5-16-2024

An IoT-Based Approach of Synthetic Data Generation with Reduced Reality Gap

Kavan Mehulkumar Dave
University of Windsor

An IoT-Based Approach of Synthetic Data Generation with Reduced Reality Gap

By

Kavan Mehulkumar Dave

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2024

An IoT-Based Approach of Synthetic Data Generation with Reduced Reality Gap

by

Kavan Mehulkumar Dave

APPROVED BY:

M. Khalid
Department of Electrical and Computer Engineering

I. Ahmad
School of Computer Science

X. Yuan, Advisor
School of Computer Science

May 9, 2024

Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Computer simulation is a powerful technique to create synthetic images for the testing of computer vision algorithms. However, the discrepancies caused by inconsistencies between the simulated environment and the physical world have made the results of algorithms testing with synthetic datasets hardly reliable in real-world applications, which is a phenomenon called the "reality gap". Among various factors that impact on photo-realism of computer-synthesized outdoor environments, researchers have analyzed the most influential and identified them as geometry, appearance, lighting, physics, environment, camera, and rendering parameters.

This thesis research develops a novel system for the generation of synthetic datasets with a reduced reality gap. In the system, a scene environment is first constructed with the geometry of objects according to real-world data. Information retrieved for the time of a day from the Internet of Things (IoT) is then used by a simulator engine to render images with the actual appearance of objects, physics of shadows and reflections, and rendering parameters, such as camera, lighting, and environmental conditions. Similarity scores between the synthesized and real-world images are finally calculated to evaluate the effectiveness of the proposed system in reducing the reality gap.

Dedication

I dedicate this thesis work to God Almighty, my beloved parents Mr. Mehulkumar Dave and Mrs. Sujataben M Dave, my loving sisters Mrs. Riddhi H Bhatt and Mrs. Siddhi I Raval, and the rest of my family and friends.

Acknowledgements

I would like to sincerely express my most profound gratitude towards my supervisor Dr. Xiaobu Yuan, whose input helped me immensely. With his input, I was able to look at my research with a different perspective and a more critical eye. Secondly, I would like to express my gratitude to my thesis committee members internal reader, Dr. Imran Ahmad, and my external reader, Dr. Mohammed Khalid, for their time, effort, and for their beneficial advice and suggestions for my thesis. I humbly extend my thanks to the School of Computer Science and all concerned people who helped me in this regard.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

This chapter introduces topics like computer vision, object recognition, image dataset and data generation, generative AI, challenges involved with image data and simulation. The introduction helps in understanding the main challenge handled in this study of generating realistic simulated outdoor image data which can be used for training different computer vision and artificial intelligence algorithms. Following the introduction, this study delves into related research in the Chapter 2, outlines the research methodology in the Chapter 3, presents experiments and analysis in the Chapter 4, and concludes with conclusion and future works in the Chapter 5. At the end, this study contributes by developing a system for generating simulated images with a reduced reality gap through enhanced data curation methods and defined metrics to measure the reality gap between the simulated and the actual real world scenario.

1.1 Computer Vision

Computer vision is a dynamic and rapidly evolving field within the realm of artificial intelligence (AI), garnering significant attention from researchers and practitioners alike. Computer vision stands out as one of the foremost research domains in AI, characterized by its ability to empower machines with visual perception capabilities akin to human vision. Through sophisticated algorithms and advanced techniques, computer vision enables computers to "see", interpret, and comprehend visual data, thereby facilitating tasks such as image recognition, description, classification, and

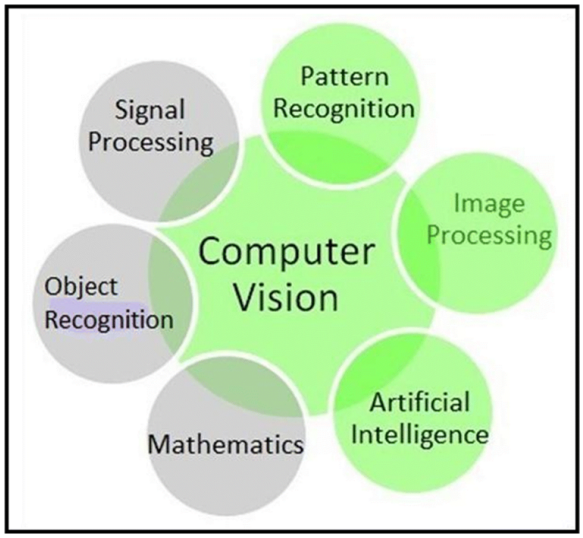


Fig. 1: Influence of Computer Vision in Other Areas [33]

pattern grouping [33].

The pivotal role played by computer vision transcends disciplinary boundaries, finding application and relevance across diverse domains including biological sciences, biomedical engineering, medicine, marketing, robotics, computer graphics, artificial intelligence, and remote sensing. The impact of computer vision extends beyond mere image analysis, it underpins critical functions and processes in various industries and scientific endeavors.

In recent years, the advent of deep learning frameworks and convolutional neural networks (CNNs) has revolutionized the field of computer vision, enabling unprecedented advancements in image understanding, object detection, semantic segmentation, and scene recognition. These breakthroughs have propelled computer vision into the forefront of technological innovation, driving transformative changes in sectors such as autonomous vehicles, healthcare diagnostics, surveillance systems, augmented reality, and industrial automation.

Figure 1 illustrates the multifaceted domains and applications influenced by computer vision, showcasing its pervasive impact on modern society and industry. As researchers continue to explore novel methodologies, algorithms, and applications within the realm of computer vision, the potential for groundbreaking discoveries and transformative solutions remains boundless.

In essence, computer vision represents a cornerstone of AI research, offering profound insights and capabilities that redefine human-computer interaction, decision-making processes, and problem-solving paradigms in the digital age.

1.2 Object Recognition

Object recognition is a fundamental aspect of computer vision, encompassing the ability of a system to identify and classify objects within visual data accurately. Object recognition is a multifaceted process that involves several sequential steps, each contributing to the overall performance and efficacy of the recognition system [33]. The steps of object recognition according to Rani et al.[33] are as follow.

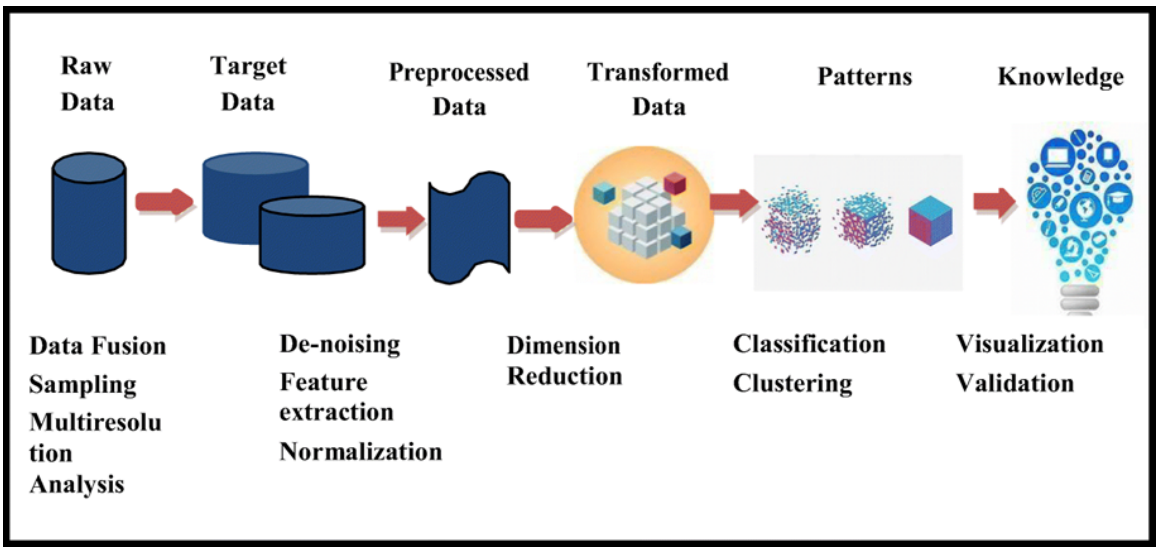


Fig. 2: Basic Steps of Pattern Recognition [33]

The initial step in object recognition typically involves data acquisition, where raw data in the form of images or videos is collected from various sources such as sensors, scanners, or digital machines. This data may undergo transformation and preprocessing to enhance its quality and suitability for subsequent analysis.

Data preprocessing, the second step in the object recognition pipeline, plays a crucial role in preparing the data for analysis. This phase involves tasks such as isolating the region of interest (ROI) from the background, filtering out noise and unwanted artifacts, and normalizing the data to ensure consistency and comparability.

Subsequently, the data analysis step involves the learning and training of the data using machine learning or deep learning algorithms. During this phase, the system extracts meaningful features and patterns from the data, building a knowledge base that will aid in the classification of objects in later stages.

The classification step is where the object recognition system leverages its learned knowledge to assign categories or labels to new input data. This decision-making process is informed by the training and analysis conducted earlier, enabling the system to make accurate predictions about the identity of objects within the visual input.

It's worth noting that the performance of an object recognition system is influenced by various factors [33]. These factors include the size and quality of the dataset used for training, the methodology and technology employed in the recognition system, the expertise of the system designer in selecting appropriate algorithms and parameters, and the specific needs and expectations of the end-users who will interact with the system.

In essence, object recognition represents a critical capability within computer vision, enabling machines to interpret and understand visual data in a manner that mimics human perception. Advances in object recognition techniques have led to significant progress in diverse fields such as autonomous navigation, image retrieval, surveillance, robotics, and augmented reality, paving the way for innovative applications and solutions in the digital era.

1.2.1 Algorithms for (3D) Object Recognition

Table 1 lists and describes different algorithms for 3D object recognition, including classification, clustering, regression, sequence labeling and parsing.

Name	Description
Classification	The system is designed for supervised machine learning, where the algorithm learns from input patterns and derives desired outputs under the supervision of a supervisor. The main goal is to accurately label new test data based on the training phase.
Clustering	Cluster analysis groups similar objects together, representing unsupervised machine learning. Unlike supervised learning, there's no need for model supervision, allowing it to discover patterns in unlabeled data. While unsupervised learning handles more complex tasks, accuracy may be lower.
Regression	Regression analysis predicts the dependent variable using known values of the independent variable. The dependent variable is the outcome, while independent variables are features or predictors. Linear regression is a common method in this analysis, primarily used for prediction and forecasting in machine learning.
Sequence labeling	This pattern recognition task assigns a class or label to each element in an observed data sequence, often a crucial step in NLP tasks. Examples include speech tagging, POS (part of speech) tagging, and NER (Name Entity Recognition). Label sequencing learning predicts sequences of labels rather than individual class labels.
Parsing	It is a process of examining the string of symbols in computer language, data structure, or natural language. In terms of computer science, this term is used to analyze the language of computers and is referred to as syntactic analysis. According to the syntactic technique, a pattern is a combination of simple sub-patterns.

Table 1: Algorithms for (3D) Object Recognition [33]

1.2.2 An Example: Enhancing Decision-making

An autonomous vehicle equipped with cameras and other sensors captures real-time images of the road. These images are processed by the vehicle's AI system, which

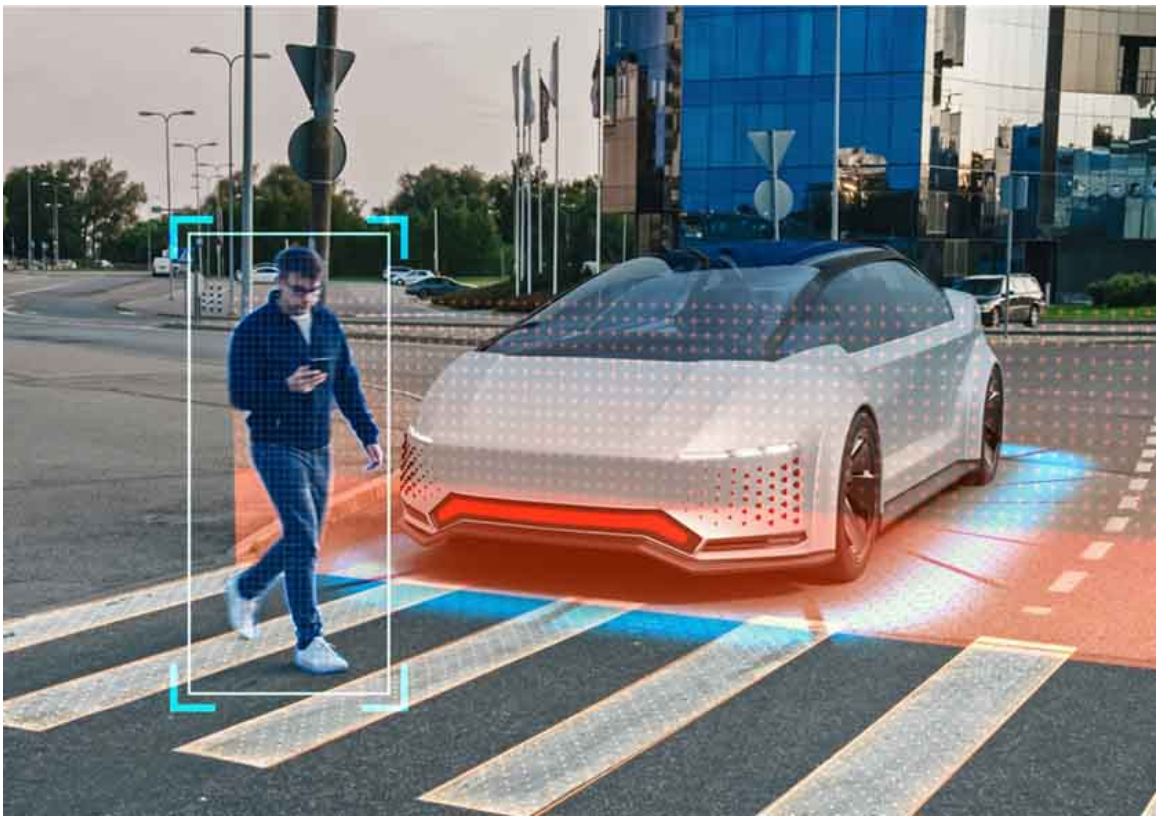


Fig. 3: Use of Object Recognition in Decision-Making [15]

analyzes the data to recognize objects, interpret the environment, and make decisions accordingly.

If the system detects a pedestrian crossing the road, it can autonomously stop or slow down to ensure safety. Here, the use of object recognition techniques helps decision making and thus images play a vital role in enhancing decision-making across various fields, one example being autonomous vehicles.

1.3 Image Datasets and Data Generation

Image dataset generation is a critical aspect of training deep learning models in computer vision. Over the past decade, the evolution of computer vision has shifted towards deep learning-based models that learn from extensive labeled data, leading to more accurate and robust systems. However, the cost and effort involved in annotating each example have increased significantly, especially as the focus expands to complex

tasks that require labeling objects or even pixels in images [2].

Traditional methods of data collection using photography devices yield high-quality data but are time-consuming and costly. As a remedy, researchers have turned to synthetic image data generation techniques [10]. Synthetic data generation involves creating virtual environments where training examples are generated in a controllable and customizable manner, circumventing the challenges of real-world data collection and annotation.

Synthetic data generation employs various methods such as simulation tools, AI models, and rendering engines to produce diverse datasets efficiently. One of the primary advantages of synthetic data generation is its reduced time and cost compared to traditional methods. While the effectiveness of synthetic data initially may be lower than real-world data, preprocessing, human verification, and rigorous testing can significantly enhance its effectiveness and utility in training deep learning models.

By leveraging synthetic image data generation techniques, researchers and practitioners can access larger and more diverse datasets, accelerating the development and deployment of robust computer vision systems. This approach not only streamlines the data collection process but also offers greater flexibility and scalability in training models for a wide range of applications across industries such as robotics, healthcare, automotive, and more.

1.4 Generative AI: Image Generation

Generative AI, with its diverse capabilities ranging from text and image generation to voice synthesis and style transfer, represents a groundbreaking advancement in artificial intelligence (AI). generative AI encompasses a wide array of applications, each showcasing its ability to create and manipulate content in innovative ways [12].

In the realm of image generation, generative AI leverages sophisticated algorithms and neural networks to produce visually appealing and contextually relevant images. One of the notable features of generative AI is its flexibility in generating images based on different inputs. For instance, images can be generated by providing textual

prompts that describe the desired scene or by using existing images as references along with text prompts, allowing for a more nuanced and personalized approach to content creation [34].

While generative AI has demonstrated remarkable performance in generating portraits, artworks, and stylized images, there are still challenges when it comes to generating realistic scenes comparable to real-world images. The complexity of capturing intricate details, spatial relationships, and environmental factors in scenes remains a frontier that generative AI continues to explore and improve upon [12][34].

Figure 4 shows two images that are presented side by side, with the first image provided as a prompt and the second image generated using AI technology. The resulted image turned out very well. The AI generated image has nature of an artwork. An artwork differs from real image with details like edges, color selections and lighting. Although the results in the Figure 4 (b) seems impressive, generative AI faces few challenges which are discussed in Section 2.6.

(a) Reference Image Given with the Prompt (b) Image Generated Using Generative AI

Fig. 4: Images Generated Using Generative AI

1.5 Challenges Involved with Image Data

Training and testing object recognition models entails a complex and resource-intensive process, starting with the acquisition and labeling of large volumes of real-world data. This data often contains uncertainties, biases, and nuances inherent to real-world scenarios, necessitating meticulous labeling and domain expertise. Subsequently, the

trained models undergo rigorous testing to assess their performance, requiring evaluation in both common and rare situations to ensure robustness [26]. This comprehensive evaluation process is vital but adds to the complexity and time investment involved in designing, implementing, and testing object recognition systems under realistic environmental conditions. Consequently, researchers and practitioners face significant challenges in navigating this arduous, costly, and often non-reproducible process, hindering advancements in the field [26].

Moreover, the necessity for continual training with fresh real-world data to enhance model decision-making capabilities over time is evident, particularly in applications like autonomous vehicles. However, the practicality of acquiring large volumes of labeled real-world data at regular intervals presents a formidable challenge. This process is not only time-consuming but also laborious, especially in domains like computer vision, where detailed labeling and data curation are crucial for effective training and testing. Thus, while ongoing training is crucial for improving decision-making techniques in computer vision models, the logistical hurdles associated with gathering and managing substantial datasets regularly remain a significant obstacle in advancing the field [16][38].

1.6 Simulation

Simulation plays a pivotal role in generating substantial volumes of training data affordably and efficiently within lifelike environments [2]. By leveraging simulation techniques, researchers and practitioners can create synthetic data that closely mimics real-world scenarios, offering numerous advantages in the realm of computer vision and artificial intelligence. As computer graphic software tools have evolved, the computer vision community has increasingly turned to simulation as a cost-effective, efficient, and scalable solution for data collection in training [35]. Simulation involves approximating real-world scenes within a computer graphic software environment, creating what is termed a simulation scenario. This scenario is then processed to generate simulated images. The simulation process typically starts with replicat-

Fig. 5: Simulated Image Using Blender Simulation Tool [39]

ing real-world objects and their environments, while also accounting for the physical properties that define these objects and settings. For example, Figure 5 illustrates a simulated indoor scene featuring various object models and environmental elements like daylight and weather conditions. These properties are effectively modeled using components such as physics engines, modeling tools, environment modules, rendering engines, and application programming interfaces (APIs). With the definition of real-world objects and environments, users can leverage the API modules to generate simulated images at scale, reducing the need for manual effort. As users have knowledge of the objects and environments being simulated, they can programmatically obtain accurate labels without human intervention. Computer graphic software tools are employed to generate these simulated images, which serve various purposes, including training computer vision [16] and object recognition algorithms and creating computer-animated videos or multimedia content. Notable software tools for simulation, such as Unity[41], Cinema4D[9], and Blender[7], have been utilized for generating simulated datasets for tasks like object recognition.

CHAPTER 2

Related Researches

This chapter discusses the relevant background and various methods for synthetic data generation. Furthermore, we discuss various computer graphics software tools used to generate simulated data. Lastly, we discuss about image similarity and the model utilised for comparing two images.

2.1 The Problem of Reality Gap

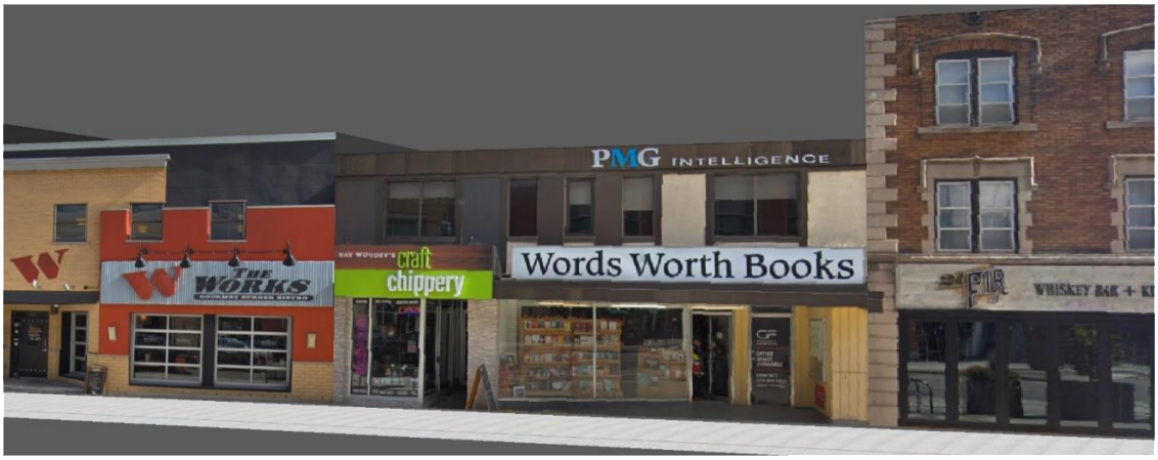
In the context of autonomous vehicle driving, simulation suites offer invaluable capabilities for automated validation techniques across diverse scenarios, facilitating the development of closed-loop validation methods like machine learning and reinforcement learning approaches. However, simulation tools encounter a significant challenge known as the Reality Gap [4].

(a) Simulated Image (b) Real-World Image

Fig. 6: Indoor Object Models in a Shelf Environment [11]

The Reality Gap represents the disparity between simulation conditions and real-world scenarios [3]. Bridging this gap is crucial to enhance the effectiveness of synthetic data generation and simulation-based validation processes. The difference that results in the problem of reality gap is due to the incorrectly simulated physical properties of scene. These physical properties are considered influential factors include texture, material, illumination (lighting condition), and weather conditions [43]. To minimize the Reality Gap, simulators must strive to create scenes that are highly realistic, mirroring real-world settings with precision.

(a) Real Image



(b) Simulated Image

Fig. 7: Simulation of an Outdoor Scene [16]

If models are tained with poorly simulated data which has large reality gap it effects models performance negatively. Poorly simulated data may not accurately

represent the real-world distribution of images. As a result, models trained on such data may exhibit bias and struggle to generalize to real-world scenarios [39][38][26]. If the simulated data does not capture the diversity and complexity of real-world images, the model may overfit to the training data, resulting in poor performance on unseen data.

The simulation process has to capture and handle various details in order to appear the same as it real world match. In Figure 6, the images look same but if the focus go to the background and its material then two scene appears as different. However in Figure 7 (b), the building objects do look close but their shape and structure as well as background makes them look different. Although these are the examples with visible differences for understanding, there are various other factors like object material, scene lighting etc. which might not visible to the human eyes but if caught by model then affects the model significantly.

Hence, it becomes imperative to develop a system capable of comparing simulated images with real-world counterparts to assess their similarity. Such a system will provide valuable insights on enhancing the effectiveness of simulated datasets and optimizing the performance of object recognition algorithms in real-world testing scenarios.

2.2 Synthetic Data Generation

Synthetic data refers to images generated using simulation techniques, yet differences between simulated and real scenarios persist due to rendering performance and scene modeling challenges. While high-quality realistic images can be generated, accurately simulating object details, materials, lighting, camera parameters, and imperfections remains a hurdle [43].

Simulation and animation primarily focus on 3D scenes to create synthetic datasets [30], aiming to bridge the Reality Gap. To reduce this gap effectively, it's crucial to prioritize rendering engines, accurately model object characteristics, adjust physics and environmental factors. By implementing these measures thoughtfully, the Real-

ity Gap in synthetic data can be minimized, leading to more effective and reliable simulation outcomes.

2.3 Influential Factors

In the context of synthetic image generation, influential factors refer to the key elements that significantly affect the quality, realism, and effectiveness of the generated images. Therefore, it is important to tackle these elements carefully and precisely in order to increase the realism of synthetic image data [30]. The identifield inflential factors are presentated in Table 2

1. Physics	12. Fracture
2. Rigid bodies	13. Atmospheric conditions
3. Physical soil	14. Wind
4. Deformable objects	15. Rain
5. Fluid dynamics	16. Snow
6. Thermodynamics	17. Clouds
7. Obstacles	18. Crowds
8. Magnetism	19. Seasonal changes
9. Soft bodies	20. Perturbations
10. Cloth	21. Object deformation
11. Slicing	

Table 2: Identified Influential Factors [30]

Table 2 considers influential factors for various indoor as well as outdoor environment. however, there are factors which affects the scene irrespective of its environment nature(Indoor or Outdoor) and these factors are discussed below.

- Lighting Conditions: The way light is simulated in the synthetic environment, affecting shadows, highlights, and overall visual appearance.
- Object Placement: The positioning and arrangement of objects within the synthetic scene, influencing spatial relationships and interactions.
- Time : This factor involves accounting for changes over time, such as dynamic lighting conditions, object movements, or alterations in the overall scene composition.
- Sky Color: The color of the sky and the way it interacts with the scene's lighting conditions impact the overall color palette and mood of the synthetic image.
- Weather Conditions: Incorporating weather effects, such as rain, snow, or fog, in the sky adds a dynamic aspect to the scene, enhancing the diversity of the synthetic dataset.

2.4 Internet of Things (IoT)

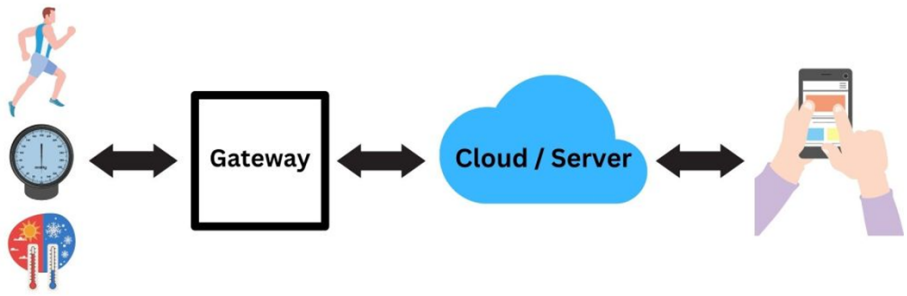


Fig. 8: IoT System Architecture [6]

The "Thing" in IoT can be any device with any type of sensor embedded with the ability to collect data and transmit it across the network without manual intervention. Internet of things systems such as networked vehicles, smart traffic systems, and sensors embedded in roads and bridges bring us closer to the idea of "smart cities", which help reduce congestion and energy consumption [24]. Once created, IoT network can add a number of different sensors and devices, including mobile phone-based

sensors(Accelerometer, Gyroscope, GPS etc.), Medical Sensors, Specialized Physical, Mechanical, and Chemical Sensors, Radio Frequency Identification (RFID) etc.

At the core of modern Smart City initiatives lies the Internet of Things (IoT), a technological framework that enables seamless connectivity and data exchange between various devices and systems. IoT plays a pivotal role within the context of Smart Cities by facilitating the collection and transmission of data from sensors deployed throughout the urban [8]. These sensors capture real-time information about the city's infrastructure, environment, and services, sending this data to a centralized cloud platform. In the cloud, the data is processed, analyzed, and mined to extract patterns and insights that inform decision-making processes aimed at optimizing city operations and improving quality of life for residents. Key components of Smart Cities, such as transportation systems, energy grids, waste management, public safety, and environmental monitoring, can be seamlessly integrated with IoT through the deployment of specific sensors tailored to each domain. This integration enables cities to leverage data-driven strategies for efficient resource management, sustainability initiatives, and enhanced citizen services [8].

2.5 Synthetic Data Generation Methods

Table 3 provides an overview of different methods for generating synthetic image data. The table categorizes the synthetic image data generation methods based on their techniques and highlights their key characteristics.

2.6 Generative AI for Synthetic Image Generation

Image generation using generative AI has revolutionized the field of computer vision and creative content generation. These models have shown impressive capabilities in creating synthetic images that closely resemble real ones. As these models are such powerful tools, is it reliable to use them for generating synthetic image data? The answer is not really due the phenomena called Model Collapse. It is a degenerative

Method	Description
The method of using the generator [31]	The method relies on an existing generator that can be pre-programmed or built using a tool for digital content creation or a physics or game engine. The generator allows the user to change the synthesis parameters, and the output does not necessarily generate in real-time.
programming method [19]	The synthetic data is created algorithmically, by direct programming of the output.
The method of using the simulator [17]	This technique relies on the programming of the respective or the use of a game engine or a modified commercial computer game to conduct a specific simulation.
The digital content creation method [37]	This method relies on tools, such as 3ds Max, Blender or After Effects, that often contain a physics engine and enable automation by programming.
Use of commercial computer game method [40]	It depends on the license which regulates whether the game may be used to generate a synthetic data. If it can, the biggest problem is the way to access the content within the game, possibly adapt it to own needs and output it in the appropriate form (final image with the corresponding annotation).
Use of game engine [36]	The technical basis of every computer game is the game engine such as Unity or Unreal Engine, which is, when used as a production tool, also a distinct method that can produce synthetic data without first making a game. Game engine method allows direct integration with machine learning platforms.
Generative adversarial training [44]	It uses neural networks (GANs) during the synthetic data generation process but is limited by the inability to create appropriate annotations automatically and is more suitable for use as an auxiliary tool for domain adaptation.

Table 3: Synthetic Data Generation Methods

process affecting generations of learned generative models, where generated data end up polluting the training set of the next generation of model, being trained on polluted data, they then misperceive reality [38].

When an AI model is used to generate synthetic image data, the data it produces might have captured noise. If the same data is used to train another model, there are

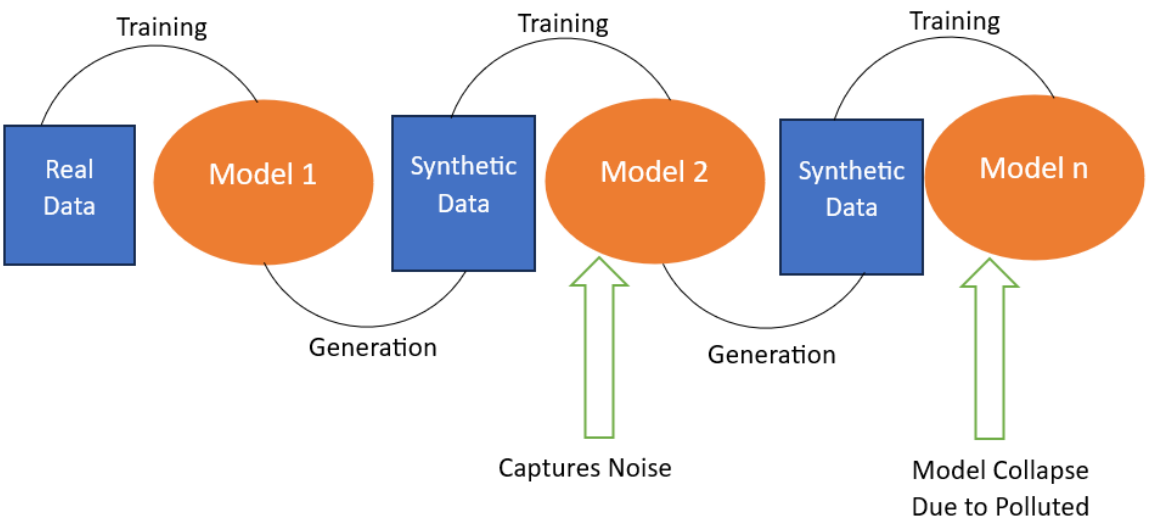


Fig. 9: Model Collapse

high chances of these noises in training data affect the model learn negatively. Now if the data from this second model used for training of other model, the same effect continues and at the end there come a n model which when use this polluted data it collapses. This phenomena is represented in Figure 9 and is called model collapse.

2.7 Simulation Engines

2.7.1 Blender

(a) Scene with indoor lighting condition and clear weather

(b) Scene with outdoor lighting condition and clear weather

Fig. 10: 3D Scenes Developed in Blender [39]

Blender is a powerful open-source 3D creation suite widely utilized for modeling,

animation, rendering, compositing, and more. Developed by the Blender Foundation, this versatile software has garnered significant attention within the computer graphics community due to its extensive feature set, intuitive interface, and active user community. Originally conceived as an in-house tool for a Dutch animation studio, Blender has evolved into a comprehensive platform that caters to the needs of professionals, enthusiasts, and researchers alike. One of the key distinguishing features of Blender is its accessibility. Being open-source, Blender is freely available for download and use, empowering users worldwide to unleash their creativity without financial barriers. Moreover, Blender's user-friendly interface and extensive documentation facilitate a smooth learning curve, making it an ideal choice for beginners venturing into the realm of 3D content creation. Blender's capabilities extend beyond mere modeling and animation. With its integrated rendering engine, cycles, and powerful node-based compositing toolset, Blender enables users to create stunning visual effects and life-like renders with ease. Furthermore, Blender's robust Python API opens up endless possibilities for customization and automation, allowing users to tailor the software to their specific requirements and integrate it seamlessly into their workflows [45][7]. The scenes created using Blender are represented in Figures 10, showcase diverse lighting and weather conditions. Blender's versatility extends to the development of datasets for indoor object recognition, autonomous driving, and robotic object detection tasks.

2.7.2 Unity

Unity is a powerful and versatile cross-platform game engine developed by Unity Technologies. It is widely used by game developers, architects, engineers, filmmakers, and other professionals to create interactive 2D and 3D experiences. Unity provides a comprehensive suite of tools for designing, developing, and deploying applications across multiple platforms, including PC, mobile devices, consoles, and augmented reality/virtual reality (AR/VR) headsets. Unity's key features include Graphics Rendering, Physics Simulation, Scripting Support, Asset Store that provides a vast library of ready-made assets, Cross-Platform Development, Networking capabilities for creating multiplayer games and online experiences and last but not the least it supports

Fig. 11: Outdoor Environment Scene Simulation in Unity Engine [42]

development for augmented reality (AR) and virtual reality (VR) applications, making it a popular choice for creating immersive experiences in these emerging technologies.Unity has gained widespread adoption in both the gaming industry and beyond, thanks to its user-friendly interface, extensive documentation, and active community support. Whether you're a seasoned game developer or a newcomer to interactive design, Unity provides the tools and resources you need to bring your ideas to life [41][42][39]. Figure 11 shows an outdoor simulation done using unity.

2.7.3 Cinema 4D

Cinema 4D, developed by Maxon Computer GmbH, has established itself as a premier software solution for 3D modeling, animation, and rendering in the creative industry. Its popularity stems from its user-friendly interface, powerful toolset, and versatility, making it a go-to choice for professionals across various fields such as motion graphics, visual effects, product design, and architectural visualization. One of the standout features of Cinema 4D is its intuitive workflow, which allows artists to seamlessly navigate through its robust suite of tools, making complex tasks more manageable and accessible. Whether creating intricate models, dynamic animations,

(a) Real Image (b) Simulated Image

Fig. 12: Cinema 4D Simulation [20]

or stunning visual effects, Cinema 4D offers a comprehensive set of features tailored to meet the demands of modern 3D production. At the core of Cinema 4D's appeal is its extensive toolset, which covers every aspect of the 3D content creation pipeline. From sophisticated modeling tools that enable artists to sculpt detailed objects with precision, to advanced animation features that facilitate the creation of lifelike character movements and dynamic simulations, Cinema 4D provides artists with the tools they need to bring their creative visions to life. Furthermore, Cinema 4D's rendering capabilities are second to none, with built-in render engines such as Physical Render and ProRender delivering high-quality, photorealistic results. Additionally, Cinema 4D seamlessly integrates with other industry-standard software packages, allowing for smooth interoperability and efficient workflow integration. With its combination of intuitive design, powerful features, and seamless integration, Cinema 4D remains a top choice for artists and designers seeking to push the boundaries of 3D creativity [9][20][39].

2.8 Image Simulator

As discussed in the previous section, there exist various simulation engines or image simulators using which simulated scenes can be generated. These simulators have number of tools and features using which realistic animation can be created. The number and variety of features provided vary in each of the simulators, but they all

share the same key modules as shown in Figure 13.

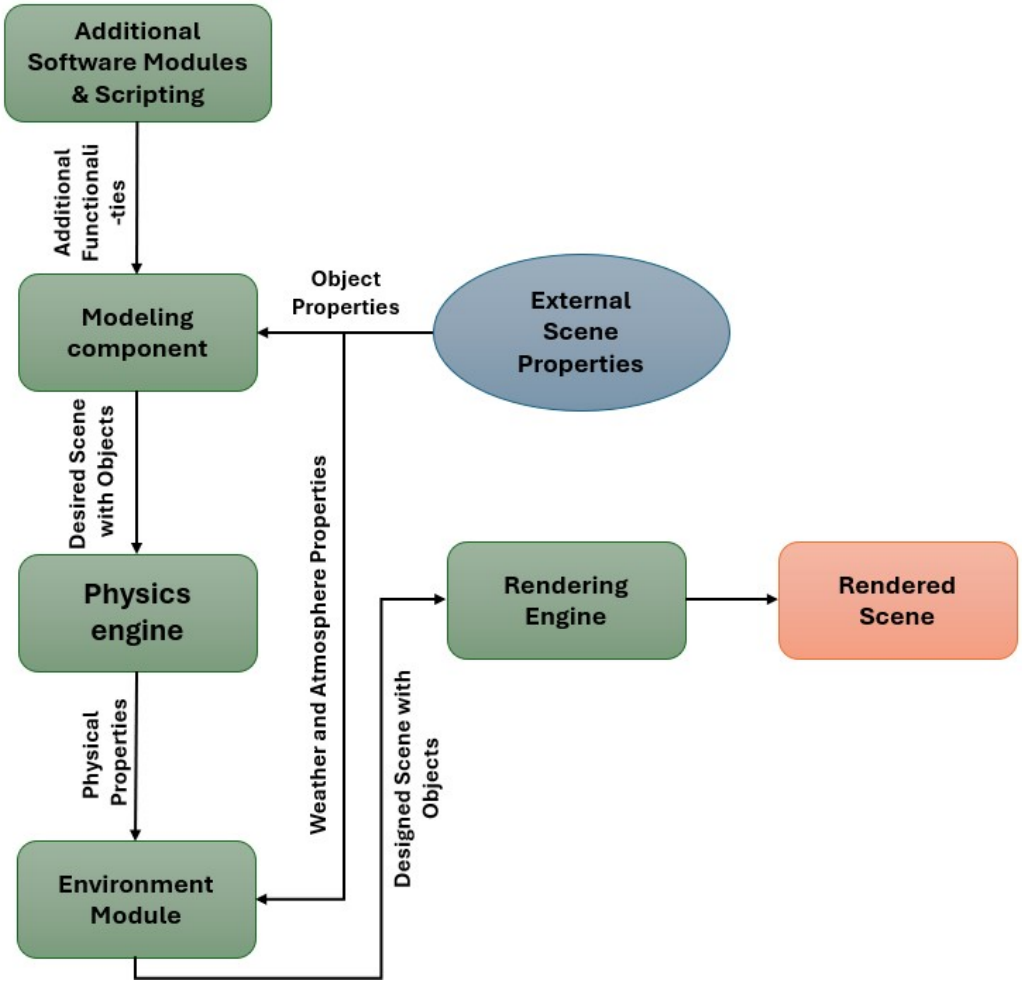


Fig. 13: Image Simulator Modules

Among the different modules, a Modelling Component It provides an interface for generating three-dimensional models and has support for a variety of geometric primitives, including polygon meshes, subdivision surface modeling, curves and metaballs. Also, it provides diverse modifiers and sculpting tools to model imperfections.

Physics Engine provides an approximate simulation of certain physical systems such as enabling gravitational properties for objects, rigid body dynamics, soft body dynamics and fluid dynamics.

Environment Module allows us to define a variety of weather and lighting conditions as it is very important to test object recognition in different conditions. This module allows to simulate different properties of environment like external and indoor

lights, weather, fluid simulator for simulating liquids, smoke simulation which will be used to model the simulated scenes.

Rendering Engine is responsible for rendering photorealistic imagery. This module efficiently renders shadows, indirect lighting, and ambient occlusion. Many rendering engines provide GPU support to enable efficient and fast rendering of scenes.

Scripting and Add-ons enable users to automate tasks, create custom tools, and extend functionality. Add-ons in a simulator refer to additional software modules or extensions that users can install to enhance the functionality or capabilities of the simulator. These add-ons are developed either by the official development team or by third-party developers and can offer a wide range of features, tools, or functionalities that are not included in the core software.

2.8.1 Image Similarity and OpenAI Clip

Image similarity refers to the degree of likeness or resemblance between two images. It is a measure of how closely the visual content of one image matches that of another. Image similarity is a fundamental concept in computer vision and image processing, used in various applications such as image search, duplicate detection, and content recommendation systems. Algorithms for measuring image similarity analyze the features, patterns, and structures present in images to determine their level of similarity, enabling efficient organization and retrieval of images from large datasets [13][30].

OpenAI's CLIP (Contrastive Language-Image Pretraining) is a cutting-edge model that has revolutionized the field of artificial intelligence by bridging the semantic gap between images and text. Unlike traditional computer vision models that rely solely on pixel-level features, CLIP leverages the power of natural language processing to understand the content of images at a more conceptual level. By pretraining on a massive dataset of images and associated text from the internet, CLIP learns to associate images with their textual descriptions in a way that captures their underlying semantics and context. This enables CLIP to perform a wide range of tasks, including image classification, zero-shot learning, and most notably, image similarity comparison [32].

One of the most compelling applications of CLIP is its ability to check the similarity of two images based on their content. By encoding both images and textual descriptions into a shared semantic space, CLIP can effectively measure the similarity between them, even if they belong to different categories or have different styles. This capability opens up a plethora of possibilities in various domains, such as content moderation, image retrieval, and creative applications. For example, CLIP can be used to identify visually similar images in large datasets, detect duplicate content across the web, or even generate artistic compositions based on a user's textual input. With its ability to understand images in a more nuanced and contextually rich manner, CLIP represents a significant leap forward in the field of multimodal AI and has the potential to drive innovation in numerous fields [28].

CHAPTER 3

Research Methodology

This chapter discusses the methodology for generating highly realistic and detailed geometric models for objects of interest with realistic lighting conditions, different weather conditions, variable and static objects, materials, and textures. The results produced using this methodology will make the simulated scene appear closer to reality.

3.1 Problem Statement

Simulation environments have emerged, and they are used as an efficient, safe, and cost-effective solution for generating large-scale synthetic data. Simulation environments have been used to generate data where objects are set in the indoor environment. However, when used for outdoor environments, they are used to imitate real-world scenarios, but minimal emphasis is given to designing real-world objects. Considering the problem of reality gap, it is essential to evaluate the most influential factors that can bridge the difference between real-world and simulated outdoor scenarios.

3.2 Contributions

- Develop an end-to-end system to generate photo-realistic simulated images using object models and environment scenes with matching physical properties.
- Investigate and get real-time data of influential factors from sensors and devices

connected to IoT networks.

- Improve the process of simulated data curation by configuring physical properties of object models and environment scenes, which assists in defining realistic scenarios suitable for training Computer Vision and AI models.
- Define metrics to measure the reality gap between the simulated and the actual real-world scenario.

3.3 System Architecture and Overall Process

The system architecture is illustrated in Figure 14. At the beginning, all the required add-ons are added to the Blender simulator to extend the functionality. The add-ons we used are listed in the Table 4.

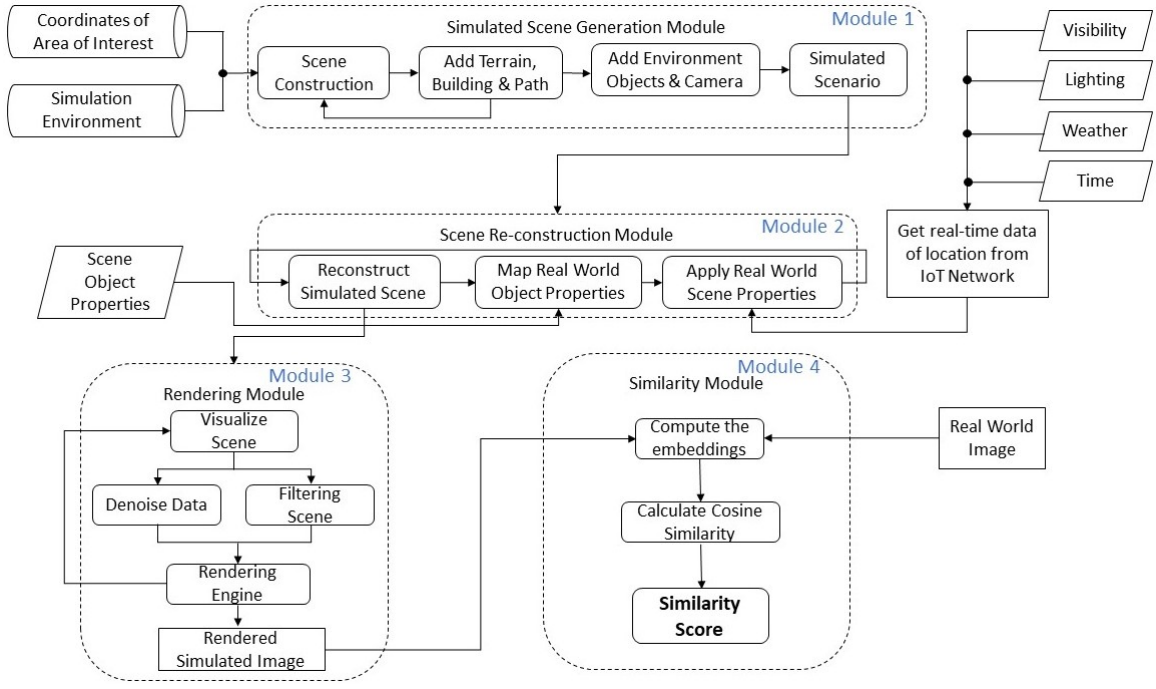


Fig. 14: System Architecture

Name	**Feature**
Blosm	First, the coordinates of the area of interest are taken, and using them, the 3D scene of that location is generated.
Object Real Snow	This feature helps in generating snow simulation on selected surfaces.

Table 4: List of Add-ons Used in the System

Algorithm 1 (System)

Input : Simulation Environment, Location and Prior Information about the Area of Interest

Output : Photorealistic Simulated Image of Scene with its Similarity Score

1. Input simulation environment and the scene objects created using coordinates of area of interest to Module 1 to initiate the simulation process
2. Input 3D scene from Module 1 to Module 2 and input scene properties and influential factors data collected from the IoT network into Module 2 for scene reconstruction process
3. Input reconstructed 3D scene data from Module 2 to Module 3 for rendering a synthetic image
4. Input simulated image from Module 3 and the real-world image counterpart to Module 4
5. Calculate the similarity score in Module 4 and interpret the gap between real-world and simulated images
6. End

The working and flow of the system are as per Algorithm 1. To initiate the process of creating synthetic images, the system creates a new Blender project for

the simulated environment. Moreover, in order to generate 3D scene of the desired location, coordinates of location are fetched from Openstreet Maps with the help of prior information. Here the prior information involves the information that helps in locating desired area on a 2D map such as Address, Landmarks, Street Names, Intersections, Reference Points and Compass Directions. By providing the discussed inputs, the process is initiated and the further tasks are then performed by each module in the system. In the Section 3.4, we discuss the workings of these modules.

3.4 Algorithms for Individual Modules

As discussed in the Section 3.3, the system comprises four modules designed to generate synthetic images and to assess their similarity to real images to gauge the reality gap. Each module is assigned specific tasks to achieve particular outcomes within its domain, collectively contributing to the creation of highly realistic synthetic images. To provide an in-depth understanding of the tasks carried out by each module, these modules are discussed next, outlining the algorithms used and presenting a test example for illustration.

3.4.1 Simulated Scene Generation Module

Algorithm 2 (Module 1)

Input : Simulation environment, location and prior information about the area of interest

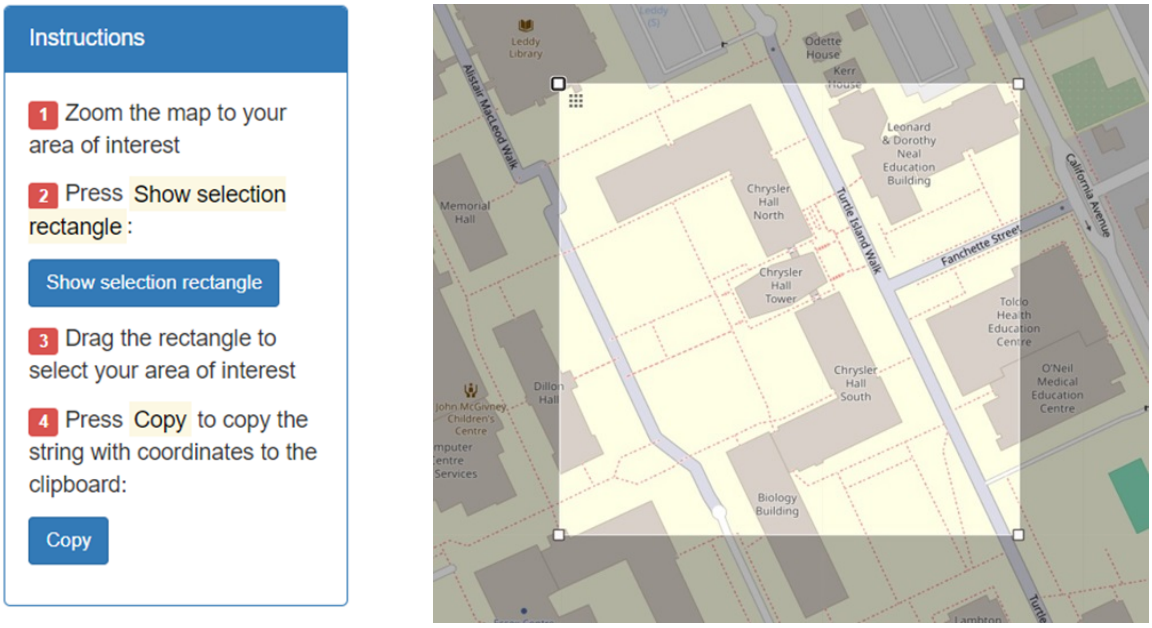
Output : 3D scene with terrain, building, road and paths

1. Using the prior information about the location, the area of interest (AoI) is fetched on a 2D map and its coordinates are obtained
2. Input AoI coordinates to simulation environment to create basic 3D scene
3. Get the terrain, buildings, roads and paths objects of the scene
4. Check the simulated scene and if there are missing objects or part of scene, go to step 1
5. Add environment objects like sun light, street lights and camera for setting up the basis for the simulated scene
6. End

The process begins with the prior information about location of interest as discussed in Section 3.3. Using the details like Address, Landmarks, Street Names, Intersections, Reference Points and Compass Direction, the coordinates of the location are fetched from a 2D map.

The obtained coordinates are then used as an input into the simulation environment software. This action defines the boundaries and positioning of the simulated space, establishing the area of the virtual scene to be created.

Using the AoI coordinates, the simulation environment generates a 3D scene representing the area of interest with basic shape and structure of terrain, buildings and raods/paths. The terrain is landscape or ground surface for 3D scenes, which is created with consideration of all the pits and hills from the real ground surface.



(a) Instruction to for Fetching Coordinates

(b) Map to Locate Area of Interest

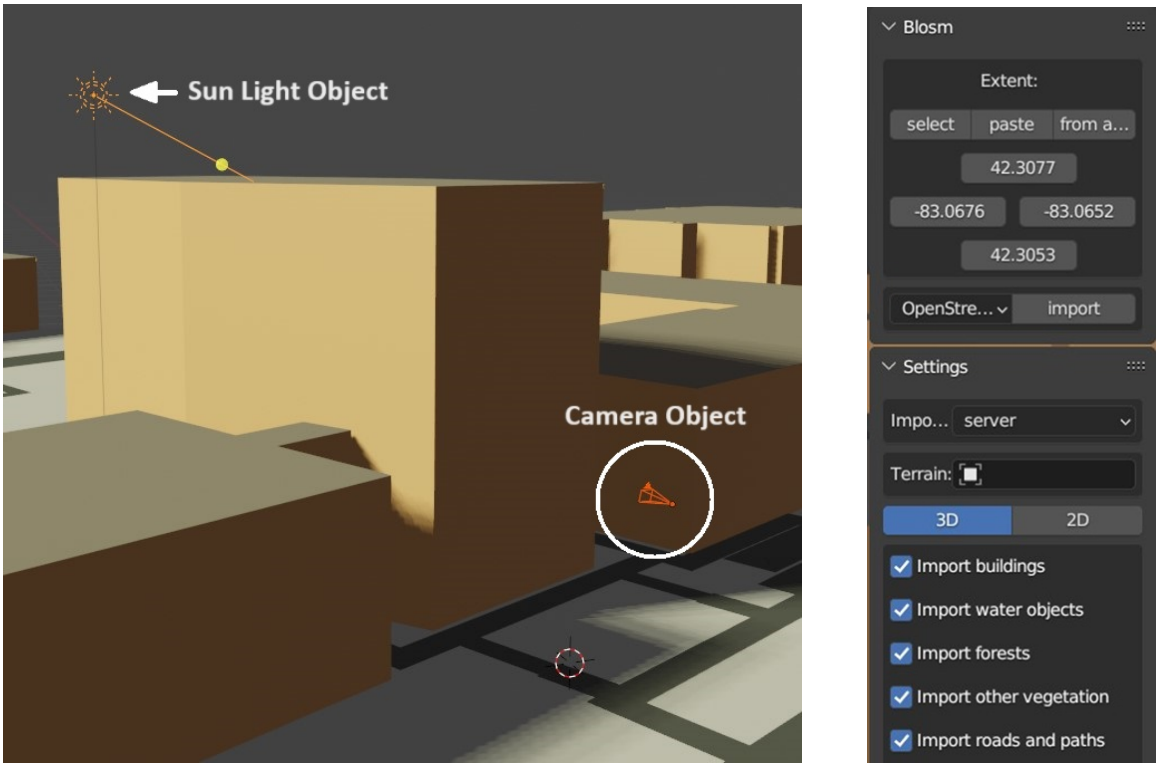
Fig. 15: Use of OpenStreetMap for Fetching Coordinates of Desired Location

Paths and Roads are important parts of the scene, and they are created by accurately adjusting width, length, curves, and turns. For buildings, Dimensions, Scale, and Primary Architectural Features are considered to ensure accuracy and realism. A 3D Map of the scene in the Figure 16 is generated here with no physical or material details.

After creating the basic 3D scene and retrieving scene objects, it is important to check for any missing components or inaccuracies. If any objects or parts of the scene are missing or incomplete, the process returns to step 1 in Algorithm 2 to refine the input data or to regenerate the scene.

Once the basic scene is constructed, additional environment objects are incorporated to enhance realism and functionality. These objects include lighting sources such as sunlight or street lights, as well as cameras for viewing and interacting with the simulated environment.

With the scene fully constructed and environment objects added, the process concludes. The Basic scene is now ready.



(a) Generated 3D Scene

(b) Blosm Menu in Blender

Fig. 16: 3D Scene Generation Using Blosm Add-On

To generate test result using the system, from the actual scene's location, the area is fetched in OpenStreetMap. From the OpenStreetMap, the coordinates of the AoI are extracted which is shown in Figure 15 and given as input in Blender as shown in Figure 16 (b). Finally by clicking on import in menu shown in Figure 16 (b), objects like terrain, roads, and buildings of the scene are generated as shown in Figure 16 (a). A 3D Map of the scene in the Figure 16 is generated with no physical or material details.

3.4.2 Scene Reconstruction Module

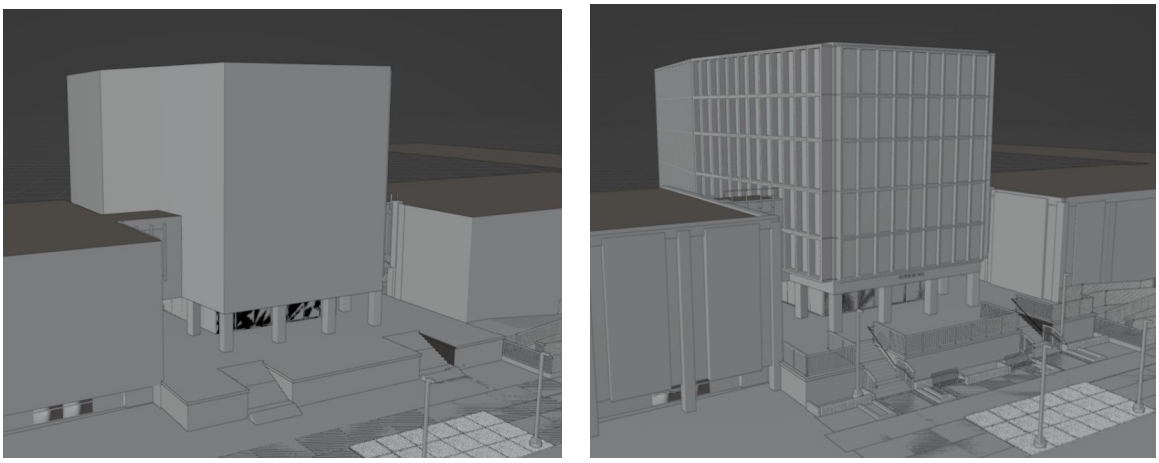
Algorithm 3 (Module 2)

Input : 3D Scene form Module 1, Influential Factors from IoT Network and Scene Object Properties

Output : Reconstructed 3D scene with accurately modeled objects, Scene and object properties, physical properties and carefully handled influential factors.

1. Input 3D scene from Module 1 and reconstruct the scene by enhancing details of architectural structure of the building objects and by adding missing objects in the scene
2. Input scene object properties like object textures and colors and map them to the 3D objects in the scene
3. Input influential factors data form Iot network and accurately apply scene environment properties like Time, Visibility, Weather and Time.
4. Check the 3D scene, and if there are missing key features, then repeat step 1, step 2 and step 3 for architectural structure, Object properties and scene environment properties respectively
5. End

The process in Algorithm 3 starts by taking the 3D scene generated in Module 1, which represents the basic layout of the area of interest. This scene is then reconstructed to enhance details of architectural structures, such as buildings. As illustrated in Figure 17, this reconstruction involves adding finer details to the building objects, refining their shapes, textures, and other architectural features. The information required in this refining task is captured from sources like Google Street View maps as well as from the real image data of the scene. For example as illustrated in Figure 16, the building object is missing the mesh window structure on its facades.



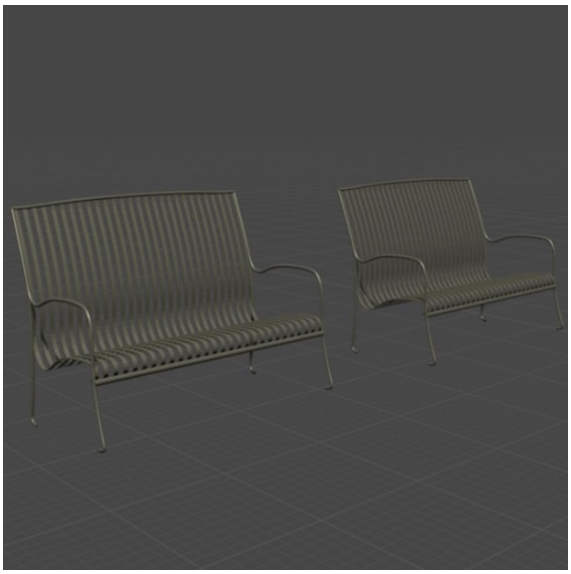
(a) Instance of 3D Scene Reconstruction Process

(b) 3D Scene after reconstruction

Fig. 17: Use of Scene Reconstruction Module for Refining details and Increase Realism

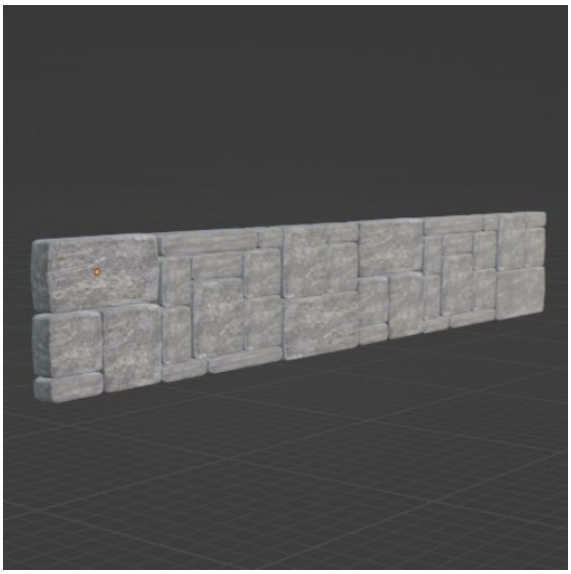
As visually this is an identifiable key feature of the building, this and the same kind of other key features are observed, noted and then modeled into the 3D scene which is shown in Figure 17. Additionally, any missing objects in the scene are identified and added to improve completeness and accuracy. The missing objects mainly are street lights, signs, etc. The information about these object for the test scene is also captured from Google Street View maps as well as from the real image data of the scene. Some examples of missing objects that are created for test result scene are shown in Figure 18.

Next, the properties of scene objects are input into the simulation environment. This includes specifying details such as textures, colors, material properties, and other visual attributes of the objects in the scene. These properties are extracted from real world object, ensuring that the virtual representation closely resembles its real world counterpart in terms of appearance and characteristics. It is very important to carefully extract and model the scene with object textures. One example is to make concrete walls, a rough surface texture is used on building walls. In the research done by Khairnar[16], Street View Static API was utilised for extracting textures of objects in the scene. Using the same approach object textures are extracted for the test results. The extracted textures are then mapped to the objects.

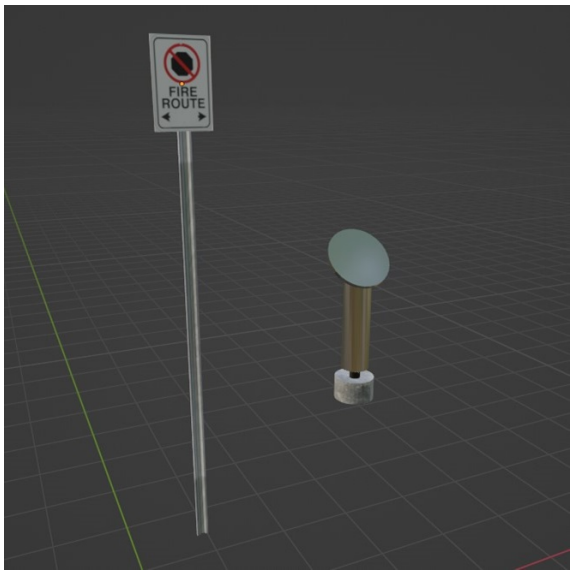


(a) Simulated Bench Objects

(b) Simulated street light Object



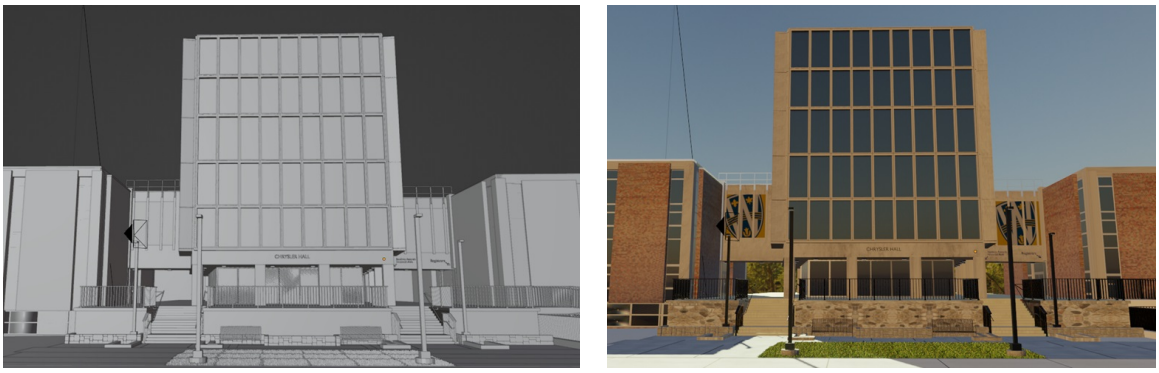
(c) Simulated Brick Wall Object



(d) Simulated Sign Objects

Fig. 18: Simulated Objects Designed and Created Additionally for the Scene

Furthermore, the data collected from IoT network, such as sensor readings or environmental conditions, is extracted. This data includes influential factors such as time of day, visibility conditions, weather conditions, and other environmental parameters. Accurately applying these influential factors to the scene environment adds realism and dynamic variability to the simulation. As discussed in Section 2.3, there are few main factors that control effects of other sub factors. These factors



(a) 3D Scene without Object and Environmental Properties

(b) 3D Scene with Object and Environmental Properties

Fig. 19: Effect of Mapping Objects Textures and Applying Influential Factors

are listed in the Table 5. Hence, to increase realism and to reduce reality gap, the properties listed in Table 5 can be extracted from the Environment Canada Website. The extracted data are then used in the test scene as the main influential factors.

Main Factor	**Sub Factor**
Time	Sun Position, Sun light Intensity
Weather	Rain, Snow, Background Sky, clouds
Lighting	Street Lights, Building Lights, Visibility, background sky
Visibility	Fog, Dust, Haze

Table 5: List of Main Factors and the Dependent Sub Factors

The Sun light is one of the most crucial factor of the real world as well as of a 3D scene. Setting the sun light source position in 3D scene is very important because it adds realism, believability, atmosphere, shadows and depth in the scene. Moreover, variation in the sun position and the sun light intensity help in adjusting time in the scene. Below is the formula that gives position of the sun in the form of coordinates using time.

$$\textbf{Solar Declination } (\delta)\textbf{:}\ \delta = 23.45^\circ \times \sin\left(\frac{365}{360} \times (d - 81\right)$$

$$\textbf{Hour Angle (H):}\ H = (t - 12) \times 15^\circ$$

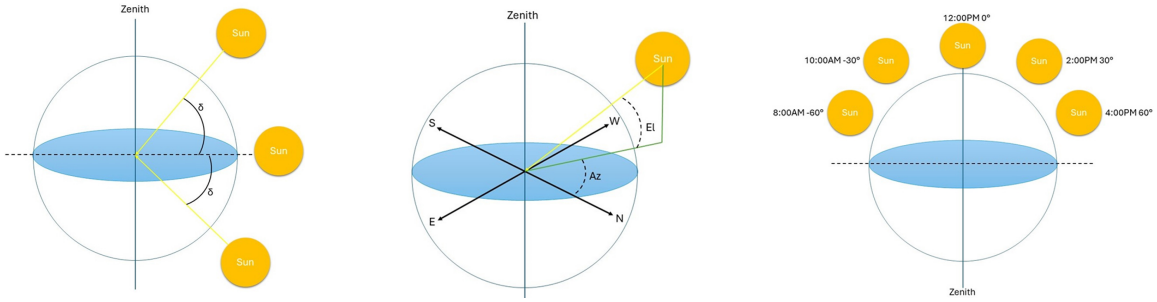
Azimuth (Az): $\text{Az} = \arctan 2\left(\sin(H), \cos(H) \cdot \sin(\phi) - \tan(\delta) \cdot \cos(\phi)\right)$

Elevation (El): $\text{El} = \arcsin\left(\sin(\phi) \times \sin(\delta) + \cos(\phi) \times \cos(\delta) \times \cos(H)\right)$

Conversion to Cartesian Coordinates (x,y,z):

$$x = \cos(\text{Az}) \times \cos(\text{El}), \quad y = \sin(\text{Az}) \times \cos(\text{El}), \quad z = \sin(\text{El})$$

Here, ϕ is the latitude of the location, d is the day of the year, t is the time of day in hours. As these factors are present in the real world and affect the way how scenes appear in reality, it is essential to take care of their effects in the simulated environment so that the generated simulated results can be close to their real-world counterparts. Figure 20 shows how the position of the earth changes with respect to the sun. Due to the variation in the position of sun and earth, calculation of the discussed angles become important in order to get the accurate position of the sun light in simulation environment.



(a) (b) (c)

Fig. 20: Figure (a) Represents Solar Declination, Figure (b) Represents Azimuth and Elevation, Figure (c) Represents Hour Angles

Through out the day intensity of sunlight also changes. Let I_{max} be the maximum intensity of sunlight (usually around noon) and t be the time, the following formula generates the intensity of sun light I(t).

Sun Light IntensityI(t): $I(t) = I_{\text{max}} \cdot \cos\left(\frac{\pi}{24} \cdot (t - 12)\right)$

Here, t is the time of day in hours, $\frac{\pi}{24}$ represents a 24-hour cycle, and (t-12) shifts the function so that it reaches its maximum at noon.

Another influential factor is the weather. The data extracted from the IoT network for the weather at a particular time is in the form of labels. The detailed list of possible weather and their label is available at Environment Canada Website [27]. Some examples of the data labels are below.

- Rain
- Snow
- Drizzle
- Hail
- Freezing Rain
- Freezing Drizzle
- Unknown Precipitation

When Unknown Precipitation weather or no obstructions to visibility occur, sky conditions are provided, reflecting the observation of the total cloud amount. The following terms are used based on the amount (in tenths) of cloud covering the dome of the sky.

- Clear (0 tenths)
- Mainly clear (1 to 4 tenths)
- Mostly cloudy (5 to 9 tenths)
- Cloudy (10 tenths)

For the test scene in this section, the clear weather is implemented with high visibility and without any effects like rain, snow, fog etc. To create weather conditions in the Blender simulator, a particle system for the weather is created to represent effects like snowfall, rainfall, etc. Moreover, in weather conditions like rain and snow, water and snow are simulated on the ground surfaces to increase the believability of

(a) Simulated Outdoor Scene in the Morning

(b) Simulated Outdoor Scene in the Night

Fig. 21: Test Results Generated Using the System

the weather effect. Finally, from the time and the weather data from the IoT network, the background sky is set in the simulated scene.

Giving simulated scenes lighting that is similar to the real world influences its realism. The scene's lighting not only decides the effect of the shadow and reflection but also sets the overall aesthetic. Lighting includes primary light sources like Sunlight or Moonlight and other sources like street lights, building ceiling lamps, etc. For example, to generate a scene as shown in Figure 21 (a), sunlight is adjusted to the following settings in Blender.

Sun Light	
Shade	FFFBB6
Strength	13
Angle	1.3d
Sun Location	
X	-69.5799 m
Y	-65.5795 m
Z	72.5111 m
Sun Rotation	
X	-50.69d
Y	-13.8412d
Z	124.941d

Table 6: Sun Object Settings in Blender for Figure 21 (a)

The settings in Table 6 give the scene night effect and change sunlight to moonlight by and setting light intensity to low levels and the shade of light. Moreover, streetlights and building ceiling lamps are turned on at night, which is also adjusted in the simulated scene. The sunlight for Figure 21 (b) is adjusted to the settings according to Table 7.

Visibility of the scene is a factor that controls the amount of clearly seeable details in the scene. When the weather is adjusted in the scene, the amount of visible information is reduced due to obstacles in the view from the atmosphere, such as

Sun Light	
Shade	FFF6EC
Strength	1
Angle	100d
Sun Location	
X	-149.857 m
Y	-51.9302 m
Z	40.4893 m
Sun Rotation	
X	-52.2676d
Y	56d
Z	154.991d

Table 7: Sun Object Settings in Blender for Figure 21 (b)

raindrops or falling snow. However, irrespective of seasonal factors, visibility levels sometimes drop due to the presence of fog, mist, dust, etc. To depict these effects, volumetric shaders in the Blender are employed, and the desired atmosphere condition is achieved by adjusting density and color.

The forms in which the data extracted from the network for the illustrated images in this section is in Table 8 and the factors simulated for each of them are also presented in the Table 8

After reconstructing the scene, mapping object properties, and applying influential factors data, the simulated 3D scene is thoroughly checked for any missing key features or inaccuracies. If any deficiencies are identified, the process iterates through steps 1, 2, and 3 accordingly, focusing on architectural structure, object properties, and scene environment properties, respectively. This iterative approach ensures that the simulated scene meets the desired level of completeness and fidelity. Figure 19 shows the scene before and after applying object and environment properties.

Influential Factors	**Form of Extracted Data**	**Adjustments in Simulator**
Time	YYYY/MM/DD, HH:MM	Sun Object Position, Sun Light Intensity, Other Lighting objects, Sky Background
Weather	Labels	Particle System, Sky Background, Ground Effects
Lighting	From Time of Day	Light Object Intensities, Sun Object Position, Sun/Moon Light Shade
Visibility	Value(km)	Particle System, Time, Lighting

Table 8: Influential Factor Adjustment in Simulator

Once the reconstructed scene is deemed satisfactory in terms of architectural detail, object properties, and scene environment properties, the Module 2 process concludes.

3.4.3 Rendering Module

Algorithm 4 (Module 3)

Input : Reconstructed 3D scene from Module 2

Output : Simulated image of 3D scene

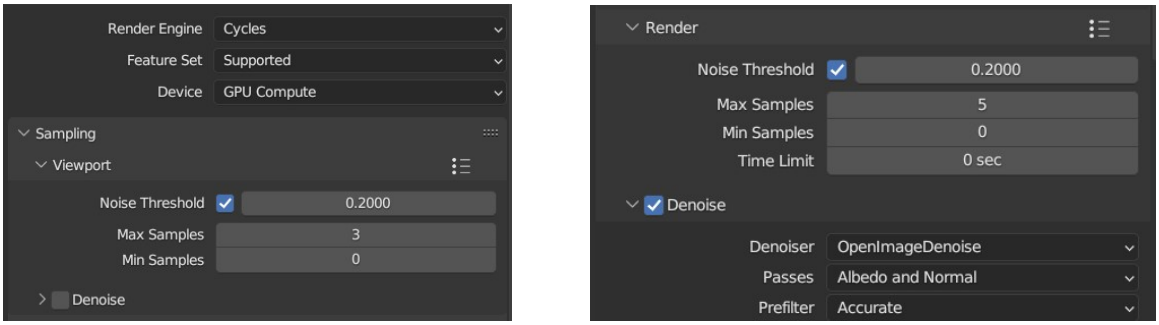
1. Input reconstructed 3D scene, set the camera position, adjust camera focal length and camera orientation according to the desired view of the image
2. Input parameters for desired denoising effect in the output image
3. Input filtering parameters for desired anti-aliasing effect in the output image
4. With the set parameters in steps 2 and 3, start the rendering process in the rendering engine
5. Analyze the results, and if the image does not have the desired view, go to step 1, if the image lacks quality, go to step 2.
6. End

The process of generating synthetic image begins by taking the reconstructed 3D scene from the previous steps, which includes all the enhanced architectural details, object properties, and environmental factors. This scene serves as the foundation for generating the final rendered image.

The camera position, focal length, and orientation are adjusted according to the desired view of the image. This step allows for specifying the viewpoint and framing of the scene, ensuring that the rendered image captures the desired perspective and composition.

Parameters for denoising, which reduce the presence of noise in the rendered image, are specified. For generating the test results in this section, denoising settings used

in Blender are shown in Figure 22. Understanding the GPU and CPU specifications of the computer system being used is essential. A computer system with a high-configuration GPU and CPU can perform better in high-quality rendering settings, while a computer system with a low-configuration GPU and CPU struggles to produce rendered images in the same settings. Denoising helps improve the image's visual quality by smoothing out irregularities and enhancing clarity, especially in areas with low light or complex textures. It is observed form the test results that the rendering engine typically takes between one to two minutes to produce a single image with the used system configuration.



(a) (b)

Fig. 22: Rendering Settings Used in Blender

Parameters for anti-aliasing, which reduce the appearance of jagged edges and smooth out pixelation in the rendered image, are defined. Anti-aliasing improves the overall visual quality of the image by mitigating the effects of aliasing artifacts, resulting in a more polished and realistic appearance.

With the camera parameters, denoising parameters, and anti-aliasing parameters set, the rendering process is initiated in the rendering engine. This process involves simulating the interaction of light rays with the scene geometry and materials to produce a photorealistic image of the virtual environment.

Once the rendering process is complete, the resulting image is analyzed to determine if it meets the desired criteria. If the image does not have the desired view, indicating that the camera parameters need adjustment, the process returns to step 2 to refine the camera settings and restart rendering. Similarly, if the image lacks

quality, suggesting that denoising or anti-aliasing parameters need adjustment, the process returns to step 3 to fine-tune these parameters and re-render the image.

The process of setting camera parameters, denoising parameters, and anti-aliasing parameters may involve iterative refinement until the desired view and quality of the rendered image are achieved. This iterative approach ensures that the final image meets the desired standards of accuracy, realism, and visual fidelity.

The scenes presented in Figure 21 are the results generated using the discussed system. Setting properties of variable objects like grass, trees, sun and clouds play vital role in giving scene weather and season conditions. Once the primary scene is ready, it becomes effortless to generate scene of different time by changing randomization properties like different weather conditions(Clouds, Leaf and Grass Color, Snow, Rain etc.), different times(Darkness, Sunlight, Sun Position) and variable objects like Grass, Snow, Fog, Sky etc.

3.4.4 Similarity Module

Algorithm 5 (Module 4)

Input : Simulated Image from Module 3, Real World Image

Output : Similarity Score of Simulated Image

1. Input the simulated image from Module 4 and the real world image to the module
2. Compute the embeddings of two images
3. Calculate cosine similarity between the two images
4. Display similarity score
5. End

In order to assess the efficacy of simulated images, it is crucial to compare them with a real-world counterpart. This comparison gives results in the form of a reality

gap, which represents the effectiveness of synthetic data. The process begins by obtaining the simulated image generated by Module 4, which represents the virtual scene created through the simulation process. Additionally, a real-world image of the same scene or environment is also provided. These images serve as the basis for comparison to evaluate the similarity between the virtual simulation and reality.

Both the simulated image and the real-world image are passed through an embedding model. An embedding is a vector representation of an image that captures its semantic features in a high-dimensional space. The embedding model transforms each image into a numerical vector representation, encoding its visual characteristics and content.

Once the embeddings of the simulated and real-world images are computed, the cosine similarity between the two vectors is calculated. Cosine similarity is a metric used to measure the similarity between two vectors in a multi-dimensional space. It calculates the cosine of the angle between the two vectors, with values ranging from -1 (completely dissimilar) to 1 (completely similar), where higher values indicate greater similarity. For testing results in this thesis, a model that utilizes Open AI CLIP by [32] is used. General formula for calculating the cosine similarity is as follows.

$$\textbf{Cosine Similarity:}\ \cos(\theta) = \frac{A\dot{B}}{\parallel A \parallel \parallel B \parallel} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

The computed cosine similarity score is displayed to provide a quantitative measure of how similar the simulated image is to the real-world image. This score indicates the degree of resemblance between the two images, with higher scores indicating greater similarity and lower scores indicating more divergence. The visual representation of this process is represented in Figure 23.

The process concludes after displaying the similarity score. This score serves as a valuable metric for assessing the accuracy and fidelity of the simulated image compared to reality. It provides insights into the effectiveness of the simulation process and can guide further refinements or adjustments to improve the realism of virtual environments.

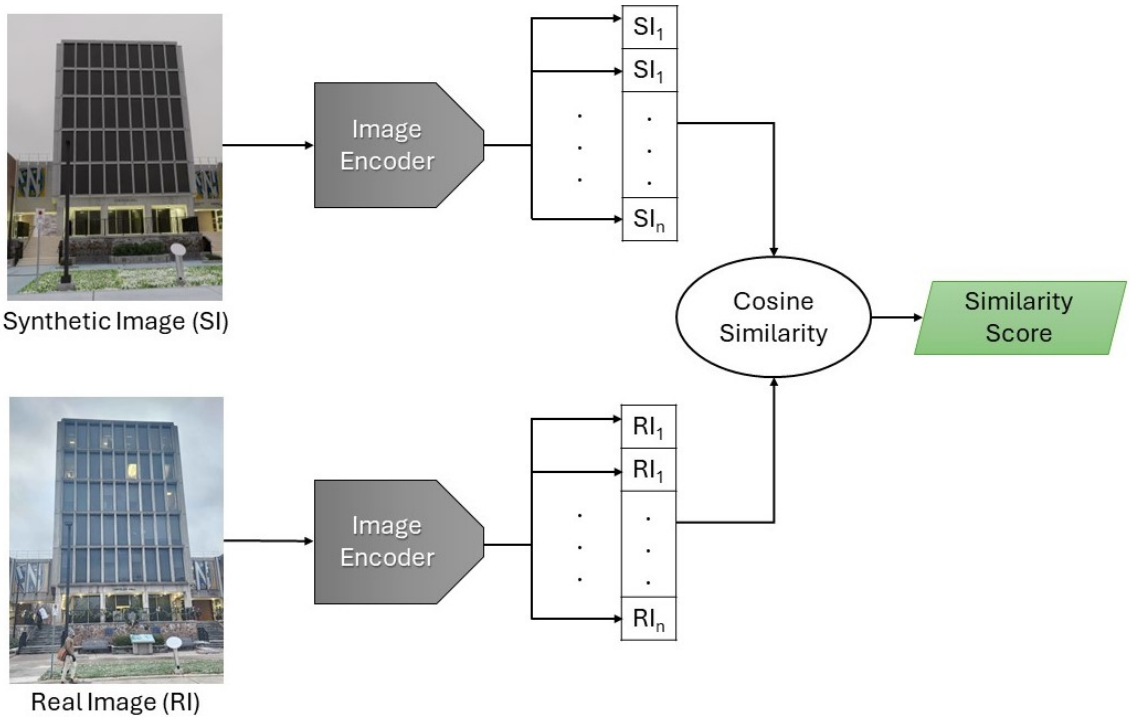


Fig. 23: Image Similarity Calculation Representation

When simulating indoor environment images, the better approach is to set up the simulation environment and then place objects in them. Then the use of object recognition algorithms for checking the reality gap provides better accurate estimation of similarity score. By training the model on the synthetic data and then using it for testing with real world counterparts provides accurate reality gap information. If the same approach from this study approach is for checking similarity, it results in higher similarity scores as replicating dimension and shapes of small objects in indoor simulation is not as difficult as outdoor scene objects. An example for this is shown in Figure 24.

Conversely, outdoor environments demand careful attention to structural information for accurate simulation. Inaccuracies in replicating architectural and structural details can exacerbate the reality gap in generated synthetic data. Therefore, outdoor scenes require meticulous efforts to ensure fidelity in simulation, as inaccuracies can significantly impact the quality of the synthetic data produced.

(a) (b)

Fig. 24: Indoor Environment Simulations and Its Real World Counterpart [39]

3.5 Time Complexity

Time complexities of all the algorithm are presented in Table 9. All the modules contributes in the time complexity of the system.

The system works in iterations for generating desired result, in the starting results may show low similarity. However, with iterative changes in reconstruction task, influential factors setting and object textures mapping the system improves and reaches the desired outcome showing higher similarity scores. Hence the time complexity of the system can be $O(n)$.

Algorithm	**Time Complexity**	**Details**
1	$O(n)$	n = number of iterations required to get desired results for a particular scene
2	$O(n)$	n = Number of iterations required to get desired scene and its objects
3	$O(l) + O(m) + O(n)$	l = Number of reconstruction iteration, m = Number of texture mapped, n = Number of adjustments made in simulator for setting influential factors
4	$O(n)$	n = Number of scenarios rendered
5	$O(n)$	n = Number of image embedding calculated

Table 9: Time Complexity of Algorithms

CHAPTER 4

Experiments and Analysis

This chapter includes a variety of experiments conducted to generate simulation images with matching or unmatched influential factors. Rather than generating a large-scale dataset of synthesized images for the evaluation of computer vision algorithms, the objective of the experiments is to assess the reality gap of synthesized images with an image similarity checking model. Therefore, all experiments focus at a fixed area of interest and investigate the difference between synthesized and real images while introducing controlled changes to the influential factors. After the explanation of implementation environment and software tools in Sec. 4.1, this chapter presents two phases of experiments that examine the impact on similarity scores when a single influential factor or multiple influential factors are different in Sec. 4.2 and Sec. 4.3 respectively. An analysis of the experiment results is finally presented in Sec. 4.4 with discussions.

4.1 Implementation and Tools

This section contain specifications of the tools that are used in the implementation of the research study. The implementation environment, followed by the development tools, are detailed below.

Implementation Environment

1. Operation system: 64-bit Windows 11, version 23H2
2. System type: x64 based processor
3. CPU: Intel Core i7-10750H CPU with 2.60GHz frequency
4. GPU: NVIDIA GeForce GTX 1650 5. RAM: 8.00 GB

Development Tools

1. Computer Graphics Software Tool: Blender v3.6
2. Programming Language: Python 3.10
3. Libraries: PIL, transformers, Numpy

4.2 Experiments and Results Phase 1

The system is implemented by creating a repository of object models with various materials and environment scenes with different lighting conditions. As presented in Figure 21, Chrysler Hall building at the University of Windsor is selected as the test scenario location. For each simulation scenario, the experiment uses Blender simulator to generate first a photorealistic image according to the physical properties of the real-world environment and then a set of synthetic images with variations being introduced to the value of influential factors. The varied simulation scenarios are compared with the real-world images to assess the reality gap. This section presents the initial set of experiments, which considers the main building object under the influence of object material and environment scene lighting conditions.

4.2.1 Sunny Afternoon in Summer

To generate a sunny afternoon scenario in summer, a real-world image of the scene was captured on 2nd October, 2023 at 5:02 PM. Using Blosm add-on in Blender 3.6, a basic 3D location scene with the right measurements of buildings, terrain, and roads is created. To make the scene photorealistic, the scene reconstruction module is used and all the factors were fine-tuned. It is crucial to model objects in this module to achieve the structure that they have in real world, hence, objects are modeled according to their structures. Furthermore, the data collected from Google Street View API is used to give the objects their appropriate material. At this stage, the 3D scene is ready with all the objects that are both realistic and have the same structure and shapes as their real-world counterparts.

(a) (b)

Fig. 25: Test Result 1: Generated Using the System where (a) is a Real World Image and (b) is a Simulated Result

Finally, the last step is to accurately set influential factors. The data gathered from Environment Canada Website for the same day and time is utilised to set the influential factors like sky, weather, time, and sunlight. Figure 25 represents the real scene and the generated simulated image result. The information about influential factors and the achieved similarity score for the scene is added in Table 10.

Similarity score	87.98
Influential Factors	
Weather	Sunny
Time	5:02 PM
Visibility	16.1KM
Lighting	20

Table 10: Results and Influential Factors Information for Sunny Afternoon in Summer Experiment

4.2.2 Cloudy Afternoon in Early Winter

(a) (b)

Fig. 26: Test Result 2: Generated Using the System where (a) is a Real World Image and (b) is a Simulated Result

To begin with, a picture of the location was captured on 27th November, 2023 at 4:36 PM from a different angle in winter season.

To generate the simulated result, the steps mentioned in Section 4.2.1 are followed until the 3D scene with realistic objects is ready, the influential factors are carefully handled. As seen in Figure 26 (a), The real scene in the winter has snow on the ground. From the Environment Canada Website, the data is extracted where cloudy sky and low visibility are the important elements.

As listed in the Table 4, the Object Real Snow add-on is used in Blender 3.6, using which snow on the ground is generated. To imitate cloudy sky background, sunlight is set to a low setting and a cloudy weather HDRI image is added in as scene world background. For this simulaed result, the visibility is not adjusted to a lower setting. In the end, after setting the camera position, the result synthetic image is rendered, which is shown in Figure 26 (b). The effect of visibility factor is reflected in the similarity score, which is added in Table 11.

Similarity score	78.4
Influential Factors	
Weather	Snow
Time	4:36 PM
Visibility	16.1KM
Lighting	0.2

Table 11: Results and Influential Factors Information for Cloudy Afternoon in Early Winter Experiment

4.2.3 Evening in Summer

To begin with, a picture of the location was captured on 18$^{\text{th}}$ September, 2023 at 7:47 PM from a different angle in the evening time.

To generate the simulated result for an evening in summer, the steps mentioned in Section 4.2.1 are followed until the 3D scene with realistic objects is ready, the influential factors are then carefully handled. As shown in Figure 26(a), The real scene is in the evening, and hence, the lighting is very low. Moreover, the streetlights

(a) (b)

Fig. 27: Test Result 3: Generated Using the System where (a) is a Real World Image and (b) is a Simulated Result

Similarity score	80.06
Influential Factors	
Weather	Clear
Time	7:47 PM
Visibility	16.1KM
Lighting	8

Table 12: Results and Influential Factors Information for Evening in Summer Experiment

and building ceiling lamps are lit up. From the Environment Canada Website, the data is extracted where clear sky and good visibility are the important factors.

As the real scene has a clear sky and low light, the sunlight is replaced with the moonlight and it is adjusted to a low-intensity setting. Although the real scene has

a clear sky that is less dark, a dark night sky with clouds HDRI image is used in the simulated scene to showcase its influence on the similarity score. In the end, by setting the appropriate camera position, the result is rendered which is shown in Figure 27. The information of influential factors and the achieved similarity score for the scene is added in Table 12. Here, it is clear that the sky as background plays a vital role in comparison, and hence, the lower similarity score is resulted.

4.2.4 Analysis

When simulating real world object, there are certain characteristics of objects which are influential factors that has to be completely accurate in order to incorporate other factors and make scene look real. From the experiments, it is observed that these are the influential that can be added in the category of default influential factors which has major effects on reality gap. The default influential factors observed from the experiments are:

- **Dimensions:** Proper dimensions replicate real-world scale, ensuring objects and environments are realistically sized. Accurate dimensions are crucial for simulations involving physics, engineering, or architectural visualization to function correctly.
- **Shapes:** Proper shapes ensure that objects behave realistically in simulations involving physics or interactions. Correct shapes contribute to visual coherence, creating a more cohesive and immersive virtual environment.

For the experiments in this study, dimensions are set accurately using open source add on feature called Blosm.

4.3 Experiments and Results Phase 2

From the results generated in the previous section using the system, it is observed that multiple factors affect the similarity score, and if any one or more than one of them

is not set accurately, the similarity gets affected significantly, and hence, the reality gap increases. These are the influential factors that need to be handled precisely in order to achieve a higher similarity score. To achieve better results and to gain more insights into how each factor affects similarity score, another phase of experiments is performed and various results are gathered.

In this experiment phase, the real-world scene is selected from Section 4.2.1 and various simulated images are generated by varying single or multiple influential factors. In this experiment, a total of 20 results are generated and presented for analysis.

4.3.1 Single Variations

Fig. 28: Real World Image Used in the Experiments for the Comparison

To check the impact of each factor on the reality gap, the results are generated by disabling one factor at a time and then compared with Figure 28. Here, disabling the factor means setting the value of that factor incorrectly or not setting it at all to give an adverse effect compared to what is present in the real-world scene.

Table 13 below shows the information of each factor with its set and unset influential factor.

To begin this experiment phase, the 3D simulation scene is selected from Section 4.2.1 in Blender, for which a simulated image is generated that is shown in Figure 28.

	Objects					**Weather**				**Camera**		
No.	**Shape**	**Material**	**Visibility**	**Time**	**Season**	**Sun**	**Rain**	**Snow**	**Background**	**Position**	**Orientation**	**Similarity**
1	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	80.43
2	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	78.95
3	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	66.27
4	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	73.65
5	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	74.02
6	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	77.50
7	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	82.52
8	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	73.54
9	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	76.88
10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	79.03
11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	77.78

Table 13: Simulated Results with Single Factor Variations

Initially, all the influential factors are already set for generating results in Figure 28, hence, first the shape of the building is changed by reducing the bounding edges of Chrysler Hall building. By variating the edges, minor changes in the building's shape are introduced, and the simulated image converted to a Rectangular building structure, which is different from its real-world structure. The real image as Figure 28 and the simulated image as Figure 29 are then used in the similarity module. After the comparison, the similarity score of 80.43% is achieved which is lower than the score achieved of 87.98 in Section 4.2.1.

The main observation here is that the shape factor contributes in increasing the realism of the simulated image, and here, as the change is minutiae, its effect on the

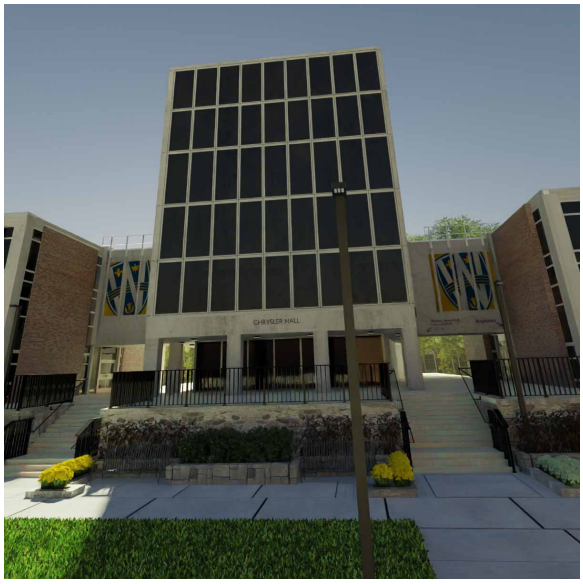


Fig. 29: Simulation Scene with Unset Shape Factor

similarity is also small and is about 7.55%.

For the second result in Table 13, the previous factor is reset to what it is in Figure 28, and for generating a new simulated image, the materials of the scene are disabled and single color textures are added to the scene. Therefore, the scene is kept with the right shades of color but no material information. The real image as Figure 28 and the simulated image as Figure 30 are then used in the similarity module. After the comparison, the similarity score of 78.95% is achieved.

The main observation here is that all the materials contribute to realism of the synthetic image. Here, as the materials are disabled and the objects are kept only with the same shades of colors, there is a reduction of 9.03% in similarity score.

For the third result in Table 13, all the materials are remapped and for generating a new simulated image, the visibility of the scene is set to low levels, which is the opposite of what is present in real world scene. The real image as Figure 28 and the simulated image as Figure 31 are then used in the similarity module. After the comparison, the similarity score of 66.27% is achieved which is the lowest score achieved in Table 13.

The observation here is that visibility is the factor that controls the amount of information visible in the scene. Therefore, if the real scene has higher visibility and

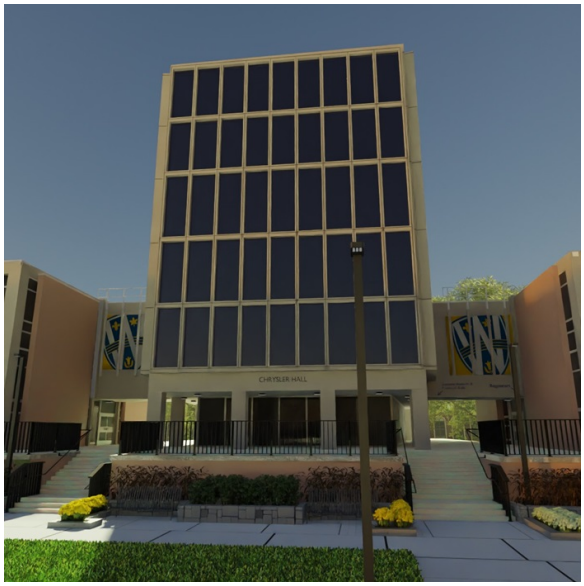


Fig. 30: Simulation Scene with Unset Materials of Objects

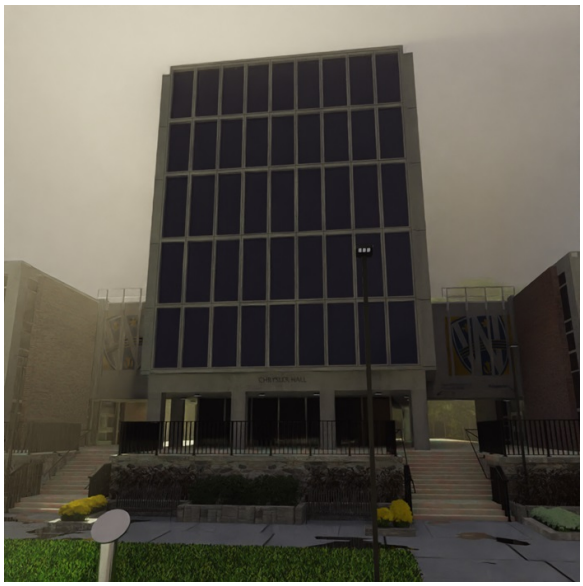


Fig. 31: Simulation Scene with Unset Visibility Factor

a clear view, setting it too low in the simulated scene makes the image look dissimilar as the synthetic images would not include as many details in the scene. Due to the lowered visibility, there is a reduction of 21.71% in the similarity score.

For the fourth result in Table 13, the visibility is reverted to what it is in Figure 28, and for generating a new simulated image, the time of scene is changed setting by setting the sunlight intensity to 0.2 in the blender. The real image as Figure 28

and the simulated image as Figure 32 are then used in the similarity module. After the comparison, the similarity score of 73.65% is achieved which shows a clear effect of reduction in realism.

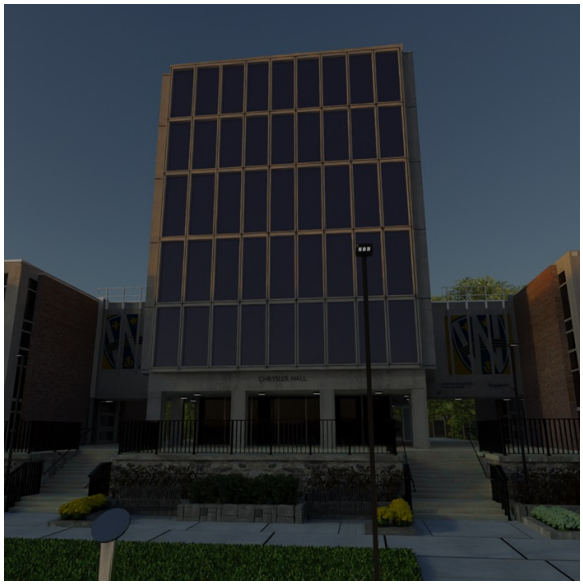


Fig. 32: Simulation Scene with Incorrectly Set Sunlight

The observation here is that time, in terms of sunlight intensity, is the factor that controls how clear and bright the objects are visible in the scene. If the intensity of light is too low or too bright in comparison to the real world, it negatively affects the similarity score, so it needs to be handled with good accuracy. Hence, when the time is disabled, a reduction of 14.33% in similarity score is resulted.

For the fifth result in Table 13, the time is changed to what it is in Figure 28, and for generating a new simulated image, the season of the scene is changed by adding rain drop filter, puddles on the ground and cloudy sky as the background. The real image as Figure 28 and the simulated image as Figure 33 are then used in the similarity module. After the comparison, the similarity score of 74.02% is achieved which shows the clear effect of reduction of realism in Table 13.

In this result, the added features to introduce the rainy season visibly make the real and the synthetic images look different. Here, as the season is disabled, there is reduction of 13.96% in similarity score.

For the sixth result in Table 13, the season is changed to sunny again, and for

Fig. 33: Simulation Scene with Changed Season Features

generating a new simulated image, position of the sun in the scene is changed. The real image as Figure 28 and the simulated image as Figure 34 are then used in the similarity module. After the comparison, the similarity score of 77.50% is achieved which shows the clear effect of reduction in realism.

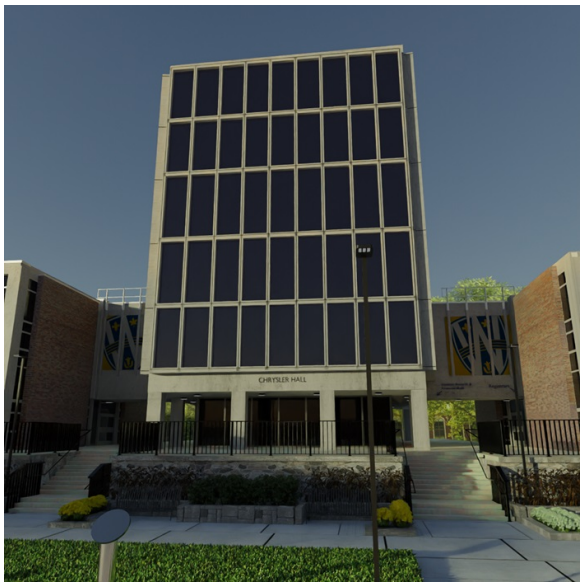


Fig. 34: Simulation Scene with Incorrect Sun Position

The observation here is that when the sun is set to a position that is not the same as the real-world image, it changes various factors in the scene. Taking about

this scene specifically, it is clear that the shadow projection and its directions are different from the real scene. Moreover, in the scene, the reflections on the glasses are also observed to be different. Here, as the sun is disabled by not setting to the right position, there is a reduction of 10.48% in the similarity score.

For the seventh result in Table 13, the sun is reverted to its original position, and to generate a new simulated image, rain is added to the scene. The real image as Figure 28 and the simulated image as Figure 35 are then used in the similarity module. After the comparison, the similarity score of 82.52% is resulted, which shows the clear effect of reduction in realism.

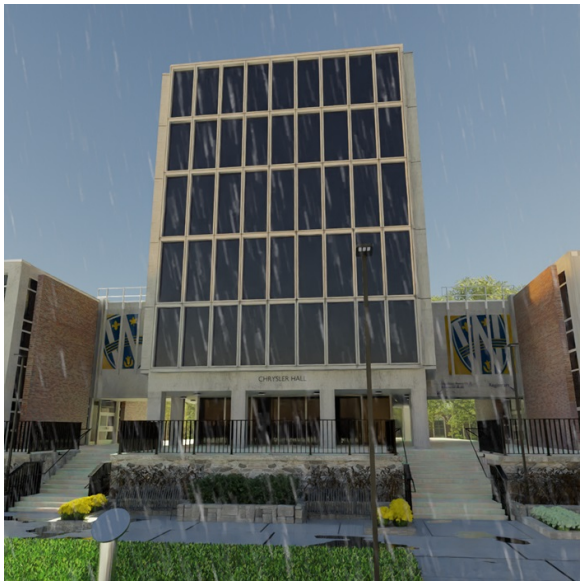


Fig. 35: Simulation Scene with Added Rain Features

The observation here is that as only the raindrops and the puddles are added but not the cloudy background, the image's overall appearance stayed the same with only raindrops and puddles in the scene. Therefore, the rain here is disabled by adding it into the scene where it is not present in the real world, there is a reduction of 5.46% in similarity score.

For result number eight in Table 13, the rain is removed, and to generate a new synthetic image, the snow is added to the scene. The real image as Figure 28 and the simulated image as Figure 36 are then used in the similarity module. After the comparison, the similarity score of 73.54% is achieved, which shows the clear effect

of reduction in realism.

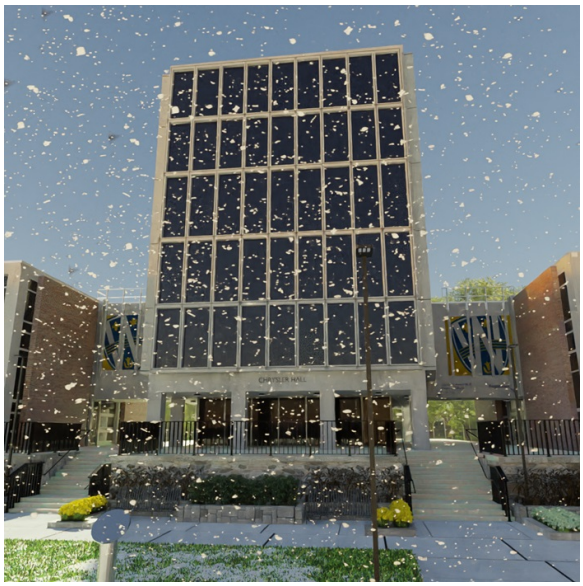


Fig. 36: Simulation Scene with Snow Features

It is observed here that as only falling snow particles and snow on the ground are added but not the cloudy background, the effects are seen the same as the last image with only rain Figure 35. Therefore, the snow here is disabled by being added into a scene where it is actually not present, there is a reduction of 14.44% in similarity score.

For the ninth result in Table 13, the snow features are removed, and to generate a new simulated image, the sky background of the scene is changed. The real image in the similarity module as Figure 28 and the simulated image as Figure 37 are then used. After the comparison, the similarity score of 76.88% is achieved, which also shows the effect of reduction in realism.

In this result, it is visible that with the change in the sky, the background image appears to be either of a different season or of a different time. However, the main object and other objects appear the same with all the other settings correctly handled. Here, as the background is disabled, the reduction of 11.4% is seen in the similarity score.

For the tenth result in Table 13, the background is changed to what is in Figure 28, and for generating a new simulated image, the camera orientation is changed in

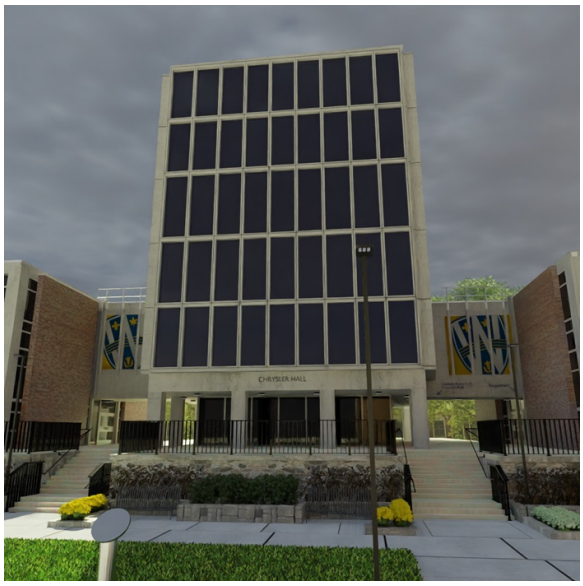


Fig. 37: Simulation Scene with Different Sky in The Background

the scene. The real image as Figure 28 and the simulated image as Figure 38 are then used in the similarity module. After the comparison, the similarity score of 79.03% is achieved, which is surprisingly low.

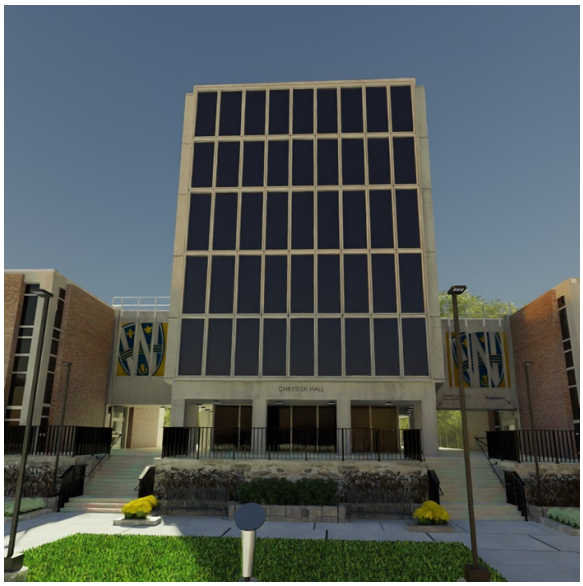


Fig. 38: Simulation Scene with Changed Camera Position

Here, the synthetic image has objects which are not presented or seen in the actual image. According to this observation, there is an imbalance of information or details between both images, where one has a smaller portion of an object than the other.

As the actual image is in portrait orientation, it covers more ground and sky view compared to the synthetic image which is in landscape orientation. Therefore, both of the images contain different information of extra details. As a result, there is a reduction of 8.95% in similarity score.

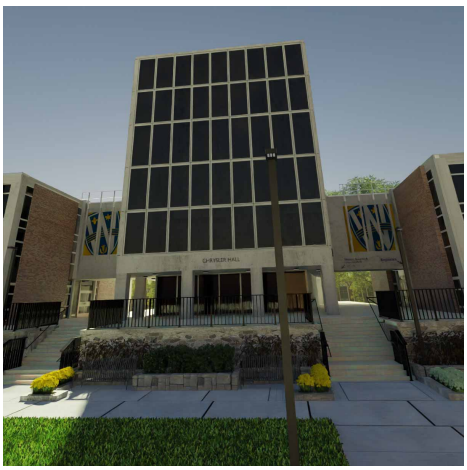
Fig. 39: Simulated Image with Changed Camera Orientation

For the result number eleventh in Table 13, the orientation is changed back to portrait, and to generate a new simulated image, the camera's position is changed, capturing the image's different view. The actual image as Figure 28 and the simulated image as Figure 39 are then used in the similarity module. After the comparison, the similarity score of 77.78% is achieved, which shows the effect of reduction in realism.

Here, we still wanted the scene to be nearly the same, hence, a slight position change is introduced to the camera object. As the view changed, the image visibly looks different and a reduction of 10.20% is seen in similarity score.

4.3.2 Multi-Factor Variations

Until now, all the experiments are performed to get insight into the effect of each individual factor on image similarity. As shown in Table 14, the cases when the number of disabled factors increases, the similarity score reduces gradually until reaching the lowest score of 59.78% at the end of Case 8.



(a) Synthetic Image of Case Number 1 in Table 14

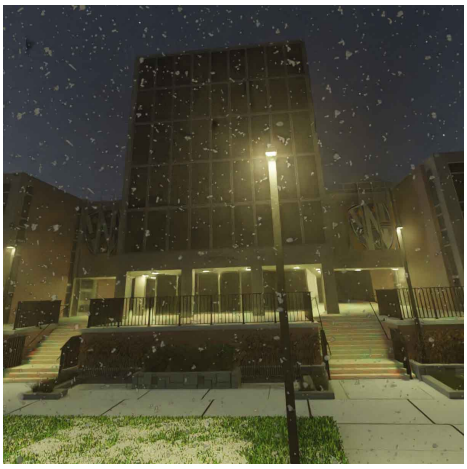
(b) Synthetic Image of Case Number 2 in Table 14

Fig. 40: Synthetic Images Used in Case 1 & 2 with Image in Figure 44 (a)

(a) Synthetic Image of Case Number 3 in Table 14

(b) Synthetic Image of Case Number 4 in Table 14

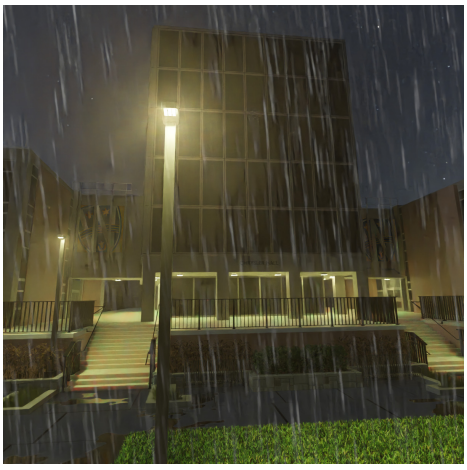
Fig. 41: Synthetic Images Used in Case 3 & 4 with Image in Figure 44 (a)



(a) Synthetic Image of Case Number 5 in Table 14

(b) Synthetic Image of Case Number 6 in Table 14

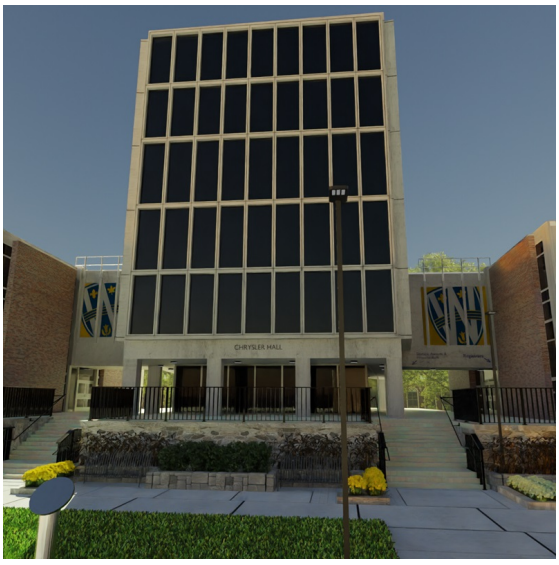
Fig. 42: Synthetic Images Used in Case 5 & 6 with Image in Figure 44 (a)



(a) Synthetic Image of Case Number 7 in Table 14

(b) Synthetic Image of Case Number 8 in Table 14

Fig. 43: Synthetic Images Used in Case 7 & 8 with Image in Figure 44 (a)



(a) Synthetic Image of Case Number 9 in Table 14

(b) Real World Image Used in all of the Experiments in Table 14

Fig. 44: Synthetic Image Used in Case 9 and the Real Image

	Objects					Weather				Camera		
No.	**Shape**	**Material**	**Visibility**	**Time**	**Season**	**Sun**	**Rain**	**Snow**	**Background**	**Position**	**Orientation**	**Similarity**
1	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	80.43
2	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	77.28
3	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	66.66
4	✗	✗	✗	✗	✓	✗	✓	✓	✗	✓	✓	75.19
5	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓	73.50
6	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓	63.56
7	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	63.35
8	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	59.78
9	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	87.98

Table 14: Simulated Results with Multiple Factor Variations

No.	Influential Factor	Enabled (✓)	Disabled (✗)
1	Shape	Building edges(Facade mashes) are correctly adjusted	Building edges are reduced to almost zero
2	Material	Objects are set with correct materials	Objects are colored without material information
3	Visibility	A fog object is added having volumetric shader with density of -0.84	The fog object is removed
4	Time	Sun light intensity: 20	Sun light intensity: 0.2
5	Season	Sunny day HDRI background, Accurate sun light	Inaccurate sun light, Rain particle system, Cloudy HDRI background, Puddles on the ground using noise texture
6	Sun	Sun Position: [X=-69.5799m, Y=-65.5795m, Z=72.5111m], Sun Rotation: [X=-50.69d, Y=-13.8412d, Z=124.941d]	Sun Position: [X=108.199m, Y=-30.6449m, Z=72.5111m], Sun Rotation: [X=-36.7609d, Y=-79.2629d, Z=194.5d]
7	Rain	No settings as the real scene has a sunny day	Rain particle system, Water using noise texture on the ground
8	Snow	No settings as the real scene has a sunny day	Snow particle system, Snow simulation on the ground using Object Real Snow add-on
9	Background	Sunny day HDRI background	Cloudy afternoon HDRI background
10	Camera Position	Camera Position: [X=4.26163m, Y=21.0894m, Z=3.84031m], Camera Rotation: [X=181.264d, Y=-76.3983d, Z=30.2728d]	Camera Position: [X=12.3236m, Y=18.0101m, Z=3.97194m], Camera Rotation: [X=98.9491d, Y=-0.531919d, Z=111.117d]
11	Camera Orientation	Resolution: [X=1332, Y=1000]	Resolution: [X=1000, Y=1332]

Table 15: Settings of Influential Factors in Blender for the Experiments

The enabled and disabled settings used in Blender for each influential factors are added in the Table 15. It was observed in this experiment phase that if multiple influential factors are disabled, realism of image affects significantly and thus reality gap increases. Therefore, in order to reduce the reality gap and increase realism of synthetic image, it is very important to handle each factor and all factors accurately. We discuss more about the above results and their analysis in Section 4.4.

4.4 Analysis and Discussions

According to results obtained from the two phases of experiments, it becomes evident that the accurate placement of the sun, sun light intensity according to the time, weather features, and visibility in synthetic scene generation significantly impact the realism and fidelity of the resulting images. During the experiments, altering influential factors results in a notable reduction in the similarity score between the synthetic and real images. Specifically, disabling the factor of sun position led to a 10.48% decrease, setting sun light not according to the time led to a 14.33% decrease, changing season led to a 13.96% decrease in similarity. The results highlight the importance of environmental parameters in scene generation. Moreover, it was noticed that visibility in the simulated scene is also one of the important influencing factors as the result generated by disabling it showed the highest similarity drop of 21.71%.

The observed differences in shadow projection and reflections on surfaces underscore the intricate interplay between lighting conditions and visual perception. Accurate modeling of visibility, including factors such as fog, haze, and occlusion, is crucial for capturing the nuances of depth and spatial relationships within a scene. Similarly, incorporating realistic weather effects such as rain, snow, or mist can greatly enhance the atmospheric authenticity of synthetic imagery, contributing to its overall realism. Furthermore, accounting for variations in time of day and dynamic lighting changes is essential for simulating naturalistic lighting conditions and accurately portraying the temporal progression of scenes over the course of a day. These considerations highlight the critical role of comprehensive environmental modeling in ensuring the

fidelity and believability of synthetic imagery across a diverse range of scenarios and conditions.

The work done by Khairnar [16] showcased automated approach of constructing 3D city models where less emphasis was given to refining objects. This study draws inspiration as well as extends work done by Sonetta [39] for generating simulated outdoor environment scene. By prioritizing the reconstruction of objects, detailed 3D models are created. Ultimately, through the meticulous adjustment of real-world properties and careful management of influential factors, the system successfully generated scenes and synthetic image data that exhibited enhanced realism.Therefore, employing our system for generating synthetic datasets to train computer vision algorithms presents a promising alternative to address the limitations of real-world image datasets' availability.

Finally, scalability and computational efficiency could be the major challenges for the system, particularly when aiming for high-resolution, photorealistic outputs. Balancing computational resources with the demand for realistic imagery presents a significant hurdle in achieving widespread adoption of synthetic data in real-world applications. Developing efficient algorithms and optimization techniques to streamline the generation process while maintaining quality standards is an area ripe for further exploration in future research.

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

In this thesis we studied the area of generating synthetic image data and the problem of reality gap. We developed a system that is capable of generating large amount synthetic image data using simulation. The data generated using the system are carefully modeled, reconstructed and has carefully handled influential factors which increases the realism of synthetic data. According to the observations from previous research done by Khairnar[16], the system in this thesis incorporates a module focusing on the reconstruction of the scene. Meticulous modeling of architectural details in 3D simulated scenes is paramount for enhancing realism and fostering immersive experiences. By faithfully replicating almost every aspect of real world structures, from textures to dimensions, quality of scene increased and helped getting higher similarity scores. Moreover, incorporating IoT technology in the system to gather real time weather data and integrating it into 3D scene simulations help dynamically adjust virtual environments to mirror real world conditions. Finally, the similarity module that utilises OpenAI CLIP is used for the task of image comparison, demonstrates remarkable accuracy in discerning visual similarities. Its versatility extends beyond traditional image recognition methods, offering robust performance across a wide range of visual tasks.

Through rigorous testing and by performing a number of experiments we showed the model's effectiveness and data quality. From the gained insights from testing we generated results and compared them with its real world counterparts by which

a similarity score of 87.98% is achieved. The achieved similarity suggests that the simulated scene effectively captures essential elements of the real environment, including geometry, textures, lighting, and overall ambiance. As the scene 3D scene are tested to be effective, by varying influential factors within the simulated scene, such as lighting conditions, object placements, camera angles, and environmental parameters, developers can generate a diverse range of data instances. This variability introduces richness and complexity into the training dataset, allowing Computer Vision and AI models to learn from a broad spectrum of scenarios and adapt more effectively to different conditions encountered in the real world. Moreover, the ability to control and manipulate these factors enables targeted data augmentation strategies, enhancing the robustness and generalization capabilities of the trained models.

5.2 Future Work

This work can be extended by exploring methods to automate the generation process. In addition, it will be beneficial to improve efficiency, for scaling up data generation to even larger amounts and for generating very large sized 3D models such as cities. Another area worth investigating is the development of techniques to dynamically adapt the synthetic data generation process based on feedback from the performance of computer vision algorithms, thereby continuously improving the realism and effectiveness of the generated data. Furthermore, exploring ways to generalize the model across different domains or tasks could expand its applicability and impact. Overall, continued research and development in this area hold the potential to further bridge the reality gap and facilitate the advancement of computer vision algorithms in various real-world applications.

REFERENCES

[1] Anderson, J. W., Ziolkowski, M., Kennedy, K., and Apon, A. W. (2022). Synthetic image data for deep learning. *arXiv preprint arXiv:2212.06232*.

[2] Borkman, S., Crespi, A., Dhakad, S., Ganguly, S., Hogins, J., Jhang, Y.-C., Kamalzadeh, M., Li, B., Leal, S., Parisi, P., et al. (2021). Unity perception: Generate synthetic data for computer vision. *arXiv preprint arXiv:2107.04259*.

[3] Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., and Fox, D. (2019). Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE.

[4] Daza, I. G., Izquierdo, R., Martínez, L. M., Benderius, O., and Llorca, D. F. (2023). Sim-to-real transfer and reality gap modeling in model predictive control for autonomous driving. *Applied Intelligence*, 53(10):12719–12735.

[5] Debs, P. and Fayad, L. M. (2023). The promise and limitations of artificial intelligence in musculoskeletal imaging. *Frontiers in Radiology*, 3:1242902.

[6] Farm, M. (2023). Introduction to iot (internet of things)-working, advantage. `https://shorturl.at/hlnx6`. Accessed: April 25, 2024.

[7] Foundatio, B. (2024). Blender - a 3D modelling and rendering package. `https://www.blender.org/`. Accessed: April 6, 2024.

[8] Ghazal, T. M., Hasan, M. K., Alshurideh, M. T., Alzoubi, H. M., Ahmad, M.,

Akbar, S. S., Al Kurdi, B., and Akour, I. A. (2021). IoT for smart cities: Machine learning approaches in smart healthcare—a review. *Future Internet*, 13(8):218.

[9] GmbH, M. C. (2024). Cinema 4D: 3D modeling, animation, and rendering software. `https://www.maxon.net/en/products/cinema-4d/overview/`. Accessed: April 7, 2024.

[10] Hinterstoisser, S., Pauly, O., Heibel, H., Martina, M., and Bokeloh, M. (2019). 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*.

[11] Hodaň, T., Vineet, V., Gal, R., Shalev, E., Hanzelka, J., Connell, T., Urbina, P., Sinha, S. N., and Guenter, B. (2019). Photorealistic image synthesis for object instance detection. In *2019 IEEE international conference on image processing (ICIP)*, pages 66–70. IEEE.

[12] Hussain, M. (2023). When, where, and which?: Navigating the intersection of computer vision and generative ai for strategic business integration. *IEEE Access*, 11:127202–127215.

[13] Kaneva, B., Torralba, A., and Freeman, W. T. (2011). Evaluation of image features using a photorealistic virtual world. In *2011 International conference on computer vision*, pages 2282–2289. IEEE.

[14] Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410.

[15] Kettering, S. (2023). A 20/20 look into computer vision. `https://ambiq.com/blog/a-20-20-look-into-computer-vision/`. Accessed: April 25, 2024.

[16] Khairnar, S. (2019). An approach of automatic reconstruction of building models for virtual cities from open resources. Master's thesis, University of Windsor.

[17] Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154.

[18] Kulkarni, C., Druga, S., Chang, M., Fiannaca, A., Cai, C., and Terry, M. (2023). A word is worth a thousand pictures: Prompts as ai design material. *arXiv preprint arXiv:2303.12647*.

[19] Little, J. and Verri, A. (1989). Analysis of differential and matching methods for optical flow. In *[1989] Proceedings. Workshop on Visual Motion*, pages 173–180. IEEE.

[20] Longford, F. (2018). Forensic architecture. `http://surl.li/swzzc`. Accessed: April 7, 2024.

[21] Luccioni, A. S., Akiki, C., Mitchell, M., and Jernite, Y. (2023). Stable bias: Analyzing societal representations in diffusion models. *arXiv preprint arXiv:2303.11408*.

[22] Man, K. and Chahl, J. (2022). A review of synthetic image data and its use in computer vision. *Journal of Imaging*, 8(11):310.

[23] Mitash, C., Bekris, K. E., and Boularias, A. (2017). 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)*, pages 545–551. IEEE.

[24] Mouha, R. A. (2021). Internet of things (iot). *Journal of Data Analysis and Information Processing*, 9(2):77–101.

[25] Müller, M. G., Durner, M., Gawel, A., Stürzl, W., Triebel, R., and Siegwart, R. (2021). A photorealistic terrain simulation pipeline for unstructured outdoor environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9765–9772. IEEE.

[26] Munn, L., Magee, L., and Arora, V. (2023). Unmaking ai imagemaking: A methodological toolkit for critical investigation. *arXiv preprint arXiv:2307.09753*.

[27] of Canada, G. (2024). Climate. `https://climate.weather.gc.ca/glossary_e.html`. Accessed: April 22, 2024.

[28] OpenAI (Accessed: 2024). Openai clip. `https://openai.com/research/clip`. Accessed: April 7, 2024.

[29] Park, J. E., Vollmuth, P., Kim, N., and Kim, H. S. (2022). Research highlight: use of generative images created with artificial intelligence for brain tumor imaging. *Korean Journal of Radiology*, 23(5):500.

[30] Paulin, G. and Ivasic-Kos, M. (2023). Review and analysis of synthetic dataset generation methods and techniques for application in computer vision. *Artificial Intelligence Review*, 56(9):9221–9265.

[31] Pomerleau, D. A. (1988). Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1.

[32] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

[33] Rani, S., Lakhwani, K., and Kumar, S. (2022). Three dimensional objects recognition & pattern recognition technique; related challenges: A review. *Multimedia Tools and Applications*, 81(12):17303–17346.

[34] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. (2016). Generative adversarial text to image synthesis. In *International conference on machine learning*, pages 1060–1069. PMLR.

[35] Ren, X., Luo, J., Solowjow, E., Ojea, J. A., Gupta, A., Tamar, A., and Abbeel, P. (2019). Domain randomization for active pose estimation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7228–7234. IEEE.

[36] Rivera-Rubio, J., Alexiou, I., and Bharath, A. A. (2015). Appearance-based indoor localization: A comparison of patch descriptor performance. *Pattern Recognition Letters*, 66:109–117.

[37] Saxena, A., Driemeyer, J., Kearns, J., and Ng, A. (2006). Robotic grasping of novel objects. *Advances in neural information processing systems*, 19.

[38] Shumailov, I., Shumaylov, Z., Zhao, Y., Gal, Y., Papernot, N., and Anderson, R. (2023). The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493*.

[39] Sonetta, H. Y. (2021). Bridging the simulation-to-reality gap: Adapting simulation environment for object recognition. Master's thesis, University of Windsor (Canada).

[40] Taylor, G. R., Chosak, A. J., and Brewer, P. C. (2007). Ovvv: Using virtual worlds to design and evaluate surveillance systems. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[41] Technologies, U. (2024). Unity: Game development platform. `https://unity.com/`. Accessed: April 7, 2024.

[42] Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S. (2018). 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 workshops*, pages 969–977.

[43] Tsirikoglou, A., Eilertsen, G., and Unger, J. (2020). A survey of image synthesis methods for visual machine learning. In *Computer Graphics Forum*, volume 39, pages 426–451. Wiley Online Library.

[44] Veeravasarapu, V., Rothkopf, C., and Visvanathan, R. (2017). Adversarially tuned scene generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2587–2595.

[45] Villarreal, A. B. (2018). *Blender 3D Incredible Machines*. Packt Publishing Ltd.

[46] Yang, Z., Zhan, F., Liu, K., Xu, M., and Lu, S. (2023). Ai-generated images as data source: The dawn of synthetic era. *arXiv preprint arXiv:2310.01830*.

VITA AUCTORIS

NAME:	Kavan Mehulkumar Dave
PLACE OF BIRTH:	Gujarat, India
YEAR OF BIRTH:	1999
EDUCATION:	Bachelor of Engineering in Information Technology Gujarat Technological University, Gujarat, India, 2022
	University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2024