limitation from the prior work is the lack of symmetry between objects placed. For example, if two lamps are specified to be placed on either side of a bed, the lamps may not be rendered symmetrically beside the bed [34]. This lack of symmetry results in the creation of less plausible or realistic scenes.
- An important feature that SceneUI does not have is the ability to generate images at scale. The images generated for the experiments require manually creating relations and attributes for objects via the interface and extracting screenshots of the scene. For the next set of images, the attributes, relations and objects expected in the scene would have to be adjusted. This is manageable and simple to do for a certain number of images but is not feasible to synthesise hundreds or thousands of images within a short duration of time.
- Another limitation related to the scalability of the method is the duration taken to generate images. SceneUI takes up to $\approx$6 seconds to load all the scene graphs and 3D models. If the dataset is augmented with new scene graphs and additional 3D objects, it could take more time to load. Additionally, based on the number and type of objects and relations specified in the input the renderer takes additional time to render the objects. A simple specification is quicker to load whereas a more complicated input with a room may take longer to generate and render due to object collisions and optimisations. This latency can be improved by increasing the efficiency of loading the dataset and the rendering of objects in the scene
- The interface allows the user to select a background as a room in the scene and the resultant image extracted will have the background. The interface enables the user to only select whether objects should or should not be placed within the room, and not the type of room to be selected in terms of the colour, size or design of the room. Selecting the 3D object that denotes the type of room would extend our current state of controllability.

- Another drawback of SceneUI is that when the context parameter to include contextual objects in the scene is set to *Yes*, the algorithm expects existing relation nodes corresponding to objects in the scene graph. In other words, the input should have specified the spatial relations between objects and this is considered a reference to introduce contextual objects. A failure case is registered if the user specifies only the objects to be initialised with random placement.
- The graph matching process described in section 3.5 finds the most similar scene graph from the dataset to the input graph and places the matched objects in positions specified in the dataset, resulting in a significant reliance on the dataset scene graphs. A complicated input specification could lead to an input scene graph with several unaligned nodes. This in turn could lead to random placement of objects with possibly an implausible layout due to a missing reference scene graph to place the given combination of objects. An increase in controllability through SceneUI can lead to intricate scene graphs synthesised as objects have more attributes to choose from. Thus, there is a tradeoff between having high controllability and plausible scenes as the most similar graph may have very few aligned nodes.

6.3. Future Work

The limitations discussed in section 6.2 provide gaps in the existing work which could be improved on to make the system better. The main limitation of SceneUI is its scalability which could not be improved due to the time constraints of the thesis. Thus, the following future research directions are envisioned and recommended to improve SceneUI's scalability in generating numerous images.

- Based on the first limitation discussed, scalability of generating images is a key issue to be resolved. This could be approached by providing an input file with all the instances of objects, their attributes, spatial relations and the number of images to be generated for the scene. Once the input scene graph from the specifications is created, the required images are extracted by varying the camera angle of the scene. This automation would require an in-depth understanding of the renderer's camera function, to vary the camera angle of the scene and save the image without any interaction. This direction would however require the user to forego the user interface designed for SceneUI.
- An alternate approach to generating numerous images is to crowdsource the generated images. This would require SceneUI to be deployed as an application on a crowdsourcing platform where the requester seeks respondents to describe a scene using the interface based on a particular setting such as a study room or a dining room. The user would then specify the objects, relations and object properties that they expect and visualise to be present in the setting of the room. In this way, several respondents can generate diverse images for a given room based on a description provided. This could be further extended by giving additional conditions that the scene should have, like the colour of a table or the number of chairs expected. In this way, the interface of SceneUI is also used to generate images.
- A future direction that supports both the scalability of SceneUI as well as its controllability would be to expand and include attributes for more objects than the existing ones augmented in the thesis. This would give more properties to choose from when selecting an object. This would involve a collaborative approach where scenes from the dataset are presented as a survey, highlighted with the objects to expand on. The objects could be identified by studying the scenes and determining recurring objects that show variations. The attributes of these objects would then be determined in the survey by collecting responses about the object's shape, functionality, material or colour, thereby increasing the controllable features. Furthermore, the possibility of merging other datasets based on indoor 3D objects with SceneSynth should also be explored to increase the variety and diversity of existing objects and scenes in the dataset.
- Another line of future work hinted in section 3.8 is to expand the query specification component of SceneUI. This would include more predicates for the *OR* query where atleast 2 objects can be specified and considering more scenarios with the combination of objects. Furthermore, complex queries with a combination of objects based on Conjunctive Normal Form should also be enabled, which would require a logic based solution to resolve the query and objects to be synthesised.

Appendix - Dataset Training Images

The images in the Appendix sections A.1, A.2, A.3 and A.4 correspond to the *Bias Detection* experiment and images in sections A.5, A.6, A.7, and A.8 are used in the *Query Generated Images* experiment.

A.1. Biased Dining Room Images

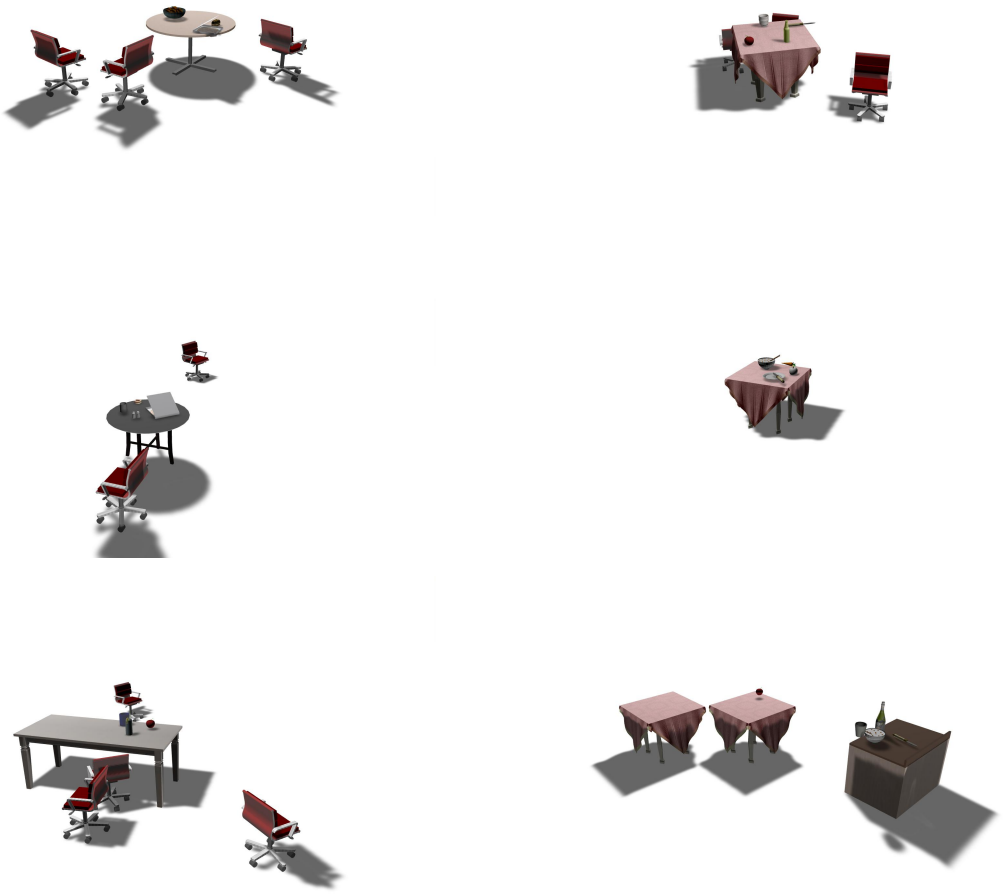


Figure A.1: Biased images from the Dining Room class.

A.2. Biased Study Room Images

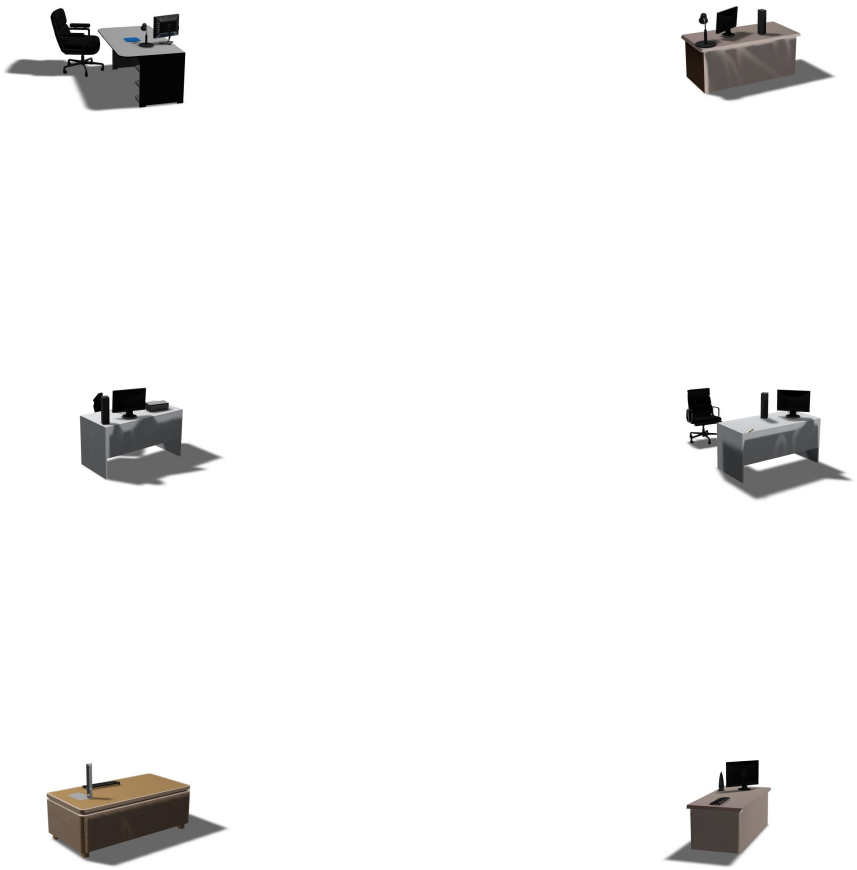


Figure A.2: Biased images from the Study Room class.

A.3. Biased Living Room Images

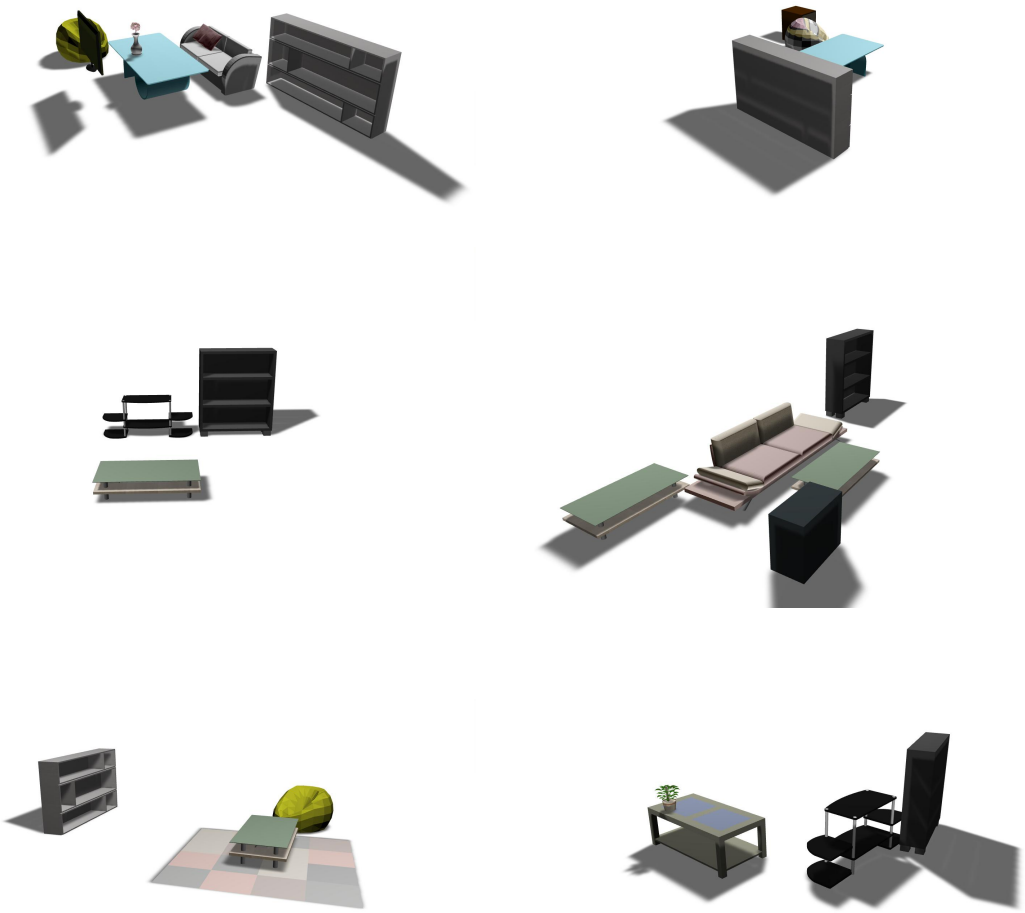


Figure A.3: Biased images from the Living Room class.

A.4. Natural Training Images

Figure A.4: Unbiased images of all classes also used for training.

A.5. Images with only Chair

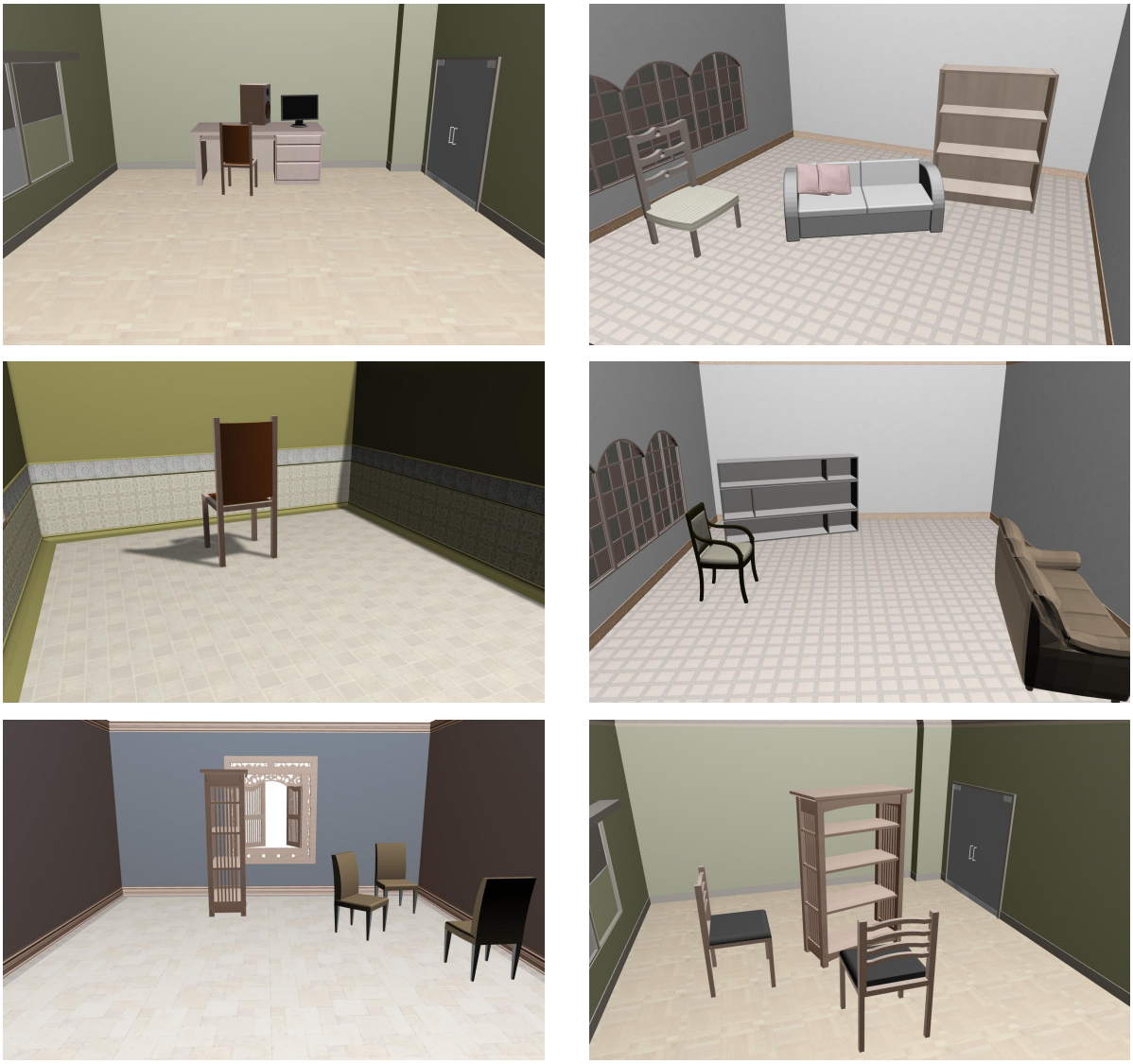


Figure A.5: Images containing a chair from the *OR* class.

A.6. Images with only Table

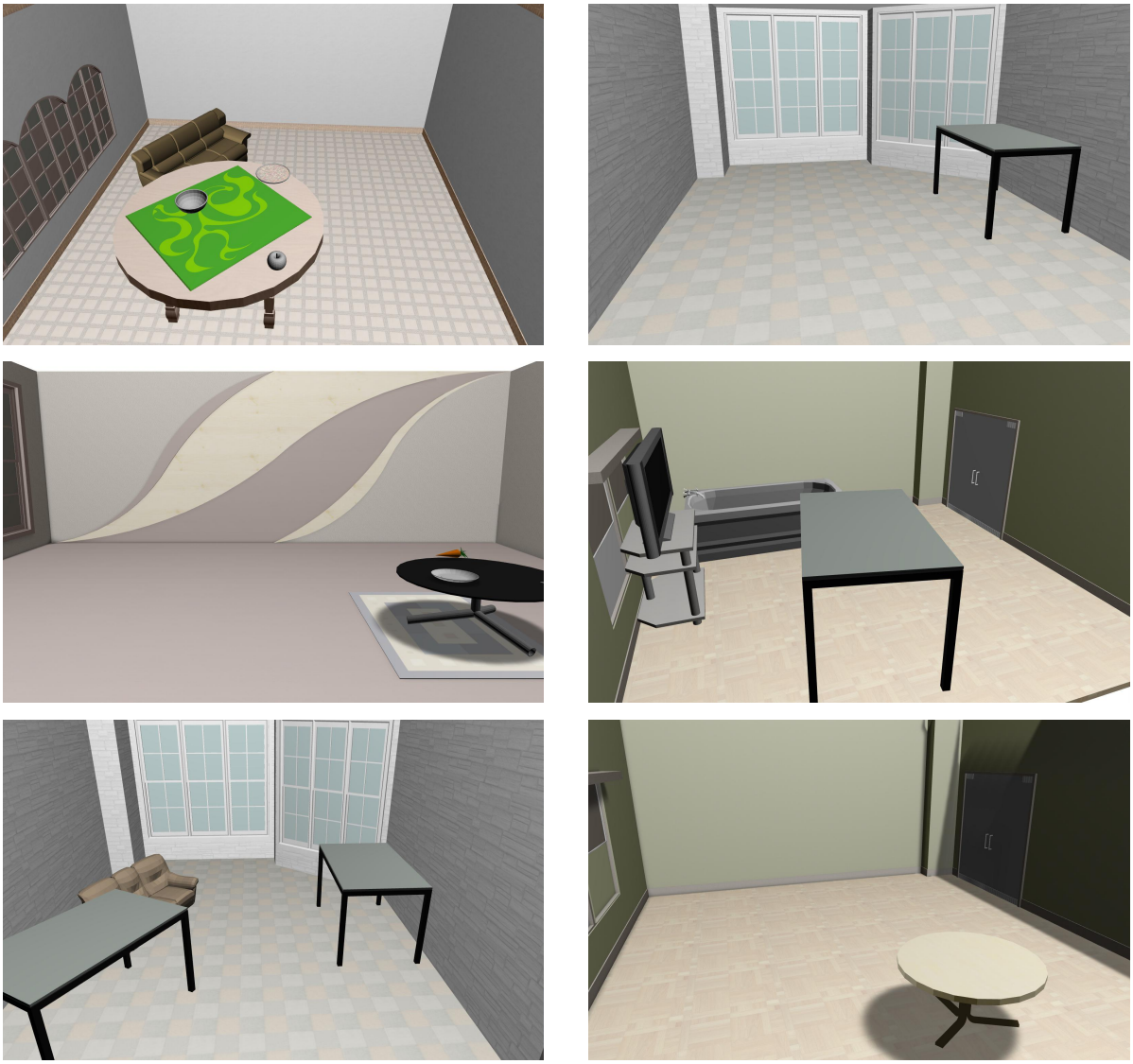


Figure A.6: Images containing a table from the *OR* class.

A.7. Images with Chair and Table

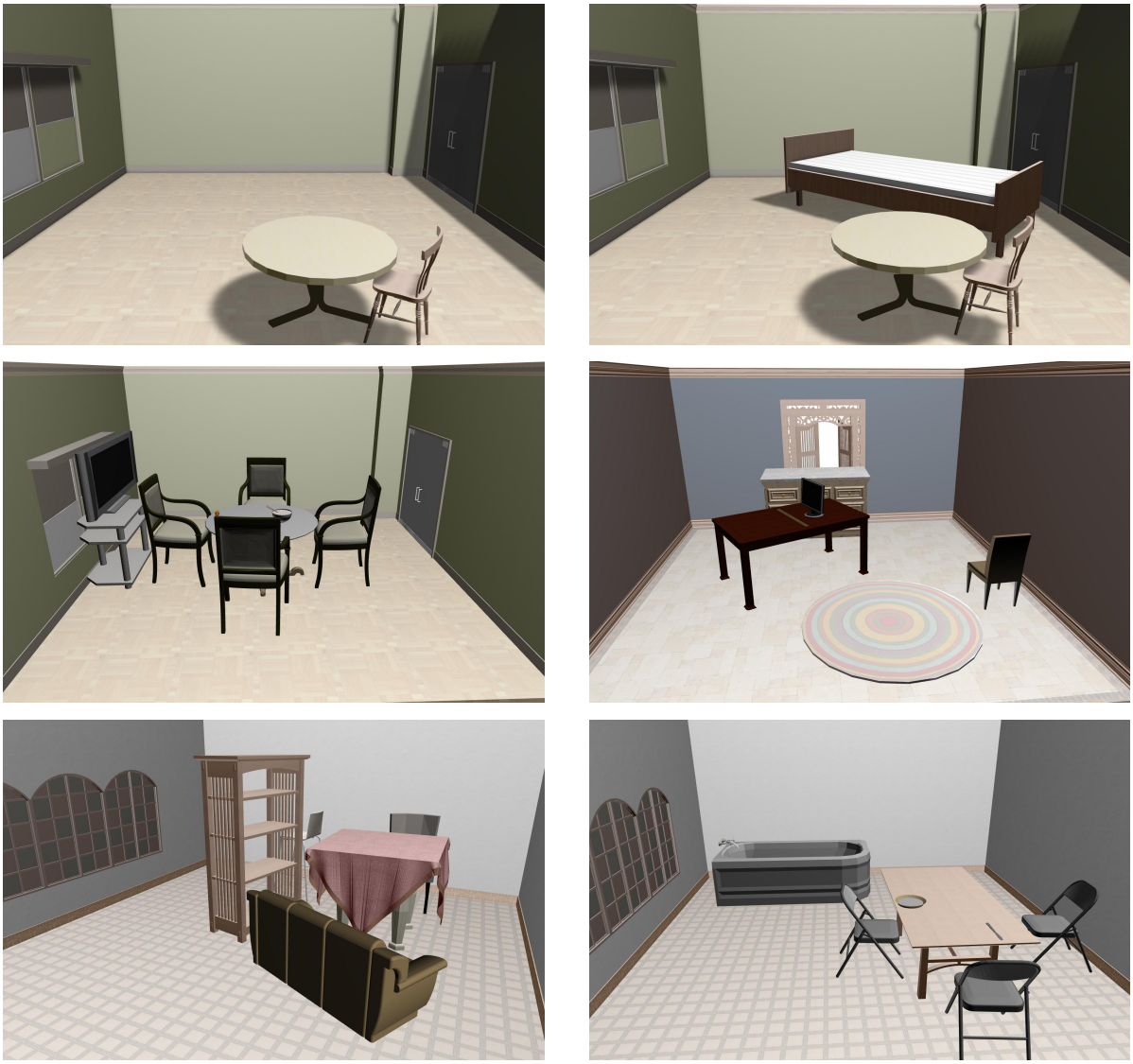


Figure A.7: Images containing a chair and table from the *OR* class.

A.8. Images without Chair and Table

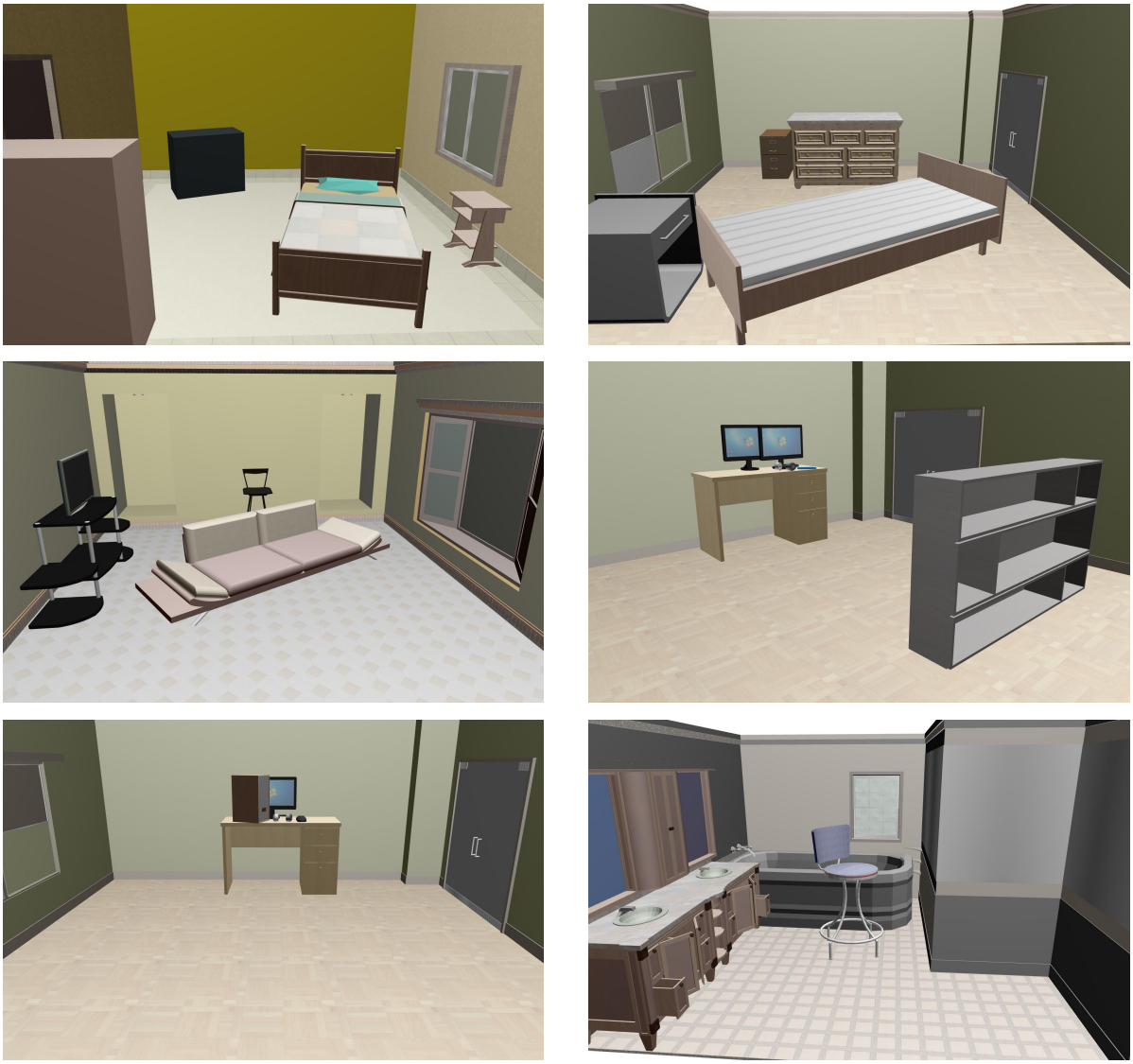


Figure A.8: Images from the *Not* class which do not contain a chair or a table.

Bibliography

[1] Agathe Balayn et al. “How can Explainability Methods be Used to Support Bug Identification in Computer Vision Models?” In: *CHI Conference on Human Factors in Computing Systems*. 2022, pp. 1–16.

[2] Agathe Balayn et al. “What do you mean? Interpreting image classification with crowdsourced concept extraction and analysis”. In: *Proceedings of the Web Conference 2021*. 2021, pp. 1937–1948.

[3] Steve Borkman et al. “Unity perception: Generate synthetic data for computer vision”. In: *arXiv preprint arXiv:2107.04259* (2021).

[4] Joao Borrego et al. “Applying domain randomization to synthetic data for object category detection”. In: *arXiv preprint arXiv:1807.09834* (2018).

[5] Angel Chang, Manolis Savva, and Christopher D Manning. “Learning spatial knowledge for text to 3D scene generation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 2028–2038.

[6] Angel Chang et al. “Matterport3D: Learning from RGB-D Data in Indoor Environments”. In: *International Conference on 3D Vision (3DV)* (2017).

[7] Angel Chang et al. “Text to 3d scene generation with rich lexical grounding”. In: *arXiv preprint arXiv:1505.06289* (2015).

[8] Angel X Chang et al. “SceneSeer: 3D scene design with natural language”. In: *arXiv preprint arXiv:1703.00050* (2017).

[9] Xiaojun Chang et al. “A Comprehensive Survey of Scene Graphs: Generation and Application”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), pp. 1–1. DOI: 10.1109/tpami.2021.3137605. URL: https://doi.org/10.1109%2Ftpami.2021.3137605.

[10] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[11] Bob Coyne and Richard Sproat. “WordsEye: An automatic text-to-scene conversion system”. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001, pp. 487–496.

[12] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[13] Maximilian Denninger et al. *BlenderProc*. 2019. DOI: 10.48550/ARXIV.1911.01911. URL: https://arxiv.org/abs/1911.01911.

[14] Finale Doshi-Velez and Been Kim. “Towards a rigorous science of interpretable machine learning”. In: *arXiv preprint arXiv:1702.08608* (2017).

[15] Tom Fawcett. “An introduction to ROC analysis”. In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[16] Matthew Fisher et al. “Example-based synthesis of 3D object arrangements”. In: *ACM Transactions on Graphics (TOG)* 31.6 (2012), pp. 1–11.

[17] Qiang Fu et al. “Adaptive synthesis of indoor scenes via activity-associated object relation graphs”. In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), pp. 1–13.

[18] Georgios Georgakis et al. *Synthesizing Training Data for Object Detection in Indoor Scenes*. 2017. DOI: 10.48550/ARXIV.1702.07836. URL: https://arxiv.org/abs/1702.07836.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[20] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014).

[21] Yash Goyal et al. “Counterfactual visual explanations”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2376–2384.

[22] Philipp Hacker and Jan-Hendrik Passoth. “Varieties of AI Explanations Under the Law. From the GDPR to the AIA, and Beyond”. In: *International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*. Springer. 2022, pp. 343–373.

[23] Stefan Hinterstoisser et al. “An annotation saved is an annotation earned: Using fully synthetic training for object detection”. In: *Proceedings of the IEEE/CVF international conference on computer vision workshops*. 2019, pp. 0–0.

[24] Sara Hooker et al. “A benchmark for interpretability methods in deep neural networks”. In: *Advances in neural information processing systems* 32 (2019).

[25] Daniel Kang, Peter Bailis, and Matei Zaharia. “Blazeit: Optimizing declarative aggregation and limit queries for neural network-based video analytics”. In: *arXiv preprint arXiv:1805.01046* (2018).

[26] Daniel Kang et al. “Noscope: optimizing neural network queries over video at scale”. In: *arXiv preprint arXiv:1703.02529* (2017).

[27] Mohammad Keshavarzi et al. *SceneGen: Generative Contextual Scene Augmentation using Scene Graph Priors*. 2020. DOI: 10.48550/ARXIV.2009.12395. URL: https://arxiv.org/abs/2009.12395.

[28] Been Kim et al. “Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)”. In: *International conference on machine learning*. PMLR. 2018, pp. 2668–2677.

[29] Ranjay Krishna et al. “Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations”. In: 2016. URL: https://arxiv.org/abs/1602.07332.

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[31] Bowen Li et al. “Controllable text-to-image generation”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[32] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. DOI: 10.48550/ARXIV.1405.0312. URL: https://arxiv.org/abs/1405.0312.

[33] Yao Lu et al. “Accelerating machine learning inference with probabilistic predicates”. In: *Proceedings of the 2018 International Conference on Management of Data*. 2018, pp. 1493–1508.

[34] Rui Ma et al. “Language-Driven Synthesis of 3D Scenes from Scene Databases”. In: *ACM Trans. Graph.* 37.6 (Dec. 2018). ISSN: 0730-0301. DOI: 10.1145/3272127.3275035. URL: https://doi.org/10.1145/3272127.3275035.

[35] Christopher D Manning et al. “The Stanford CoreNLP natural language processing toolkit”. In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014, pp. 55–60.

[36] Ojas Mehta et al. “Machine Learning in Medical Imaging–Clinical Applications and Challenges in Computer Vision”. In: *Artificial Intelligence in Medicine* (2022), pp. 79–99.

[37] Chaitanya Mitash, Kostas E Bekris, and Abdeslam Boularias. “A self-supervised learning system for object detection using physics simulation and multi-view pose estimation”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 545–551.

[38] David MW Powers. “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation”. In: *arXiv preprint arXiv:2010.16061* (2020).

[39] Aayush Prakash et al. “Structured domain randomization: Bridging the reality gap by context-aware synthetic data”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7249–7255.

[40] *QT | Cross-platform Software Development for Embedded & Desktop*. https://www.qt.io/. Accessed: 2022-07-08.

[41] Aditya Ramesh et al. “Zero-shot text-to-image generation”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8821–8831.

[42] Chitwan Saharia et al. “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”. In: *arXiv preprint arXiv:2205.11487* (2022).

[43] Axel Sauer, Katja Schwarz, and Andreas Geiger. “Stylegan-xl: Scaling stylegan to large diverse datasets”. In: *arXiv preprint arXiv:2202.00273* 1 (2022).

[44] Lee M. Seversky and Lijun Yin. “Real-Time Automatic 3D Scene Generation from Natural Language Voice and Text Descriptions”. In: *Proceedings of the 14th ACM International Conference on Multimedia*. MM ’06. Santa Barbara, CA, USA: Association for Computing Machinery, 2006, pp. 61–64. ISBN: 1595934472. DOI: 10.1145/1180639.1180660. URL: https://doi.org/10.1145/1180639.1180660.

[45] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep inside convolutional networks: Visualising image classification models and saliency maps”. In: *arXiv preprint arXiv:1312.6034* (2013).

[46] Amitojdeep Singh, Sourya Sengupta, and Vasudevan Lakshminarayanan. “Explainable deep learning models in medical image analysis”. In: *Journal of Imaging* 6.6 (2020), p. 52.

[47] Shashi Pal Singh et al. “Machine translation using deep learning: An overview”. In: *2017 International Conference on Computer, Communications and Electronics (Comptelix)*. 2017, pp. 162–167. DOI: 10.1109/COMPTELIX.2017.8003957.

[48] Daniel Smilkov et al. “Smoothgrad: removing noise by adding noise”. In: *arXiv preprint arXiv:1706.03825* (2017).

[49] Shuran Song et al. *Semantic Scene Completion from a Single Depth Image*. 2016. DOI: 10.48550/ARXIV.1611.08974. URL: https://arxiv.org/abs/1611.08974.

[50] Kai Ming Ting. “Confusion Matrix”. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 209–209. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_157. URL: https://doi.org/10.1007/978-0-387-30164-8_157.

[51] Josh Tobin et al. “Domain randomization for transferring deep neural networks from simulation to the real world”. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[52] Jonathan Tremblay et al. “Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2018.

[53] Apostolia Tsirikoglou, Gabriel Eilertsen, and Jonas Unger. “A survey of image synthesis methods for visual machine learning”. In: *Computer Graphics Forum*. Vol. 39. 6. Wiley Online Library. 2020, pp. 426–451.

[54] Unity. *Unity Technologies*. 2022.

[55] Gul Varol et al. “Learning from Synthetic Humans”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.492. URL: https://doi.org/10.1109%2Fcvpr.2017.492.

[56] Mason Woo et al. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[57] Tao Xu et al. “Attngan: Fine-grained text to image generation with attentional generative adversarial networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1316–1324.

[58] Ke Yang, Joshua R Loftus, and Julia Stoyanovich. “Causal intersectionality for fair ranking”. In: *arXiv preprint arXiv:2006.08688* (2020).

[59] Mengjiao Yang and Been Kim. “Benchmarking attribution methods with relative feature importance”. In: *arXiv preprint arXiv:1907.09701* (2019).

[60] Song-Hai Zhang et al. “A survey of 3D indoor scene synthesis”. In: *Journal of Computer Science and Technology* 34.3 (2019), pp. 594–608.