Otto-von-Guericke University Magdeburg

Faculty of Computer Science

Master's Thesis

Synth2Real : 3D-Furniture Reconstruction in Ersatz Environment (S2R:3D-FREE)

Author:

Kartik Prabhu

October 15, 2021

Advisors:

Prof. X

National Aeronautics and Space Administration

Dr. No

Department of Technical and Business Information Systems

Otto-von-Guericke University Magdeburg

Prabhu, Kartik:
Synth2Real : 3D-Furniture Reconstruction in Ersatz Environment (S2R:3D-FREE)
Master's Thesis, Otto-von-Guericke University Magdeburg, 2021.

Abstract

Besides the title, the abstract is the most important part of your thesis, as most readers will only read title and abstract. Your goal is to advertise the rest of the thesis for potential readers. For that, you briefly explain what you are focusing on in the thesis. With a misleading abstract, you will miss interested readers and maybe even attract readers with wrong assumptions about your work which will stop reading soon. The abstract should describe the general area as well as the most interesting insights of the work. It is crucial to find the right level of abstraction and length. An abstract typically consists of one paragraph that is significantly shorter than the introduction.

Abstracts typically follow the same structure. You start by describing the research area as well as the general and the specific problem you are focusing on. Then, you outline how you approach the problem in terms of concepts and evaluations. Finally, you close with the most interesting insights that you gained and why they are relevant for the research area.

Acknowledgments

If you want to, you can thank your advisors and/or anyone else who supported you during the thesis in some way.

Contents

List of Figures

List of Tables

List of Code Listings

List of Acronyms

FOL First-Order Logic
SLR Systematic Literature Review
SPL Software Product Line

List of Symbols

General

$\alpha[x\backslash y]$	Substitution of x in α with y
$\mathcal{P}(A)$	Power set of a set A
$f\vert_{A'}$	Restriction of a function $f\colon A \to B$ to a smaller domain $A' \subseteq A$

First-Order and Dynamic Logic

$A \wedge B$	Formalization of the sentence "P and Q"

1. Introduction

As Deep Learning field goes deeper and wider, the need for abundant data has become the basic requirement. The performance of computer vision based on deep learning models heavily depends on the consistency of labelled images. In practice, collecting data manually is time-consuming, expensive and needs a lot of effort. As a solution, we can train models on a synthetic dataset and deploy the models to real world problems. Unfortunately it is seen that with supervised learning, the model does not generalise when the distributions of dataset diverges and hence a model trained on synthetic dataset may not perform consistently with a real-world data.

Synthetic data can be defined as data created artificially using computer simulations or algorithms and not from real world events. These data can be created in abundance and hence are frontrunners for Deep Learning training. Synthetic approaches are also means of getting data which might be difficult to collect and annotate in real world. Further more, companies have realised that synthetic data are a necessity not just for the task which are inconvenient to annotate, but also for improving the performance of existing models. In a research article published by Gartner **?**, they claim that by 2030, synthetic datasets will overshadow a real dataset in all domains. The progression of synthetic data generation is predicted to be as shown in 1.1.

Deep Learning has made leaps and bounds in solving 2D-image tasks like classification, segmentations, object detection, etc. It has now entered the realm of Three-Dimension. An image is a 3D objects projects onto a 2D-surface. While doing so, some data from higher dimension is being lost in lower dimension. This inverse ability to reconstruct 3D objects from 2D images has a wide scale application in computer vision and robotics. Tasks like human body shapes and face reconstructions **?????**, 3D-Scene reconstructions **????**, generic object mesh reconstruction with textures are some of the advancements we had over recent years. Here comes the challenge;3D data is not easy to collect. 3D data is not just limited but also noisy and expensive to collect. The results of models depend on the consistency of the labelled data. In this thesis we discuss how game engines can contribute to generating quality labeled dataset for 3D reconstruction task.

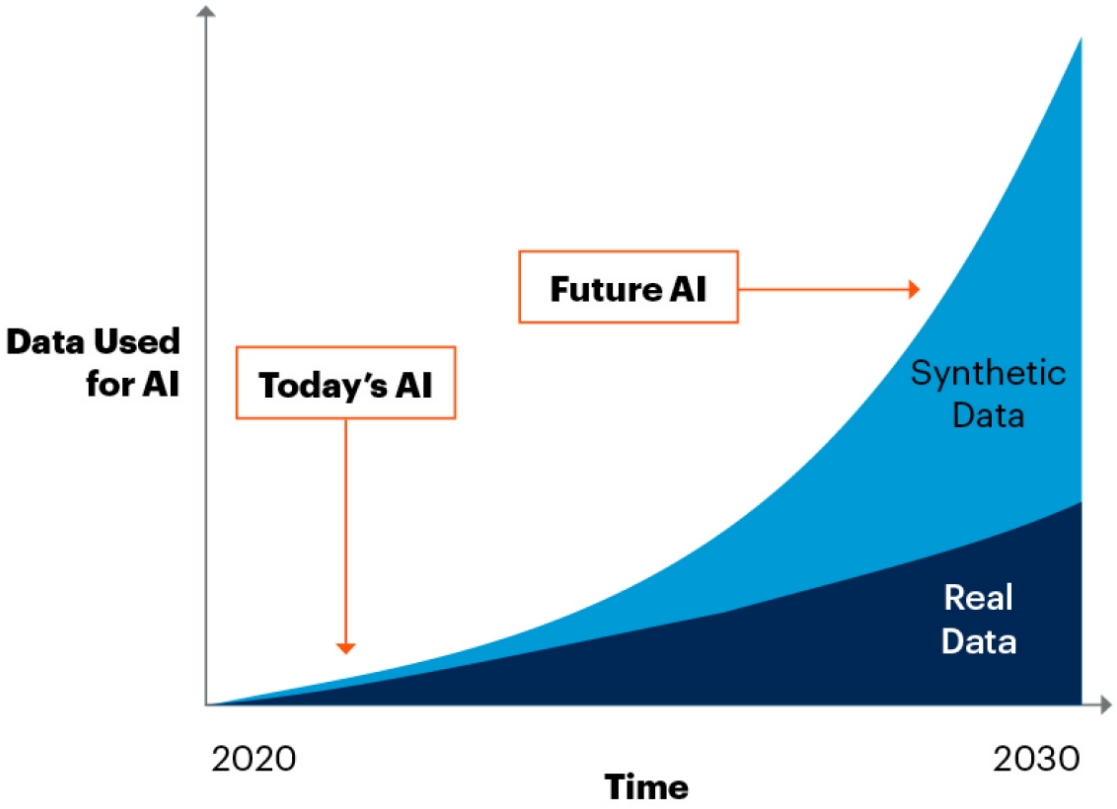


Figure 1.1: A graph showing how synthetic dataset will evolve overtime. The research graph shows that synthetic dataset will overshadow the existing real dataset **?**. Source: Research article from Gartner on "Synthetic Data Is the Future of AI" **?**

1.1 Domain Adaptation vs Domain randomisation

As mentioned earlier, the success of Convolutional Neural Networks depends highly on the training data. As it so happens, not every task will have abundant data to train with. One solution for this problem is using a pre-trained network of similar problem and using transfer learning. The pre-trained network is then trained over with the limited data available for the problem at hand. In this case the features learnt from larger dataset is fine-tuned by overwriting few layers using small annotated dataset of a different domain. Domain adaptation is field of study in machine learning where the distribution of training data differs from testing or target distribution. One shallow approach is as mentioned above; re-weighting the network with testing data to adapt to the problem domain **?**. *add more citations for domain adaptation*. "Deep domain adaptation methods leverage deep networks to learn more transferable representations by embedding domain adaptation in the pipeline of deep learning" **?** For this thesis, domain adaptation is relevant as synthetic dataset might not represent the real or target domain, which can lead to drop in performance.

Domain randomisation on the other hand, is improving data quality such that the target domain distribution is included in the training domain. This is mainly used to bridge the 'reality gap' as termed in **?**, between simulated environment and real images. This is achieved by randomising the rendering of objects in the synthetic data. The key concept here is that by rendering objects with large variance, the real-world distribution will appear to be just a variation in the dataset. Some features that can be randomised in a synthetic dataset are textures, brightness, shadows, camera settings, object pose, etc. This will be further discussed in chapter 3.

1.2 Volumetric representations of 3D shapes

The universal representation of images in computer graphics is via pixels. But unlike 2D, 3D data can be represented in various formats like voxels, mesh and point clouds. A sample of these 3 formats is shown in 1.2. Each of the representation have its own sets of advantages and disadvantages. Voxels, which is short for Volumetric Pixels is a 3D-grid of pixels of constant size. As indicated in **?**, the main advantage of voxels is that Convolution Neural Networks can be easily applied to 3-Dimensions as in 2-Dimensions. Since most of the 3D geometry is surface based, it can be wasteful and computationally expensive. Mesh is composed of vertices, edges and faces in 3D space that indicate the formation of 3D objects. This form of representation is far more compact at granular level depending on the resolution. Point clouds on the other hand are collection of 3D points with (x,y,z) coordinates on the surface of the object. The collection of points determines detailing the 3D object representation. In both these cases CNN is not directly applicable. In this thesis we will be dealing with a voxel based models as the focus is on performance of synthetic data rather than the model, or the 3d representation itself.

1.3 GameEngines

As per **?**, game engine is defined as "A series of modules and interfaces that allows a development team to focus on product gameplay content, rather than technical

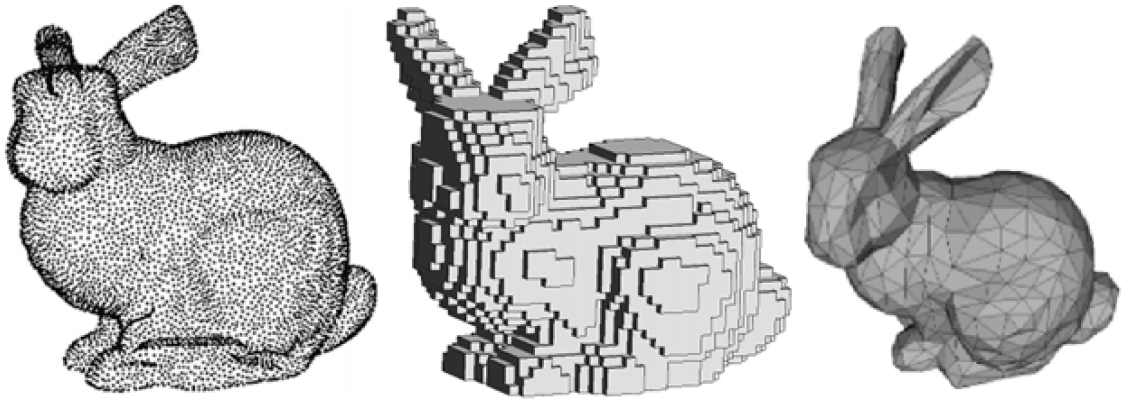


Figure 1.2: 3D representation of Standford bunny model.(left to right) Point cloud, voxel and mesh **?**

content". In layman term game engines are software used for development of games. Unreal Engine, Unity, GameMaker, Amazon Lumeryard and CryEngine are some popular game engines in modern times. Over time, the graphics in each of the game engines has improved to an extent where visually we are not able to differentiate between reality and graphics. These use object-oriented techniques which help in modularity. Also, they are GUI-oriented and hence are user-friendly and accessible even to a novice programmer. The key component in game engines which we will be focusing on is the 'Rendering engine'. 3D Rendering is process of converting 3D data into 2D images. Rendering can either be real-time or offline/pre-rendering. Game engines aim to make photorealism at highest level with least possible rendering time.

1.4 Background and motivation

3D reconstruction is of great significance to understand scene and to cross the 2D realm. But the task is not trivial, considering the input is still a Two-dimensional RGB image. It is even more challenging since obtaining new data is expensive and time-consuming. The 3D reconstruction field has datasets like ShapeNet **?**, 3D-Front **?**. But there are no corresponding real-images to check if the model will work in real world scenarios. Also searching for real-world examples or creating a set-up scene and then labelling the images manually can lead to human mistakes resulting in wrong labels. A solution to this problem is automating the task of generating synthetic data which can produce pixel-perfect annotations. This can solve human-in-the-loop problem, and while time gets saved, large quantity of data can be generated. But quantity should not the only focus while creating the data, equal importance needs to be given to quality of the data. Game engines have come long way from just supporting 2D to supporting games with photorealistic rendering. Even though game engines have grown to be successful, not many tools are based on it. As 3D models available, we can import these models into the game engine and create an ersatz environment to generate photorealistic dataset. Further, we can check if these photorealisitc synthetic data can be used for 3D reconstruction task.

1.5 Goal of this Thesis

The following research questions will be answered:

1. Are game engines a good medium to create photorealistic synthetic dataset?

   This will be evaluated with a survey to comparing images from real dataset, generate dataset and other proclaimed photorealistic datasets.

2. Can Ersatz environment from a game engine like Unity replace real data for training in 3d reconstruction task?

   Using the dataset created using Unity game engine, a single-view image to 3d reconstruction of furniture will be trained and tested on real image dataset.

3. To what extent can the performance of model Pre-trained with real dataset improved with Synthetic dataset from game engine, with size of synthetic dataset being 10*(depends on how much data is generated)* times that of real dataset?

   We use the dataset created using Unity game engine to train a single-view image to 3d reconstruction of furniture and further fine tune it with real dataset to know if the performance is enhanced. Further an ablation study on chair dataset to check the affects domain randomisation parameters. This will also answer whether augmenting images with domain randomisation with real data during training improve the performance?

4. *(optional: Can the photorealism of synthetic image further be enhanced using domain adaptation/GAN techniques?)*

1.6 Contributions of this Thesis

In answering the research questions formulated in 1.5, this thesis contributes the following:

- A unity based framework to create a synthetic dataset with domain randomisation. The tool supports both automated and manual image creation.
- A survey of photorealism to check if Game engine produce quality synthetic dataset with respect to other state of the art photorealistic indoor datasets.
- Provide a new dataset which can be used for 3d reconstruction tasks, or Synthetic to real domain adaptation tasks. This includes G-buffers like RGB, normal, depthmaps and semantic segmentation images.
- A focused synthetic chair dataset with different parameters of domain randomisation to conduct study on domain randomisation.
- A study on mixing ratio between real and synthetic images to improve performance of 3D reconstruction task.

1.7 Structure of this Thesis

The rest of this thesis is structured as follows:

- In Chapter 2, we visit some related work on synthetic data generation and state of the art 3d reconstruction networks.

- Chapter 3 deals with concepts and design choices. We will discuss in detail the choices for the dataset and model selection for 3d reconstruction.
- In Chapter 4, we discuss in detail the 3DScene tool developed using Unity game engine for creating ersatz environment. In the deeplearning side, we will discuss the pipeline design for 3D reconstruction task.
- Chapter 5 is dedicated to reviewing and discussing evaluation results obtained by comparison of synthetic and real dataset from the survey conducted and Deep learning based 3D reconstruction task.
- Finally in Chapter 6, we conclude this thesis with results of the study, highlight few limitations and discuss future improvements.

2. Related Work

In this chapter we discuss the previous contributions to indoor synthetic datasets 2.1.1, what tasks each of the dataset supports and some of the disadvantages of each dataset. Further we discuss the available tools which support creation of indoor dataset 2.1.2.

2.1 Indoor datasets

Indoor scene dataset has been in the rise with increasing interest in scene processing understanding **??????????**. Synthetic dataset is not something new in the world of machine learning. As researchers realised the disadvantages of real dataset, focus was shifted to synthetic dataset. While **?** are real-world datasets, **????** are synthetically produced. The real-world dataset are gathered from live scans. The synthetic dataset can either bet manually configured by a professional or automated by a programmer using a tool.

2.1.1 Indoor synthetic datasets

Alibaba group introduced 3D-FRONT **?** which stands for 3D Furnished Rooms with layOuts and semaNTics dataset which comprises of synthetic indoor scenes designed under the supervision of professionals. It consists of 18,968 rooms and 13,151 textured furnitures. SceneNet **?** is a large collection of photorealistic images and trajectories. This is discussed in detail in 3.2 section.

SunCG **?** was a key dataset for scene understanding. The dataset contained over 45,000 variations of scenes with realistic room layout created manually. Each scene was semantically labeled and also provided with volumetric ground truth data. The datset has also been used for task like depth estimation, semantic scene completion, SLAM, indoor navigation, etc. Unfortunately due to legal issues [1] the dataset has been made publically unavailable which has left a void in the field.

[1] https://futurism.com/tech-suing-facebook-princeton-data

Structure3D **?** is another impressive synthetic data for indoor scenes which introduced their own photorealistic renderer. The dataset comprises 21,835 rooms in 3,500 scenes and 196k 2D-images rendered with photo-realism. But the CAD models of the 3D furnitures which is used to populate the scenes is not made available to public. And hence 3D reconstruction related tasks can not be performed. It is also demonstrated that with combination of synthetic and real dataset deep learning task for room layout estimation improved performance on benchamark datasets. This dataset is more focused on room layout estimation and not 3D reconstruction, but with few modification, it can be mapped to 3d furntiture reconstuction tasks.

Openrooms **?** use Scannet **?** as their layout foundation, retrieve corresponding models from shapenet **?** and then replace the CAD model with retrieved model with proper alignment. They further add reflectance and illumination to compose photo-realistic images. As of August 2021, only the dataset has been made public and not the generation tool or the CAD models. Though the underlying concept of Openrooms is to convert existing scans into photo-realistic synthetic images, in terms of the output images we consider them our counterpart, as the framework can produce normals, depth maps, instancs segmentation and masks same as we do.

Hypermism **?** is Apple's repository for holistic indoor scene understanding. It is a collection of synthetic scenes created with the help of professional artist. **?** was the starting point for the dataset for which assets were purchased from **?**. The dataset includes images, 3D assets, semantic instance segmentations, and a disentangled image representation with diffused lighting and shading. Even though the 3d triangle meshes for each asset is available online, we have to purchase them to create custom dataset. They also admit that the costs to generate the dataset is expensive {approximately $57K **?**}.

InteriorNet **?** claims to be a photo-realistic indoor scene simulator with realistic lighting and scenes which change over time. The image dataset includes rgb, depth and semantic segmentations. Along with images, they also provide synthesized realistic trajectories at video-frame rate with various motion patterns. The simulator also supports scenes from **?** and **?** along with their own database.

Another simulated framework for visual research is House Of inteRactions (THOR) introduced in AI2-Thor **?**. This is again a Agent focused photo-realistic dataset with the key factor being actionable objects so that agents can interact with the objects or manipulate them. The underlying renderer for this framework is Unity game engine. RoboThor **?** is built upon AI2-Thor which consists of real scenes and its corresponding synthetic equivalent. This helps in the study of behavior of agents in real world when trained on synthetic data.

Habitat: A Platform for Embodied AI Research **?**, is a photorealistic 3D simulation which can be used for training virtual agents for tasks like navigation, question answering, intruction foloowing. The paper introduces Habitat-Sim which renders scenes from Matterport3d **?**, Gibson **?**, Replica **?** and some other datasets. The focus of the simulator is providing the agent with sensor data and allowing additional sensors as plugins. At the foundation level Habitat-sim uses Magnum

graphics middleware library [2] which supports cross-platform on various hardware configuration.

In figure 2.1, we can see samples from each of the datasets which have claimed to be photorealistic. The images from these dataset are used in a research survey to determine the perception of humans about photorealism.

A table to explain diffs between various methods(only if 3D-FREE has more advantage)

2.1.2 Tools to create synthetic datasets

BlenderProc: Reducing the Reality Gap with Photorealistic Rendering **?** is a python based pipeline to create synthetic dataset. As the name suggests, the key underlying framework is Blender **?**, which is a 3D modelling and rendering package. Like most of the synthetic data generation tools it provides with rgb, depth maps, normals, semantic segmentation. Figure 2.2 is a collection of sample images generated using BlenderPoc. As far as we searched for synthetic data generation tools, Blenderproc supports maximum number of existing dataset. Ikea **?**, Pix3d **?**, Shapenet **?**,3DFront **?** Replica **?**, SunCG **?** are some of the popular dataset it supports. It also has some combinations of these datasets like ShapeNet with SunCG or SceneNet. Even though it supports rendering of Pix3D dataset, the model is rendered without any background. One extreme advantage of this toolkit, is that it is opensource and hence a wider open community to contribute.

NVIDIA developed a Deep learning Dataset Synthesizer (NDDS) **?** in the form of a plugin for Unreal Engine 4(UE4). The plugin can synthesize images,per-pixel segmentation, depth, object 3D pose, 2D/3D bounding box, keypoints, and custom stencils. It even supports domain randomisation of objects, lighting, camera position, poses and textures. Leveraging the asynchronous-multithreaded frames, data was generated at high rates(50–100 Hz) for Falling Things (FAT) **?** dataset.

"SynthDet: An end-to-end object detection pipeline using synthetic data" **?**, is an open source project supported by Unity technologies using Unity Engine. This is an ML pipeline which uses 63 categories of common objects(example: cereal box, cady, cartons, etc) as synthetic objects to generate 2D bounding boxes. This project was highly influenced by **?**, where in they use synthetic 3D models with randomised background to generate synthetic dataset for object detection task. With domain randomisation the author of **?** proved that no real data or mixed training is required for Deep Learning model to perform significantly.

UnrealCV **?** is open source project built upon Unreal Engine 4 (UE4) **?**). It supports some pre-built indoor architectures available on the asset store, which are unfortunately paid versions. Along with this limitation, the plugin only creates depthmaps, normals and segmentation masks, with no mapping to 3D models. Another tool based on Unreal engine is UnrealROX+**?**. This pluggin design is based on UnrealCV and provides similar data.

[2] https://magnum.graphics/

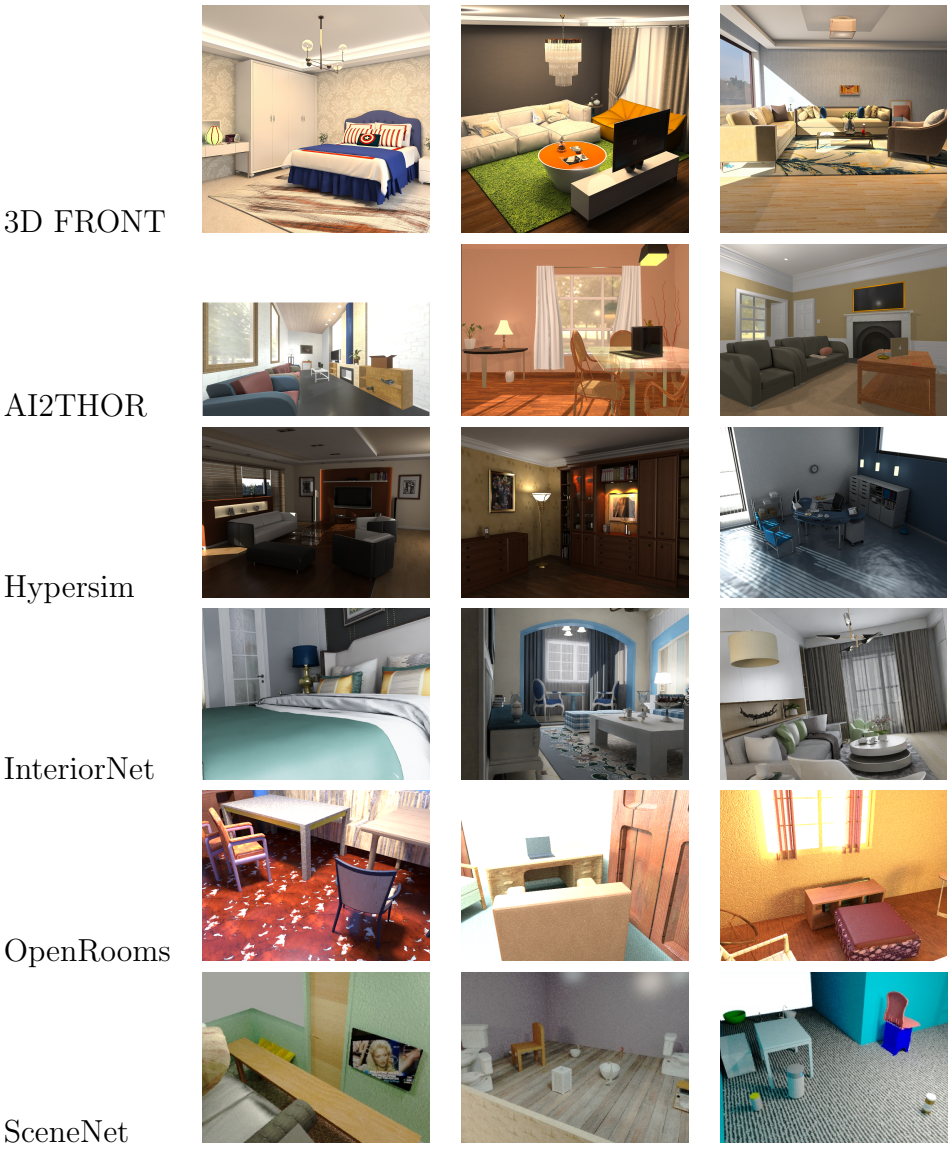


Figure 2.1: A collection of photorealistic synthetic datasets. The first column of each row indicates the dataset name followed by randomly selected images from the same dataset.

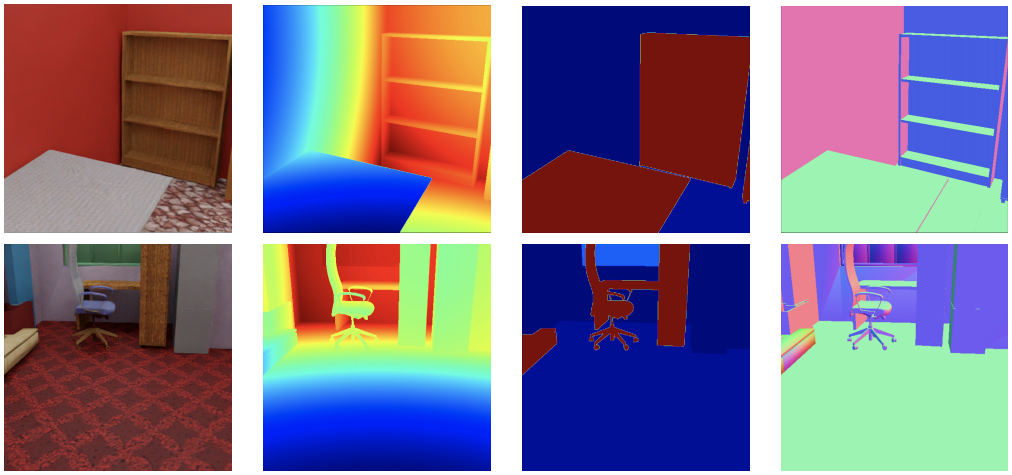


Figure 2.2: Sample images created from BlenderProc using SceneNet dataset.(Left to right) RGB images, Depth Maps, Instance Segmenatations and Normals. Each row is an independent sample.

2.2 State of the art for 3d-reconstruction

3D reconstruction task is on the rise with improvement in technologies like Augmented and virtual reality. Scene understanding becomes important for these technologies to succeed. As discussed in 1.2, 3D models can be represented as voxels,meshes or point clouds. Each of the representations have been explored for reconstruction and this section explores some approaches.

The task can either be applied to multi-view, or a single view image. **???????**, are architectures based on multiple input images of same images from different views to learn the underlying 3D structure. **??** are architectures based on Recurrent Neural Networks (RNN) and have the disadvantage of memory, long-term dependencies and long inference time. **???** explore different ways of pooling(attentional, max and average pooling respectively) to aggregate the feature vectors from different views. The pooling method tend to capture only partial information with loss in detailing.

For single-view reconstruction we have architectures like **?????**. **?** predicts the 3D structure by estimating 2.5D properties like surface normals, segmentation and depths. **?** is extension of MarrNet which proposes a network to preserve the naturalness of the surface. Inspired by generative models **??**, many 3D reconstruction architectures were published **????**. Both **??** also have the capability to predict 3D structure from a single view image.

To overcome the memory issues caused by 3D voxels many new architectures have also been proposed **???????**. These are approaches that are based out of oct-tree, meshes, occupancy of voxel, etc. Occupancy networks are a implicit way of representing 3D surface as decision boundary using network as a deep learning classifier. For mesh-based reconstruction, a template mesh such as an ellipsoid is deformed to get the structure of 3D model. For the Oct-tree based reconstruction, as the name itself suggest, instead of 3D voxels the models are discretized in Oct-Tree representation. This further makes it memory and compute efficient providing higher resolution than 3D voxel representation.

Supervision is another factor to consider for 3D reconstruction tasks. There are method based on 3D-Supervision or 2D-Supervision. **??????** are few of the 3D-supervision based architecture. For all these architectures it is necessary to either have the 3D voxel grid or 3D mesh for supervised training. For models described in **??**, the 3D reconstruction only needs 2D supervision like silhouette, One might argue that for 3D supervision we need large number of 3D models designed and created by professional artists. With the advent of Virtual and Augmented realities, 3D models are which are available to public are increasing exponentially, which might resolve the issue.

2.3 Mitigating domain shift

In **?**, the author tries to find how much of real data is required with synthetic dataset to increase the performance of object detection task. A study was conducted by training various object detection models on mixed data containing different ratios of real and synthetic dataset. The four ratios considered for mixed training were, 0%,90%,95%,97.5% of synthtic data to real data. Another study was conducted by fine tuning a model pre-trained on a synthetic dataset with different quantity of real dataset. They observed that fine-tuning with limited real dataset is better than mixed training with synthetic dataset.

The idea of domain randomisation is that among the set of training images from synthetic dataset, real images appears to be one of the variations. Thus, testing with real world images should give comparable performance. **?** used this principle for small objects detection task used for robotics. This paper tries different parameters in domain randomisation like lights, camera viewpoints, textures and further combines domain randomisation with domain adaptation.

? explores domain randomisation on real-world object detection. The author also proposes a new approach of domain randomisation where in random 3d objects called "flying distractors" are randomly placed in the scene, so that the model learns to ignore objects which are not of interest. A focused ablation study with individual domain randomisation parameter was also conducted to understand its contribution. The parameters included lights, textures, data augmentation and flying distractor as mentioned above.

? introduces Structured domain randomisation, where in the structure and context is preserved for the problem at hand while creating synthetic images. Example: A road is randomly placed with parameters like curvature, lighting and camera position. With this as context, the pedestrian walk, road lanes, etc are generated. After these 2 steps, the objects such as cars, cyclists, houses, buildings are placed in the scene. As ablation study, the author changes domain randomisation parameters and checks the performance on car detection task.

2.4 GAN based style transfer

3. Concept

In this section we discuss about the dataset and design choice,and the rationale behind reducing the domain gap.

3.1 Pix3D: A large-scale benchmark

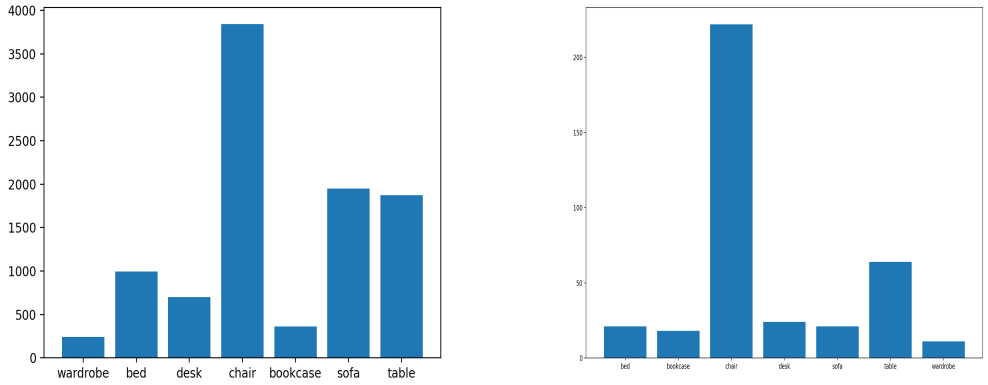
In **?**, a large-scale benchmark for 2D-3D alignment was introduced. The raw images from web search engines were collected and labelled keypoints were used to align the 2D images with the corresponding 3D shapes. The 3D models are extension from IKEA dataset **?**, which is a collection of high-quality IKEA furniture. The dataset also provides with binary mask and the keypoints for the object under observation. For adding more images to the IKEA dataset, the authors of **?** conducted manual web search on Google, Bing, and Baidu, using Ikea model name as keywords. to get around 104,220 images which were further filtered by removing irrelevant images with the help of Amazon Mechanical Turk (AMT) workers. After this manual experiment the total images in Pix3D, only 14,600 images were selected for the 219 Ikea models. For our experiment on synthetic to real dataset, we chose to select only furniture classes from Pix3D, leaving out "misc and tools" classes, which were significantly less to begin with.

3.1.1 Disadvantages of Pix3D

The distribution of models in pix3d is as shown in 3.1. As we can see, the dataset distribution is uneven across classes and more than 50% of classes have less than 1000 images.

Though Pix3D set a benchmark for 2D-3D alignment, here are few disadvantages of using this real dataset.

1. For Deep Learning approaches, we need large-scale data and 14,600 might not be sufficient.
2. The orientation of object is not randomised.
3. The dataset does not provide 2.5D information (i.e. depth and normal) which can be crucial for 3D learning.



(a) **(b)**

Figure 3.1: Distribution of pix3D **?** images(left), unique models(right)

3.1.2 Why Pix3D?

Pix3D is a collection of indoor scenes with complex background, varying light conditions with shadows, reflective surfaces and even varying occlusion levels. Each image comprises a collection of objects in the scene, but only one object from the catagory is annotated. It is a perfect example of having limited real world data. Since 3D models are available for each furniture,a synthetic data can be generated in abundance from those models. Samples from Pix2D are as shown in figure 3.2.

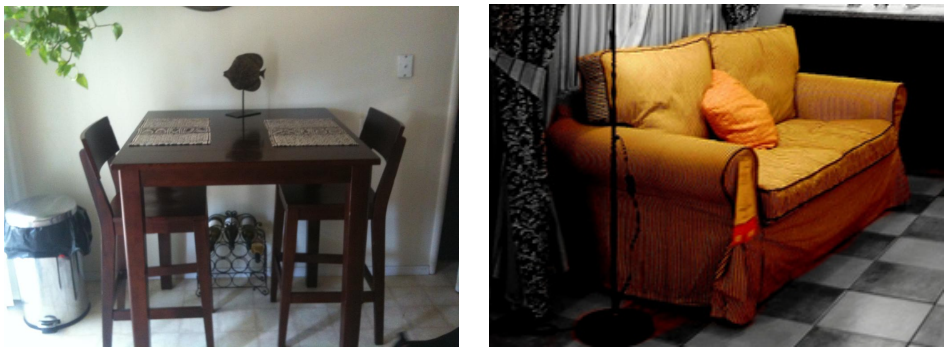
3.2 Role of SceneNet

SceneNet **?** is a large collection photorealistic indoor scene trajectories. The dataset provides with images and videos of indoor scene which can be used for tasks like SLAM, semantic and instance segmentation, object detection and further enhanced for other vision problems like optical flow depth and pose estimation **?**). They use ShapeNet **?** models to occupy 57 indoor scenes which gives the scene unlimited configurations. Unfortunately there is no mapping from scene to 3D model for tasks like 3D reconstructions. In our approach, we utilize the scene provided by SceneNet as layout for our indoor scenes. This includes initial shapenet models with furniture placement. We then replace the class under observation with a corresponding model from Pix3D.

3.3 Unity based pipeline

Unity game engine is used for creating the synthetic dataset. The reasons for selecting Unity as our platform for the application are:

1. Cross-platform game engine and hence can be used on an Operating system.
2. The basic version is available for free.
3. Provides all the necessary tools to create an ersatz environment with a well maintained documentation.



(a) (b)

(c) (d)

Figure 3.2: Sample RGB images from pix3D

4. An active developer community.

There is no official comparison between Unreal Engine and Unity engine to have a deciding factor. The selection of unity engine was purely individual choice and ease of use. But both these game engines have ability to create realistic looking scenes which can be used for synthetic dataset generation. With Unity game engine, the available scenes from scenenet and 3D models from pix3d can be imported to form an ersatz environment. Further domain randomisation can be applied to create a dataset of photorealistic images and 2.5D data like normal, depthmaps, masks.

3.3.1 Domain Randomisation with Unity Engine

In this section we discuss what kind of domain randomisation is applied for the dataset creation.

3.3.1.1 Scenes from scenenet

As discussed in section 3.2, SceneNet provides wide variety of indoor scenes which includes bedrooms, bathroom, living room, kitchen and office. For our case, we did not consider the bathroom, as the categories under observations are furnitures which we rarely see in the bathroom. For the rest of the scenes we have a pre-determined setup given by scenenet. We use 25 scenes in total as basic rooms for our target object to be place. And with further randomisation we feel this should be sufficient to check if unity can produce a useful dataset.

3.3.1.2 Camera Viewpoints

The position and orientation of camera gives us the most randomisation in this setup. The minimum and maximum distance can be chosen by the user for the camera to be places from the target object. A random point can be selected in this range as the position for the camera. As for the orientation of the camera, we can make the camera to always look at the target object, irrespective of the position. With this, we achieved different background for target object, and the target object itself appears to be in different position and orientation.

3.3.1.3 Lighting and shadows

Lighting plays an important role in photorealism. If the lighting is not setup correctly the synthetic dataset will never seem to be photorealistic. Unity offers wide variety of lighting like global light which acts like the sunlight, and various indoor lighting systems. Ideally we should make the luminous objects like lamps, chandelier, bulbs etc be the source of light for indoor scenes, but we observed that the room does not lit up uniformly which makes in less photorealistic. Hence we use some pre-determined lighting settings which will be further discussed in implementation section 4.1.

3.3.1.4 Randomised textures

SceneNet **?** also provides with textures for different categories in the scene. We then further increase the texture database by adding few more textures from ambientCG.com, which provide license for free. We randomly allocate textures to each object making sure that each category has same texture to make the scene more uniform.

3.3.1.5 Replacing target objects

To further randomise the scene, the category of target objects in the scene are replaced by the object under observation. When there are more than one object of same category we randomise the selection of which object is to be replaced which further randomises the captured data. The scale of the target object is made to match the scale of the category object in the scene in the least dimension. For example if the length of the length of the category object in the scene is the least amonth length,width and height, then the target object is scaled to match this length. This makes the target object blend in with the scene.

3.4 S2R:3D-FREE, a Pix3D based sythetic dataset

S2R:3D-FREE dataset, which stands for Synth2Real: 3-Dimensional Furniture Reconstruction from Ersatz Environment, is a combination of SceneNet and Pix3D dataset. We utilize the availibility of 3D models of rooms and furnitures from these 2 dataset to create an ersatz environment using Unity as our framework. We randomise the indoor scenes from SceneNet **?** along with textures provided by them with some additional complex textures. The other option would have been to use the shapenet as the target model, but in this case we would not be having a real dataset to compare to. Ideally a model trained on shapenet should also perform well with a real dataset like pix3d, but we decided to have a synthetic dataset based on pix3d itself for better comparison.

3.5 3D reconstruction

Now that we discussed on how the dataset will be created, we move to the deep learning aspect of this thesis. 3D reconstruction is the task to convert a 2-Dimensional RGB image into a known 3D representation of the model under observation. For voxel-based approach, the 3D model space is discritized to obtain 3D voxel grid X. Let $I = \{I_k \mid k = 1, 2, \dots n\}$ with n≥1 be a set of single view RGB images with atleast one object X. 3D reconstruction can be defined as a task to learn a function $f(\theta)$ such that the objective function $L(I) = d(f(\theta), X)$ is minimised. θ is the parameters of predictor function $f.$, (d) is a distance measure between real shape X and shape predicted by $f(\theta)$,

In this thesis we consider an architecture which consists of encoder and decoder. As discussed in **?**, an encoder maps a 2D image into a latent space by converting pixel grid into feature vectors. If we formulate the problem, feature vector $v = Enc(I) \in X$, where Enc is the encoder function and I is the input image and X is the latent space. An encoder is a good mapping function if for given images of same category $I = \{I_k \mid k = 1, 2, \dots c\}$, the feature vectors $v = \{v_k \mid k = 1, 2, \dots c\}$ are close together in given latent space X irrespective of the extrinsic parameters like textures,background, camera positions,etc.

Decoder is an expansive network, which converts a feature vector from the latent space into a 3D voxel grid. $V = \{g(x_k) \forall k \in X\}$, for any vector in latent space X the decoder should be able to create a 3D voxel model from V. If the images are encoded perfectly, a small change in feature vector will give rise to a small variation in the 3D voxel.

3.6 3D reconstruction pipeline

We create a pipeline for processing the above mentioned 3D reconstruction task. The backbone of the pipeline is the base model and the dataset being used to train. In this section we discuss the model used as a base and the rationale behind its selection.

3.6.1 Pix2Vox and Pix2Vox++

The architecture of pix2vox **?** as in 3.3. The network consists of 4 modules: Encoder, Decoder, Merger and Refiner. Merger plays significant part when it comes to multi-view reconstruction of 3d objects. As we focus on single-view 3d reconstruction the merger module will not influence the output significantly. As Encoder, pix2vox has utilizes VGG16 network **?** pre-trained on ImageNet **?**. Decoder is an expansive network which converts 2D embeddings to 3D voxel grid. Refiner is an auto-encoder which takes the 3D output from the merger and produces more refined final output. A second paper based on pix2vox was titled Pix2Vox++ **?**. The extended work replaces VGG encoder **?** with ResNet **?**.

3.6.1.1 Why Pix2Vox?

Pix2Vox has been used as baseline by most of the research oriented to 3D reconstruction. This network is one of the few networks to be tested on pix3d dataset.

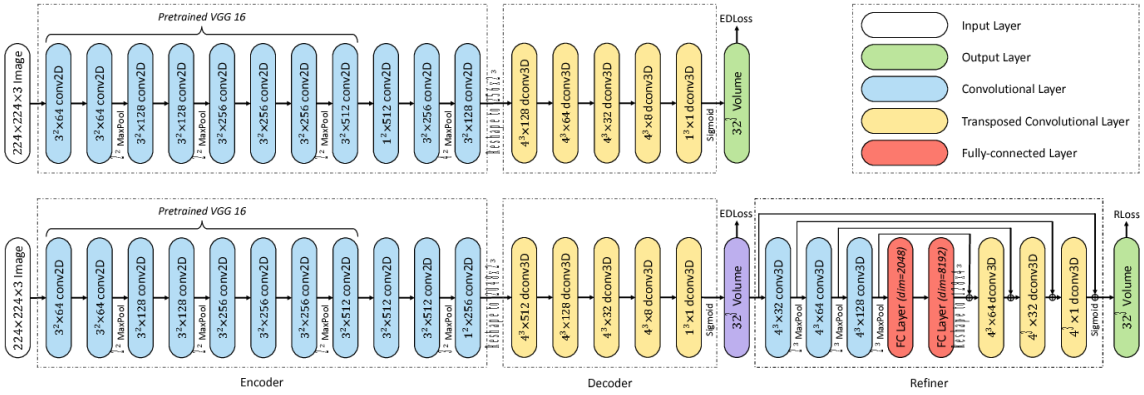


Figure 3.3: Network architecture for pix2vox **?**

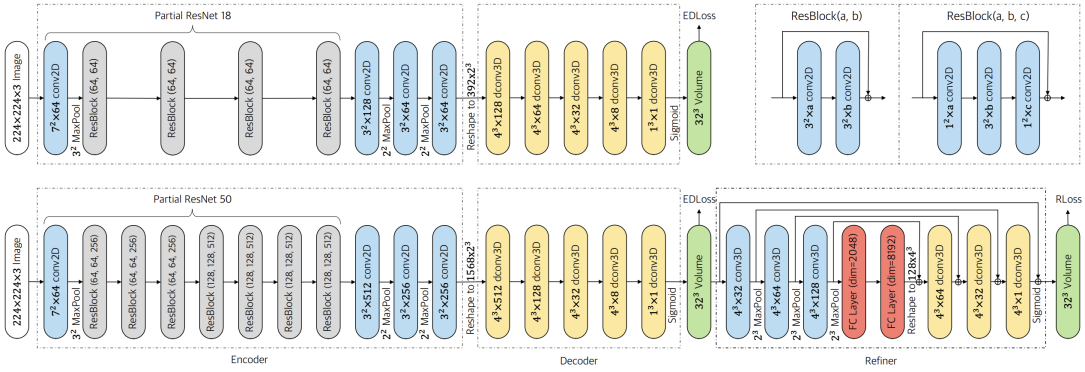


Figure 3.4: Network architecture for pix2vox++ **?**

According to the survey conducted by **?**, performance of pix2vox **?** has been noted to be significantly higher when compared to previous work *cite other papers* as shown in 3.5. While this comparison was made on 3d reconstruction of shapenet dataset, since pix3d was not available at the time when previous work were published. From our survey, only CoReNet **?** had a slight gain in performance compared to pix2vox. When trained on shapenet and tested with pix3d, CoReNet gave a result of 29.7% IoU while Pix2Vox gave a result of 28.8% IoU and Pix2Vox++ a result of 29.2% IoU. Since the difference in the performance was not statistically significant we decided to stick with the baseline model itself. Another reason for selecting pix2vox model is that the backbone of the architecture is pre-trained with ImageNet and hence the embeddings generated from this encoder can help in visualising the domain space of both Pix3D (real images) and S2R:3DFREE(synthetic images). As mentioned above, for pix2vox++ the Resnet is the backbone encoder which has 25% lesser parameters and 5% lesser inference time compared to VGG. In addition, the author even proved that Pix2vox++ performs 1.5% better than pix2vox. The architecture of pix2vox++ is as shown in figure 3.4. Furthermore, the focus of this thesis is not to check which is the best model to reconstruct the model, but to check if game engines can produce photorealistic images usable for 3d reconstruction. Hence the selection of the model was not of utmost importance. But since the 2 architecture are relatable, it would be interesting to compare the results for 3D Reconstuction task.

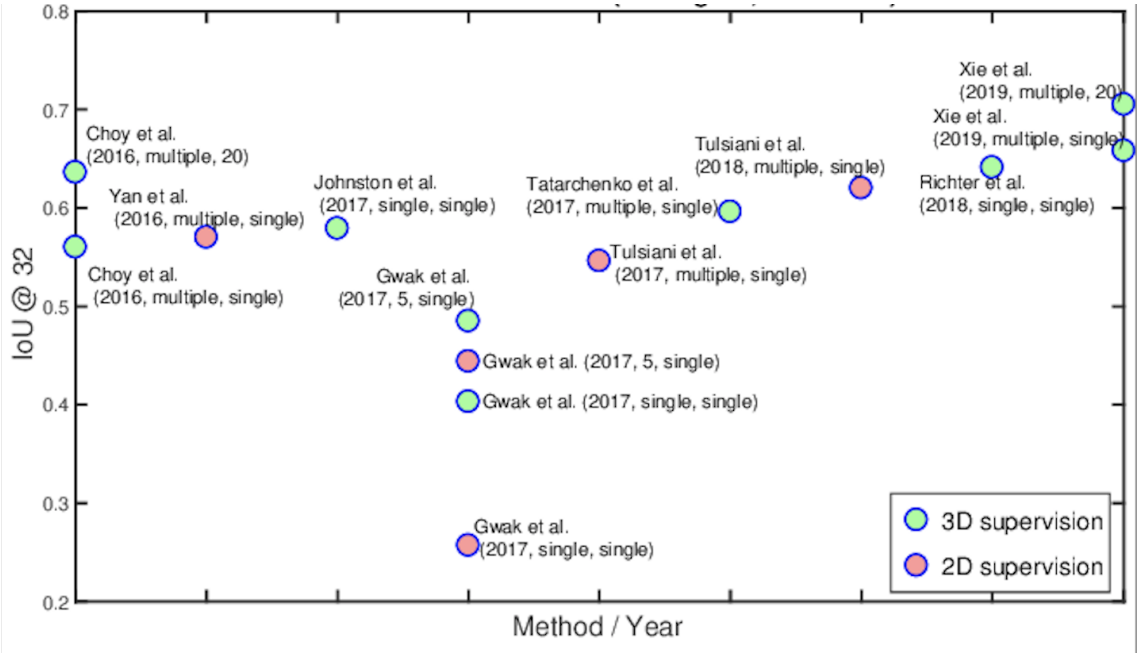


Figure 3.5: A survey conducted by **?**, proves that Pix2Vox is considerably a good 3D reconstruction model. *replace this figure with better one*

3.7 Domain adaptation with Gan *if we decide to go with this approach*

3.8 Training

*r*e-write **?** proved mixed training with synthetic and real dataset improves the performance. They used 40% mixing ratio

? used mixing ratios for classification task for each of the minibatches and observed 5-13% improvement in performance.

4. Implementation

4.1 3D-Scene framework

3D-Scene is a Unity based application used in editor mode written in CSharp programming language. The First step of the application is to import existing .obj files for 3D models, along with 3D models of rooms. These rooms and furniture models can be randomly textured and placed. In the final step, the main camera is used for taking snapshots of the scene. Along with RGB images, depth maps, normals, semantic segmentation are also saved. The pipeline is as shown in figure 4.1. For automation based pipeline, the process loops through each step, while in manaual pipeline the user decides which block needs to be executed.

4.1.1 Modes of operations

Creation of dataset can either be automated or by manual intervention. Automated images may not give us the perfect images which we expect. There can be some bad lighting, unexpected intersections with other objects, unforseen camera angles, etc. To facilitate ease of use for the user, the application has 3 key pipelines.

1. Single Room pipeline

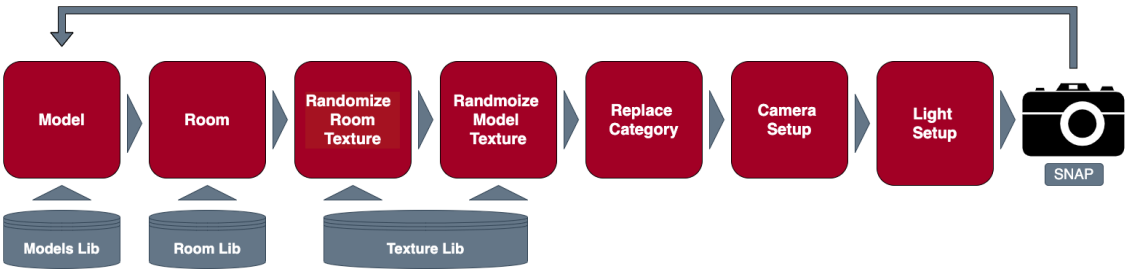


Figure 4.1: Automated pipeline for image generatiion with different blocks and external libraries.

2. Manual Room pipeline
3. Multi Objects pipeline

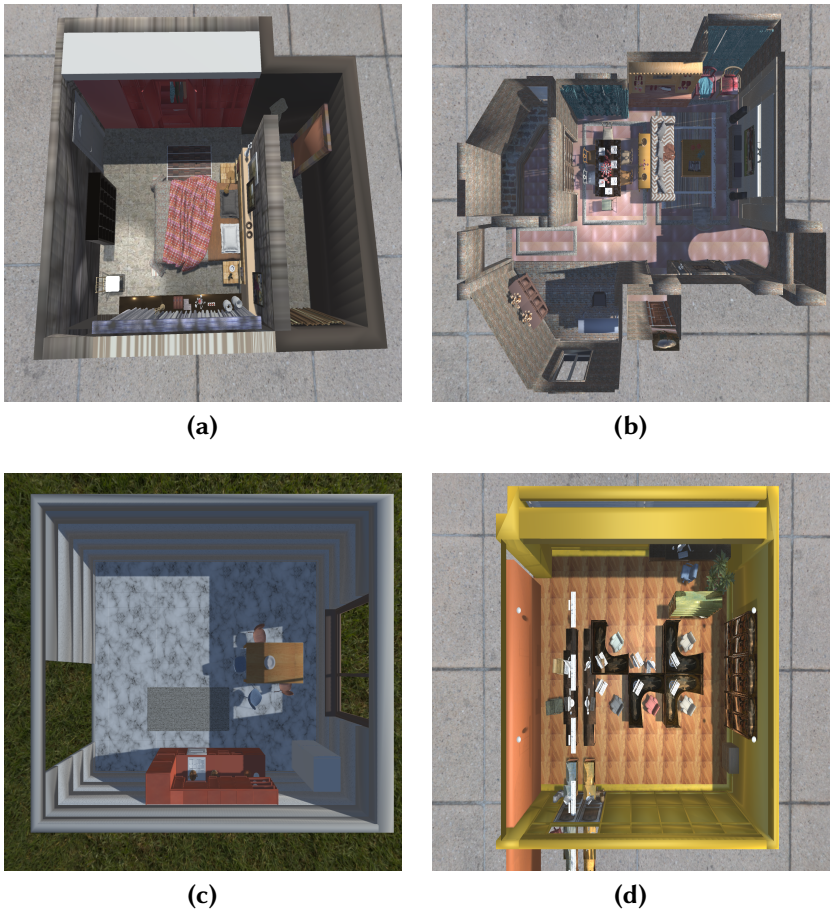
Single Room pipeline is used to create the dataset with objects in the center of an empty room. A room path can either be provided, else a default room is imported. Manual Room pipeline randomises a furnished room, and then replaces the category under observation. The selection of object to be replaced is also random. In this mode, the user has the control over taking the snap of the scene. The user can randomise model an room textures, randomise camera position or manually set it and randomise the lighting conditions. Once the user is satisfied with the view of the scene, images can be saved with a click of the Snap button. In Multi Objects pipeline, all the process mentioned in the Manual Room pipeline is automated. The user won't have control over any of the process, while the program snaps images at random. Another version of pipeline which is similar to Single Room pipeline is the Multi-threaded Single Room pipeline. As the name suggests, multiple rooms are created based on the number of categories and for each room the Single Room pipeline is applied in parallel. This is an attempt to let the tool perform faster with multi-threading. The multi-process uses Coroutines which is still a sequential operation and hence this pipeline needs some work. To our estimate even the Multi-threaded pipeline runs 1.5 times faster than Single room pipeline.

4.1.2 Scenes from SceneNet

For the rooms in our ersatz environments we utilize scenes provided by SceneNet **?**. SceneNet has made 25 rooms available for public. We only use scenes of type Bedroom(11), Kitchen(1), Living room(6) and office(7). Figure 4.2 shows different types of scenes utilised to generate synthetic dataset. These scenes are further modified by adding few more objects of categories present in Pix3D, so that we get more variations in the dataset. The distribution of furnitures matching the category of Pix3D in the scenes are as shown in figrue 4.3. Each scene is replaced for every snap we take for the dataset we create at random.

4.1.3 Camera ViewPoints

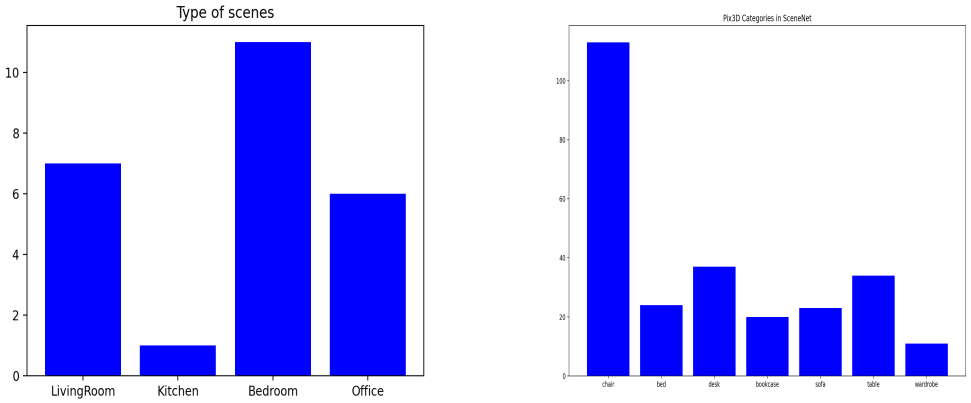
The distance of camera from the target object is a configurable entity. The user can set the minimum and maximum distance to the target object for the camera to be placed. The program will then randomly select a point within this range. Similarly the height of the camera can be configured by the user. The camera is programed to always look towards the target object and will be changed if the object is not in the frame. This is acheived by passing a ray and determining if the center of the target object is visible. To make the frame realistic, we avoid unrelatable views by applying a constraint on the camera poisiton and make sure that the camera is in front of the target object within an angle of 60 degrees. Figure 4.4 shows samples of different camera view points on a constant object and scene.



(a) **(b)**

(c) **(d)**

Figure 4.2: Sample scene layouts from SceneNet. Types: (a)Bedroom, (b)LivingRoom, (c)Kitchen and (d)Office



(a) **(b)**

Figure 4.3: (a)Distribution of types ofs scenes, (b) Distribution of objects matching the categories of pix3d in scenes

Figure 4.4: Sample images with different camera viewpoints of same object with a constant scene.

4.1.4 Lightings and Shadows

We consider lighting to be a key component of the photorealism. The shadows formed with different lighting condition enhance the photorealism of the images. For this purpose we use 3 types of lights offered in Unity.

- Point light
- Spot light
- Sunlight

Point lights act as indoor lights for the room the range of which can be varied. We pre-define 6 light patterns for a room with cuboid shape. Knowing the bounds of the room we decide how to place the lights by randomly select 1 to 6 lights with corresponding light patterns on the ceiling. For the spotlight, only a single light is placed a meter above the target object. This light gives a variation which focuses only on the target object. Both point and spot-light have color variation along with varing brightness. Sunlight is the default light settings in Unity. This is specifically effective when the room has windows forming more realistic shadows. In this case, we modulate the brightness and the angle at which the rays are emitted, which simulates different times of the day. Additional to these different types of light, we give ceilings the property of light with limited brightness as we observed that the entire room does not lit up for single light source. This is similar to what BlenderProc **?** does for all scenarios. Samples of different lighting condition and shadows with varying color is displayed in figure 4.5.

Figure 4.5: Sample images with different lighting and shadows conditions.First row is samples for light with different intensity and direction. Second row is differnt color for light.

Another randomised entity is the skybox. Skybox acts as the outdoor scene for a given room. For each of the pipeline, skybox changes for every snap taken. This simulates different outdoor places and weather scenarios, thus providing more randomisation for rooms have open doors and windows. Samples for different skyboxes are as shown in figure 4.8.

4.1.5 Randomised Texture

The objects in the scene are renamed to standard objects seen in day-to-day life. The textures are grouped together to these names as folder names. A total of 982 texture images, some of which are just JPG or PNG images, while there are few texutres from cctextures.com with more details like normals, displacement and roughness. This makes the textures more realistic. Distribution of textures used for randomisation of scenes is as shown in figure 4.7, with categories of Pix3d having higher number of images. Each scene is randomised for every snap taken of the target object. Samples of texture randomisation of background in the scene with target object in the focus is as shown in figure 4.6. Randomising outdoor environment is important in cases where we have open doors and windows. Unity provides a wrapper for scenes called skyboxes. This is something of a globe around the room under observation. This gives us varying outdoor environment for the scene with sceneray at the horizon. We randomize 10 skyboxes to achieve this. Samples of skybox is as shown in figure 4.8.

4.1.6 Replacing target Objects

Once the room is setup using Scenenet **?** which contains objects from ShapeNet **?**, the objects are renamed such that it matches the classes for the Pix3D dataset. Since a given scene can have more than one category under observation, a single model is chosen randomly out of all the present models. The chosen model is replaced with a model in Pix3D of same category. In case the replaced model is intersecting with any other models in the scene, we make it invisible. Further, the camera checks if the centre of replaced model is in the frame. If not, the camera viewpoint is changed untill it satisfies the condition. After the snapshot is taken, the scene is reset to the

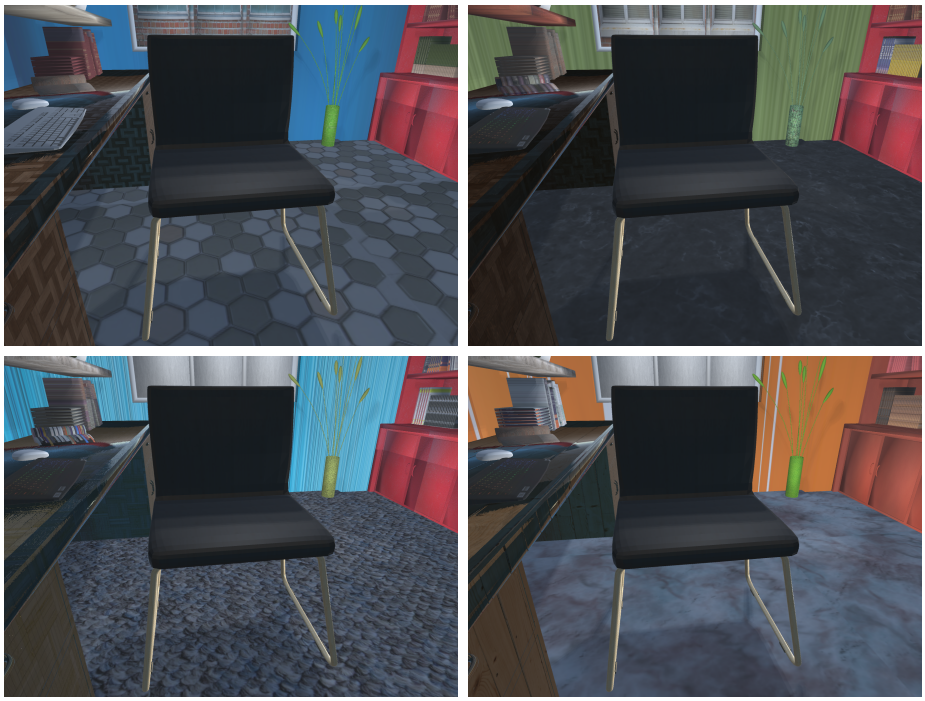


Figure 4.6: Sample images with different textures for same scene.

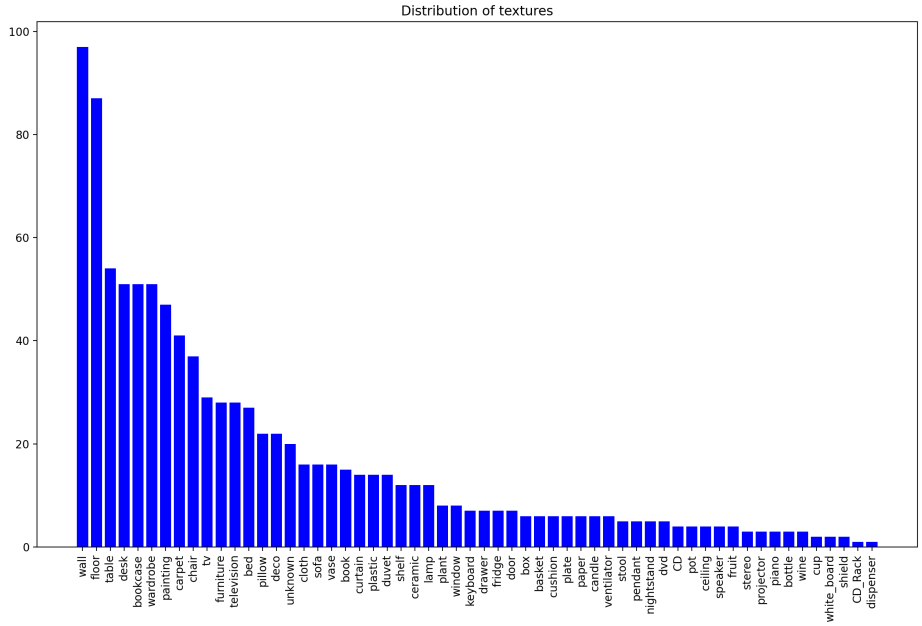


Figure 4.7: Distribution of textures used on scenes. The categories of pix3d(target furnitures) have higher number of images.

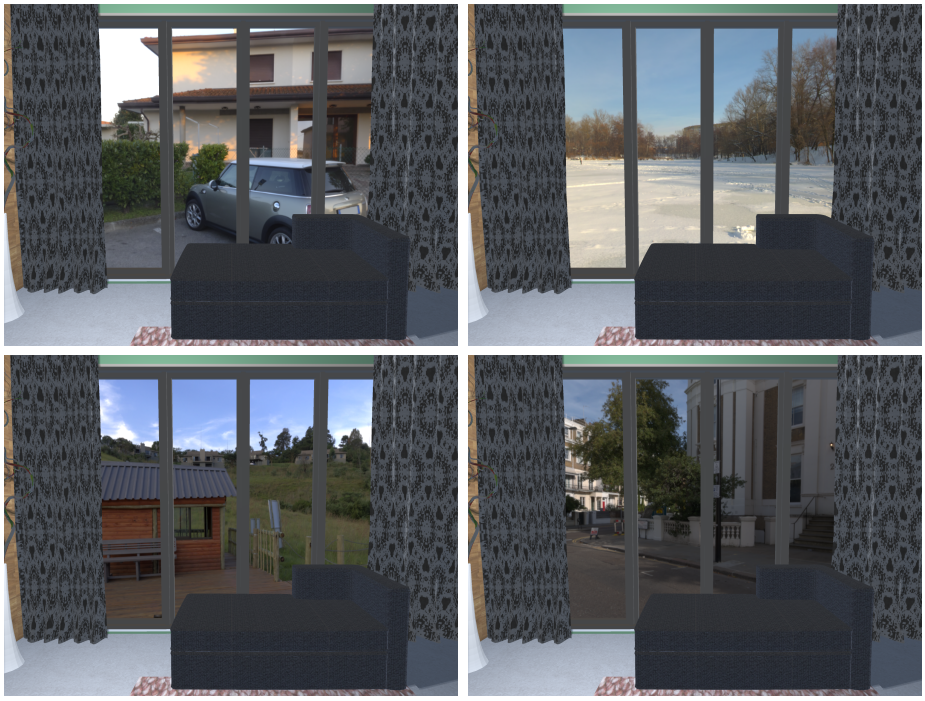


Figure 4.8: Samples for different skyboxes which change the outdoor environment for the scenes. In the figure we see an open window with changing skybox.

original and next model is imported. Samples for replacing a target object from original scene from Scenenet is shown in figure 4.9

4.1.7 Snapshots with ML-ImageSynthesis

ML-ImageSynthesis **?** is a library provided by Unity to support synthetic dataset creation. It supports the following G-buffers Image segmentation, Object categorization, Optical flow, Depth, Normals, etc. G-buffers is basically a collective term to represent light-properties which are agregated to give the final rendering. In image segmentation each object is assigned unique color. In object categorization each category of the objects is assigned a single color. For optical flow each pixel is colored depending on Unity's per-pixel Motion Vectors with respect to the camera. Depth is as the word suggests the distance of each pixel from the camera encoded in 8-bit channels of the PNG image. Normals are color encoding based on the orientation of object with respect to the camera. For the snapshot, each of these properties is assigned a hidden camera and have each one overrides rendering scene with custom shaders to generate corresponding output. These camera outputs can be seen in editor mode by changing the display window. Samples of G-buffer collected for dataset are as shown in figure 4.10.

Replacing models, lighting(indoor and outdoor), randomisation of texture, camera distance, camera angles, outdoor skybox, UI controls Manual inputs, automated Unity - ML-ImageSynthesis UML diagram is needed? (If it helps)

4.2 3d-Reconstruction framework

Development environment Training setup(hardware, voxel size. . .) Mixed precision using apex If mixed precision helped performance, memory, speed, evaluation metric

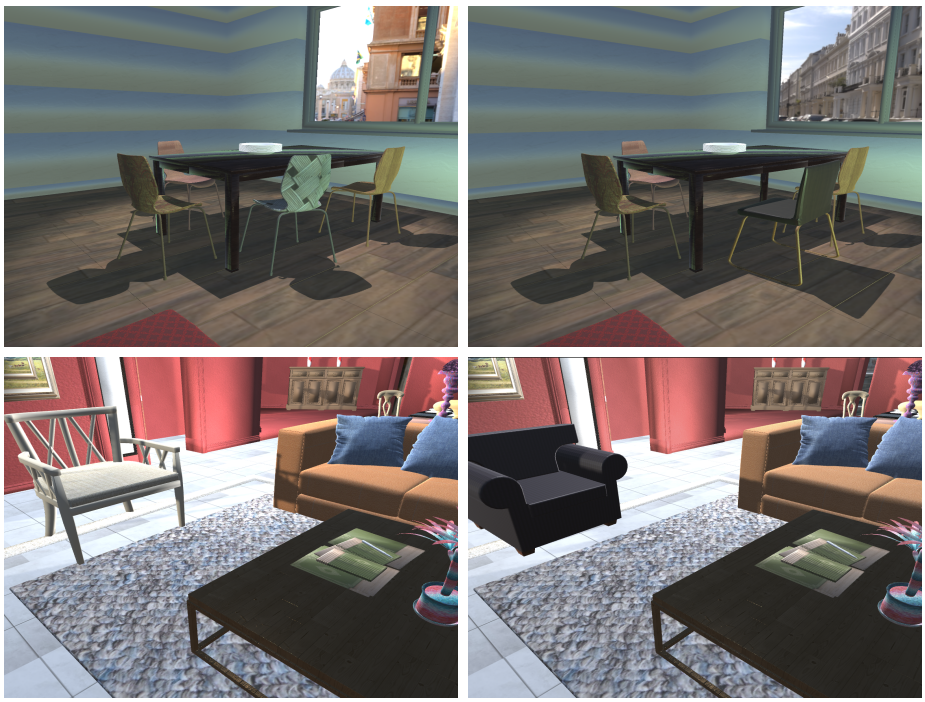


Figure 4.9: Samples for object replacement. The Left column shows a scene from SceneNet, while the right column shows an object being replaced in original scene.

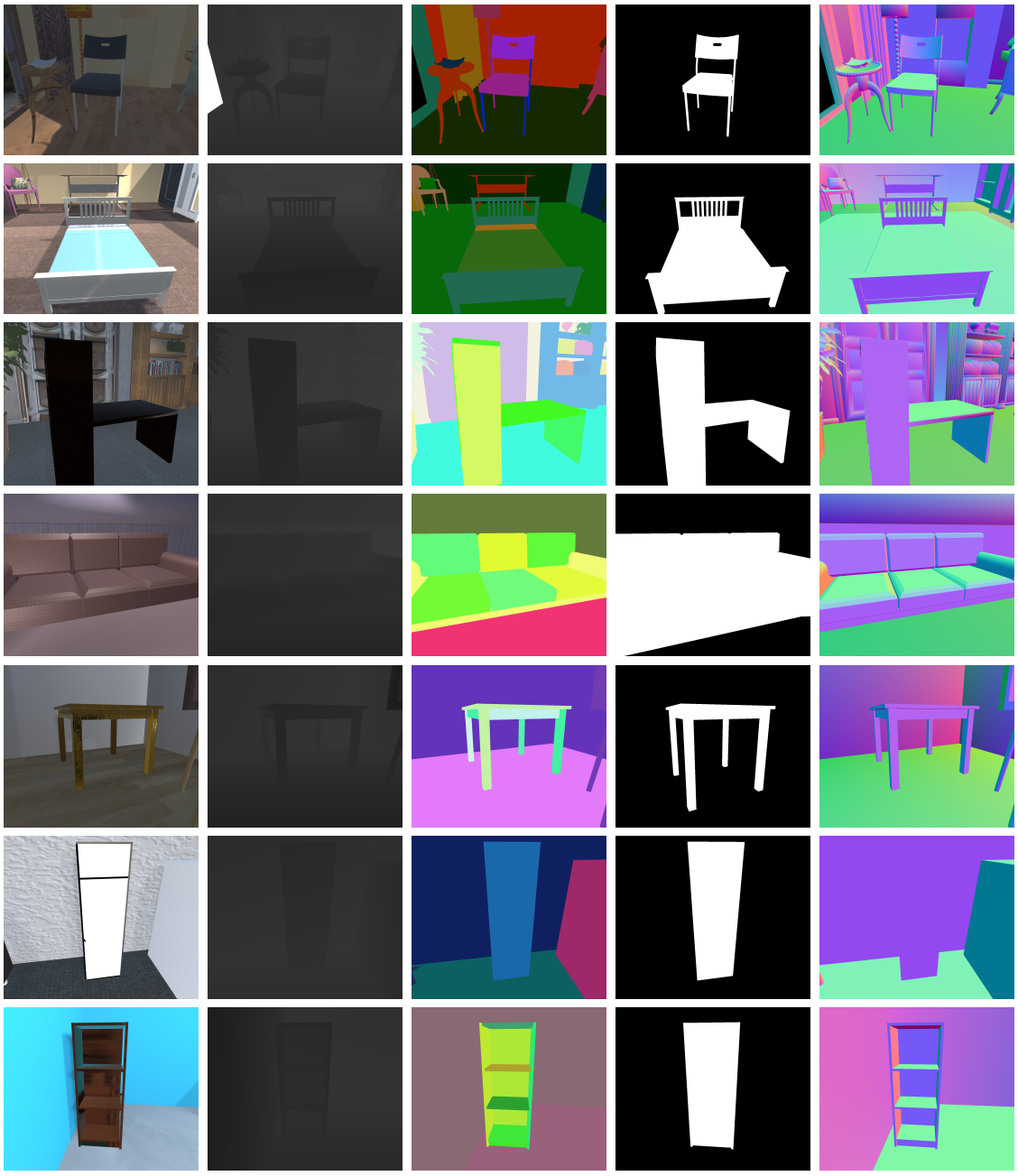


Figure 4.10: Samples for G-buffers collected from ImageSynthesis as part of the dataset. In the figure, (From left to right) RGB image, Depth map, Instance segmentation, Semantic Segmentation and Normal map

5. Evaluation

5.1 Domain gaps - Qualitative

Visualize domain gap between S2R:3D-FREE and real image(pix3d)

Visualize domain gap between other synthetic dataset like 3dfront, scenenet tsne

5.2 Domain gaps - Quantitative

Maximum Mean Discrepancy Kullback-Leibler divergence

5.3 Performance

Col1	Col2	Col2	Col3
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344

Performance of model trained on synthetic dataset by testing model with real dataset(pix3d) Qualitative: Voxel output comparisons Quantitative: IOU, Dice

5.4 Domain shift technique

Compare the differences in domain shift of models learnt on S2R:3D-FREE and a traditional domain shift learning.

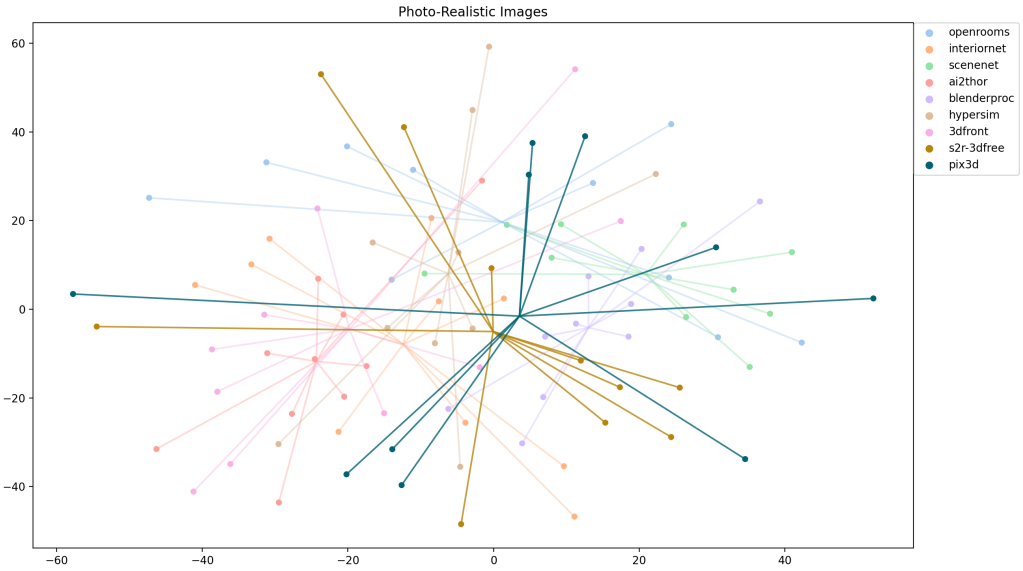


Figure 5.1: T-SNE visualisation for images from various photo-realistic synthetic dataset. Pix3d and S2R3DFREE is highlighted with bolder colors.

5.5 Ablation study on chairs

5.5.1 Domain randomisation on chair dataset

Check if randomising textures of chair and background helps improve performance. Example: Create dataset with constant room texture vs randomising texture, Check if lighting helps (indoor, outdoor). Example: Create a dataset with constant lighting(only outdoor lighting) vs indoor lighting. Iou per category: as in https://arxiv.org/pdf/1905.03678.pdf

6. Conclusion

6.1 Summary

6.2 Limitations of game engines

6.3 Future Work

Appendix

This table is automatically generated from a CSV file. Of course, you can also create tables from scratch.

	List_ Insert	List_ Search	Ordered_ Insert	Ordered_ Search	Ordered_ Min	Ordered_ Sort	Set_ Insert
coarse, Ex., Strict	2400	1418	10608	10824	2871	10169	23366
coarse, Ex., Def.	2400	1418	9652	20172	2871	14043	35884
coarse, Ex., None	2400	1418	14347	20160	2871	12687	54907
coarse, L.S., Strict	2400	1418	10608	10824	2871	10169	21047
coarse, L.S., Default	2400	1418	9430	20172	2871	13835	24688
coarse, L.S., None	2400	1418	13802	20160	2871	12687	28501
coarse, Complete	2400	1418	4504	10149	2871	10047	16114
coarse, Product	9600	2836	9010	20298	5742	20094	16114
fine, Ex., Strict	2211	1321	11210	16984	2501	6500	12901
fine, Ex., Def.	2211	1321	10299	16971	2501	6420	14695
fine, Ex., None	2211	1321	10881	16836	2501	7115	23005
fine, L.S., Strict	2211	1321	11210	16984	2501	6500	12301
fine, L.S., Default	2211	1321	10251	16971	2501	6420	13173
fine, L.S., None	2211	1321	11116	16836	2501	7115	20053
fine, Complete	2211	1321	2018	16531	2501	6613	9336
fine, Product	8844	2642	4036	33062	5002	13226	9336

Figure A.1: Caption in the text.

B. Tutorial

Acronyms like software product lines (SPLs) and first-order logic (FOL) are automatically added to the list of acronyms above. To reference a chapter, subsection, figure, etc., put a label after the command (see above) and then use Chapter 1.

You can mark stuff as TODO, which is useful for writing drafts.[3]

Or put notes to your advisors in the margin.

Theorem B.1: My Cool Theorem

You can also add definitions and theorems in mathematical theses.

Proof. You can also add proofs, though most theses do not require any. □

In Figure B.5, we show an example of how to typeset a complex figure only with LaTeX by using TikZ.

B.1 This is a section

For citing, put entries in the BibTeX file (one example is there) and use the following to reference the authors directly: **?** conducted a systematic literature review (SLR) (this is an acronym defined in the main file), or use the following for references in parenthesis [**?**]. **?** **?**

B.1.1 This is a subsection

Numerated list

1. One
2. Two
3. Three

[3] Footnotes are also possible, but are rarely used in computer science.

0.29

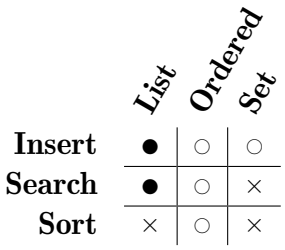


Figure B.1: List

0.21

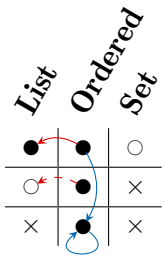


Figure B.2: List, Ordered

0.19

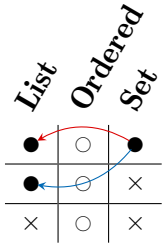


Figure B.3: List, Set

0.26

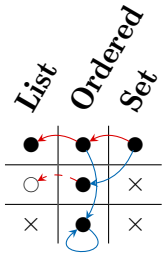


Figure B.4: List, Ordered, Set

Figure B.5: Drawing with tables and TikZ.

Table B.1: Long caption for a table.

Column1	Column2	Column3	Column4
Left	Right	Centered	Left, fixed width
		Multi column	
Multi row	X		
	Y		

Bullet list

- One
- Two
- Three

I herewith assure that I wrote the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or to their meaning.

Magdeburg, 15th October 2021