FACULTY
OF ELECTRICAL
ENGINEERING
CTU IN PRAGUE

Master thesis

Using artificial intelligence to generate content for augmented reality

Bc. Alena Žižková

Department of Computer Graphics and Integration
Supervisor: Ing. David Sedláček, Ph.D.

Prague, May 2024

Thesis Supervisor:
Ing. David Sedláček, Ph.D.
Department of Computer Graphics and Integration
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2
160 00 Prague 6
Czech Republic

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Žižková** Jméno: **Alena** Osobní číslo: **475667**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Počítačová grafika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Využití umělé inteligence pro generování obsahu pro rozšířenou realitu

Název diplomové práce anglicky:

Using artificial intelligence to generate content for augmented reality

Pokyny pro vypracování:

1) Proveďte rešerši prací využívajících umělou inteligenci (AI) pro generování 2D/3D objektů na základě kontextuální informace (např. vstup uživatele, fotografie, místo dle GPS).
2) Seznamte se s knihovnami pro rozšířenou realitu (AR). Vyberte ty, které umožňují zobrazení AR scény venku. Dále proveďte rešerši alespoň pěti aplikací, jejichž primárním cílem je zobrazení AR scény venku. Zhodnoťte jejich výhody a nevýhody z pohledu zobrazení virtuální scény a uživatelského rozhraní.
3) Navrhněte a implementujte aplikaci, která bude pomocí AI generovat a zobrazovat v rozšířené realitě objekty dle vaší volby (druh a množství objektů konzultujte s vedoucím), přičemž generované objekty budou reflektovat místo umístění, návštěvnost, případně další kontextuální informace dle vašeho návrhu.
4) Navrhněte a realizujte způsob testování kvality umístění a zobrazení virtuálních objektů aplikací, stabilitu generování obsahu a jeho přizpůsobitelnosti. Aplikaci otestujte s minimálně deseti uživateli.

Seznam doporučené literatury:

1] DreamFusion: Text-to-3D using 2D Diffusion, Poole et al.. arXiv 2022.
2] Steve Aukstakalnis. Practical Augmented Reality: A Guide to the Technologies, Applications, and Human Factors for AR and VR (Usability). Addison Wesley 2017.
3] Dieter Schmalstieg, and Tobias Hollerer. Augmented Reality: Principles and Practice (Usability), Addison Wesley 2016
4] https://developers.google.com/ar

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. David Sedláček, Ph.D. katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **13.09.2023** Termín odevzdání diplomové práce: **24.05.2024**

Platnost zadání diplomové práce: **16.02.2025**

Ing. David Sedláček, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studentky

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as schoolwork under the provisions of Article 60(1) of the Act.

In Prague, May 2024

.....
Bc. Alena Žižková

Abstract

The application of Artificial Intelligence (AI), particularly through generative models, is rapidly advancing. This thesis examines the use of generative models in creating digital content for Augmented Reality (AR) applications, an area experiencing significant growth supported by advancements in localization systems, including Visual Positioning System (VPS).

This thesis focuses on the generation of digital content for AR applications, with emphasis how such content is customized to specific geographic locations. It involves integrating location-based data—such as environmental context, points of interest, and address information—into a text prompt. This prompt then serves as input to the generative model that creates 3D models. These models are then loaded into an AR scene.

In addition to analyzing generative AI, the thesis examines crucial localization services for AR applications and explores visual enhancements that could improve appearance of digital content in the real-world environment view. A significant outcome of this research is the design and the development of a system that dynamically generates models based on the surrounding environment.

The system architecture comprises three main components: the AR application, a computation server, and a communication server with a database where all models are stored. This structure not only does support the generation of new models but also enables users to access models created by others. The thesis demonstrates the potential of using generative models to enhance AR applications with context-sensitive digital content.

Keywords: Generative artificial intelligence, text-to-image, text-to-3D, augmented reality, localization service, prompt generation, VPS, Geospatial API, Meshy, REST API server

Abstrakt

Využití umělé inteligence (AI) , zejména prostřednictvím generativních modelů, se rychle rozvíjí. Tato diplomová práce zkoumá využití generativních modelů při tvorbě digitálního obsahu pro aplikace rozšířené reality (AR). Tato oblast zaznamenala významný růst, který byl podpořený pokroky v lokalizačních systémech, včetně systémů vizuálního pozicování (VPS).

Práce se zaměřuje na generování digitálního obsahu pro aplikace AR s důrazem na to, jak je tento obsah přizpůsoben konkrétním geografickým lokalitám. Zahrnuje integraci dat založených na umístění - jako je kontext prostředí, body zájmu a informace o lokaci - do textového řetězce. Tento řetězec slouží jako vstup pro generativní model, který produkuje 3D modely. Tyto modely jsou následně načteny v AR scéně.

Kromě analýzy generativní AI se práce věnuje i zásadním lokalizačním službám pro aplikace AR a zkoumá vizuální vylepšení, které mohou zlepšit vzhled digitálního obsahu v reálném prostředí. Významným výsledkem této práce je návrh a vývoj systému, který dynamicky generuje modely na základě okolního prostředí.

Architektura systému zahrnuje tři hlavní komponenty: aplikaci rozšířené reality, výpočetní server a komunikační server s databází, kde jsou uloženy všechny modely. Tato struktura podporuje nejen generování nových modelů, ale také umožňuje uživatelům přístup k modelům vytvořeným jinými uživateli. Práce demonstruje potenciál využití generativních modelů k obohacení aplikací AR o kontextově citlivý digitální obsah.

Klíčová slova: Generativní umělá inteligence, přeměna textu na obrázek, přeměna textu na 3D model, rozšířená realita, lokalizační systémy, generování promptu, VPS, Geospatial API, Meshy, REST API server

Acknowledgements

I would like to give thanks to my supervisor Ing. David Sedláček, Ph.D. who has led and directed me in writing this thesis. I also appreciate all the help from all professors and teachers with whom I was in touch. All of them influenced me and gave me a lot of view of points to many topics. Special thanks to professor Allen C.-H. Wu from National Tsing Hua University (NTHU) who introduced me to an existence of relationship between artificial intelligence and art.

Last but not least I would like to express a massive thanks to my family and friends who have supported me during my studies.

Contents

List of Figures

List of Acronyms

AGI General Artificial Intelligence.

AI Artificial Intelligence.

ANI Artificial Narrow Intelligence.

ANN Artificial Neural Network.

API Application Programming Interface.

AR Augmented Reality.

ASI Artificial Superintelligence.

BLIP Bootstrapping Language-Image Pre-training.

CLIP Contrastive Language-Image Pre-training.

CNN Convolutional Neural Network.

DNN Deep Neural Network.

GAN Generative Adversarial Network.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

GPT Generative pre-trained transformer.

HDR High Dynamic Range.

HTTP Hypertext Transfer Protocol.

INS Inertial Navigation Systems.

IP Internet Protocol.

ML Machine Learning.

MLP Multi-layer Perceptron.

MR Mixed Reality.

NeRF Neural Radiance Fields.

REST API Representational State Transfer Application Programming Interface.

SDK Software Development Kit.

SLAM Simultaneous Localization and Mapping.

SSD Stein Score Distillation.

UI User Interface.

URL Uniform Resource Locator, an address on the web.

VPS Visual Positioning System.

VR Virtual Reality.

ZSL Zero-shot Learning.

CHAPTER 1

Introduction

The application of Artificial Intelligence (AI) is rapidly advancing, especially through generative models. This thesis explores the use of generative models to create digital content for Augmented Reality (AR) applications. This area is experiencing significant growth, bolstered by advancements in localization systems such as Visual Positioning System (VPS).

This thesis primarily focuses on generating digital content for AR environment, with a particular emphasis on adapting this content to specific geographic locations. It involves integrating location-based data—such as environmental context, points of interest, and address information—into a text prompt. This prompt is then used as input for a generative model that creates 3D models, which are subsequently loaded into an AR scene.

The analysis begins by examining localization systems used in AR and discusses visual enhancements that could improve the integration of digital content with real-world views. The role of AI is then addressed, with a detailed look at how generative AI models, specifically text-to-image and text-to-3D, work and have potential for content creation. The analysis chapter concludes with a review of related work in fields.

The next chapter presents the design of the 3D model generation system and set the stage for the implementation details that will be discussed later. The system is consists of three interconnected components: a mobile phone application, a communication server, and a computation server The implementation chapter describes these individual components, their communication, and the model creation process, which collectively enhance a better understanding of the overall system. The system's effectiveness is evaluated through testing focused on localization accuracy, visual enhancements, and user experience. The final section presents the outcomes obtained.

CHAPTER 2

Analysis

This chapter is divided into two main sections. The first part provides an introduction to Augmented Reality (AR), discussing the various technologies and techniques essential to AR, including localization, visual enhancement, and scene understanding. It concludes with an overview of development frameworks for AR applications. The second part focuses on Artificial Intelligence (AI), starting with the basics of AI and progressing to a detailed exploration of generative AI and its applications.

2.1 Augmented Reality (AR)

To clarify some terms, I will explain fundamentals of Augmented Reality and differences to other technologies. Augmented Reality (AR) extends our real world by virtual object. It displays virtual objects within the real-world environment captured by the camera in real-time. We can use our smartphones 2.1, tablets, or AR headsets, which offer hands-free interaction. We often hear about Virtual Reality (VR), which creates its own virtual world where virtual objects exist. To experience this reality, it requires having special equipment such as a headset 2.2. Another term is Mixed Reality (MR). MR combines elements from both the real and digital worlds to create new environments and visualizations, similar to AR [1]. However, in Mixed Reality, physical objects coexist with digital objects and interact in real time 2.3.

Augmented Reality (AR) can be categorized into several types based on how it interacts with the real world and the technology it uses. on what we are looking at, how to display, and what is enabling the AR (hand handle, head set, glasses) it interacts with the real world and the technology it uses.

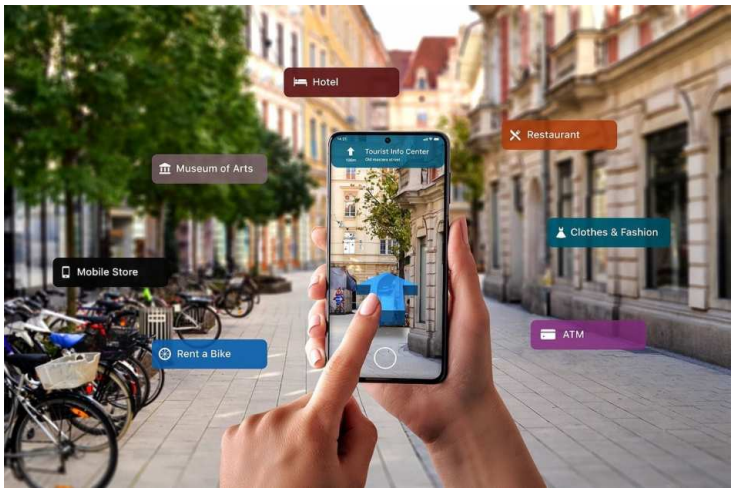

Figure 2.1: A Navigation Application in Augmented Reality [2]

Figure 2.2: Virtual Reality Training for Healthcare [3]

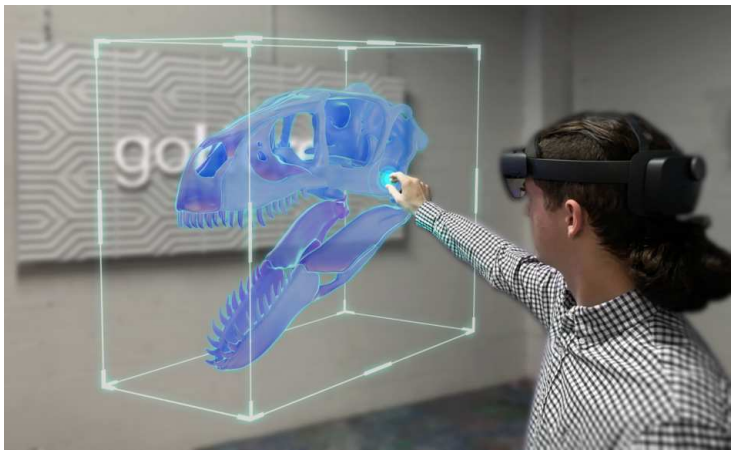

Figure 2.3: Mixed reality in education [4]

Marker-based AR [5] uses predefined markers or patterns to activate augmented content. The device features a tracking system that detects these markers, allowing digital objects to be accurately placed within a specified space. Commonly employed in advertising, gaming, and educational applications, this type of AR provides users with an interactive and immersive way to engage with digital content overlaid on the real world. Although marker-based AR is considered somewhat outdated today, it still serves particular use cases. This technology was originally developed to address localization challenges in areas where location information (e.g. GPS) is unavailable or of poor quality, such as in a subway station. [6]

Model-based AR [7] is technology that is highly dependent on object recognition. It identifies objects in the real world and either replaces or enhances their original appearance. For example, in a factory setting, superimposition-based AR could enable employees to understand the function of specific machine parts or view instructions for routine maintenance directly overlaid on the equipment.

Marker-less based AR [5], which comprises also the **Location-based AR**, operates independently of markers, instead utilizing the device's built-in components and sensors, such as the camera, GPS, and accelerometers, to detect and track the physical environment. This enables the accurate placement of digital objects or information. Virtual content is anchored to the user's real-world surroundings. Due to its independence from markers, Marker-less AR requires sophisticated systems that can identify, track, and render objects in the physical environment.

Projection-based AR [8] merges the virtual and physical worlds by projecting virtual content onto physical objects or surfaces, effectively merging these two realms. This technology uses specialized projectors to display interactive visuals directly on these surfaces. Although the system is straightforward, it

may not perform optimally in well-lit areas. As an alternative, lasers paired with depth perception sensors can be used to enhance performance and interaction under various lighting conditions. This approach typically employs spatial displays.

There are two primary approaches [8] used to display digital content in the real world. The first approach, video-mixing, captures the real environment with a camera and then integrates it with virtual information. The second approach overlays digital content onto the real-world view. This is achieved by using an optical combiner, such as a transparent monitor, exemplified by devices like the Hololens. The display of digital content can be facilitated through three types of devices. **Head-attached** displays, worn on the user's head, are one type. The most commonly used type is the hand-held display, typically in the form of tablets or smartphones, also known as video see-through displays. The main drawbacks of **hand-held** displays include their small size, limited resolution, and the necessity for the user to hold and maneuver the device, which restricts the use of both hands. These two types are often categorized as **body-attached** displays. The third type, **spatial displays**, integrates digital visuals directly into the surrounding environment.

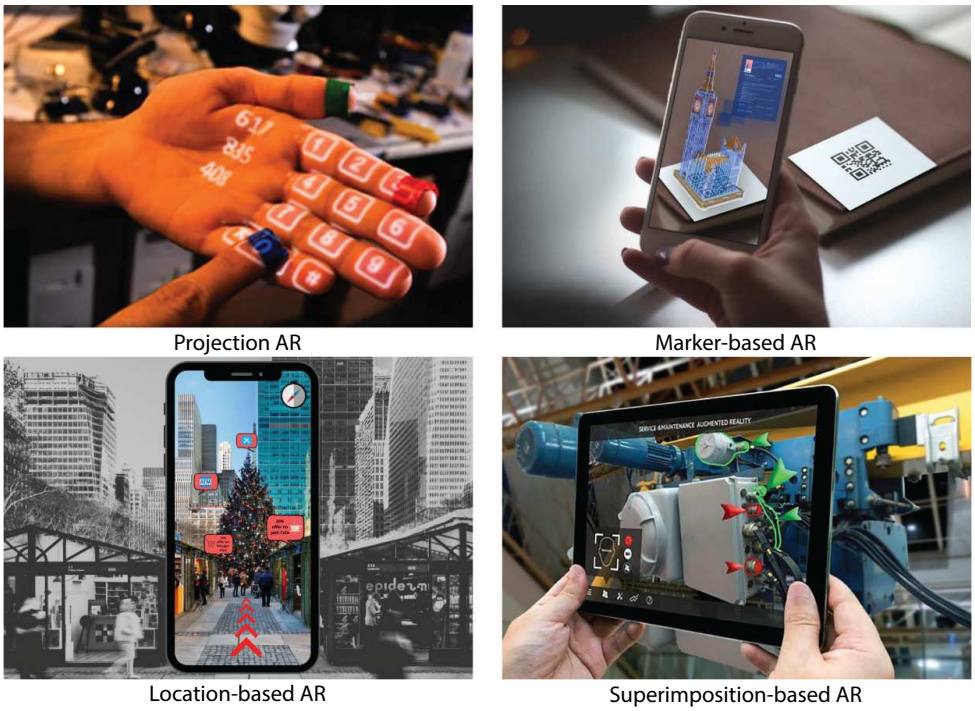

Figure 2.4: Types of Augmented Reality [9][10][11]

2.2 Localization in AR

For my application, I will utilize Location-based AR. This approach incorporates several techniques for localizing virtual objects, and the implementation can vary depending on the technology used. The selection of these techniques is tailored to meet the specific application requirements. Typically, using a combination of localization techniques ensures accurate positional information.

2.2.1 Cell Tower and Wi-Fi Triangulation/ Trilateration

This method is similar to GPS tracking in many ways. Cell tower triangulation is a technique used to estimate the location of a mobile device based on the signal strengths and geographic positions of nearby cell towers. When a device communicates with multiple cell towers, the signals can be analyzed to determine a rough location by measuring the time delay and signal strength from each tower [12]. In trilateration, the position of a mobile device is estimated by measuring its distances from multiple reference points. Trilateration is also referred to a range measurement technique (Figure 2.5). Conversely, in triangulation, a mobile device is located by calculating the angles relative to various reference points. This method does not require GPS. Wi-Fi triangulation similarly uses the signals from multiple Wi-Fi access points. It relies on the strength of the Wi-Fi signal and the known positions of the Wi-Fi routers. Wi-Fi triangulation is especially effective indoors or in urban environments where GPS signals may be weak or obstructed.

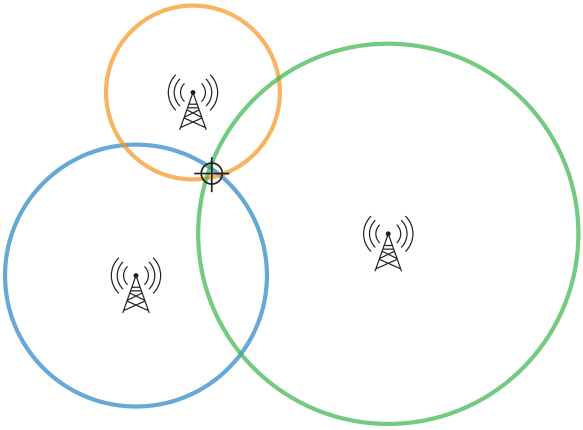

Figure 2.5: Diagram of Cell Tower Trilateration

2.2.2 Inertial Navigation Systems (INS)

These systems utilize a mix of accelerometers, gyroscopes, and sometimes magnetometers to monitor the position and orientation of a device, starting from an initial location and tracking its movements over time. Inertial Navigation Systems (INS) can function independently, but they are typically used in conjunction with other technologies such as GPS to improve the accuracy and reliability of positional data.

2.2.3 Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) technology is employed to construct maps of unfamiliar environments while simultaneously tracking the user's position within them. This feature is especially beneficial in indoor settings where GPS signals are either weak or non-existent. It facilitates the navigation of robots through unknown territories with enhanced accuracy and efficiency, proving valuable across various industries. SLAM can incorporate multiple types of sensors, such as cameras, lidar, and radar, to generate detailed and precise maps for exact localization. However, it also faces challenges, including the need of high-quality sensor data and sophisticated processing capabilities. This technology finds applications in robotics, autonomous systems, 3D reconstruction, etc.

Figure 2.6: An example map created via Simultaneous Localization and Mapping [13]

2.2.4 Global Positioning System (GPS)

The American Global Positioning System (GPS) is arguably the most widely recognized Global Navigation Satellite System (GNSS). A GNSS consists of satellites orbiting the Earth that transmit radio signals to the surface. Each satellite broadcasts data including its position and the time from its internal atomic clock. A receiver on the ground uses this signal data to determine his location in terms of latitude, longitude, (and altitude). A calculation process is similar to cell tower trilateration. Several countries and regional entities operate their own GNSS. Besides the American GPS, there are Russia's GLONASS, Europe's Galileo, China's BeiDou, India's NavIC, and Japan's QZSS. GPS is a prevalent method used in Location-Based Augmented Reality for outdoor environments. Using GPS has some drawbacks, including high power consumption and potential signal obstruction by buildings, trees, and sometimes adverse weather conditions.

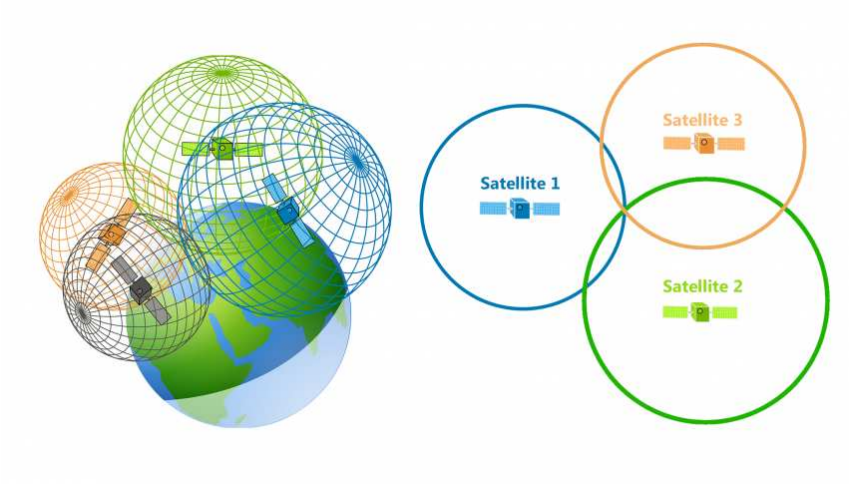

Figure 2.7: Global Navigation Satellite System trilateration [14]

2.2.5 Visual Positioning Service (VPS)

The Visual Positioning System (VPS) [15] is a relatively new technology that leverages environmental visual features to enhance the accuracy and reliability of positioning in both indoor and outdoor environ-

ments. VPS begins by creating a map from a series of images with known locations, analyzing them for key visual features such as the outlines of buildings or bridges, to create a large-scale, quickly searchable index of these features. To determine the device's location, VPS compares the features [16] in the device's camera imagery with those in the VPS index. This method is particularly effective in dense urban areas or places with poor GPS reception.
It offers location accuracy significantly better than GPS alone, as it includes additional data on the device's rotation within its environment. This positioning provides six degrees of freedom, allowing complete tracking of the device's movements. However, the precision of VPS localization significantly depends on the quality of both the imagery and its associated location data. The system's effectiveness is diminished in poor lighting condition.[17] In 2019, Scape Technologies developed a system for localization called the Vision Engine. Scape's Vision Engine [18] is a large-scale mapping pipeline that creates 3D maps from ordinary images and video. Camera devices can then use the Visual Positioning Service API to determine their precise location. The Visual Positioning Service was available in London for select partners via Scape's SDK.

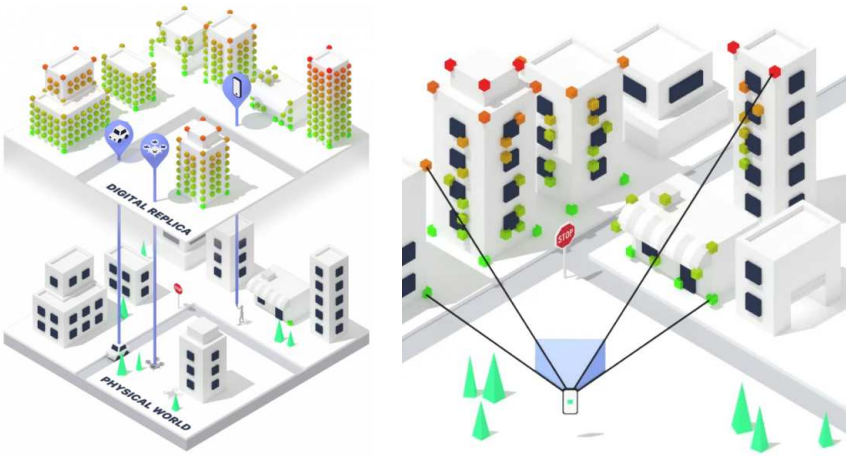

Figure 2.8: Scape's Vision Engine: a large-scale, image-based mapping and localization pipeline [19][20]

Google capitalized on its extensive collection of Street View images from Google Maps, captured globally for more than 15 years. These images serve as the foundation of Google's VPS [17]. Google identifies and describes features within these images that are likely to remain recognizable over long periods using deep neural network. These features are then aggregated from tens of billions of images to generate a 3D point cloud that models the comprehensive localization model. The model is made of trillions of data points and covers nearly all countries.

In 2019, Google introduced Google Maps Live View, marking the company's first implementation of the Visual Positioning System (VPS). This system enhances navigation by overlaying directional arrows directly onto the real-world environment using Augmented Reality (AR), helping users navigate more effectively to their destinations [21]. Google announced a service known as the Geospatial API in 2022. The technology behind the Geospatial API and Google Maps Live View is the same. The Geospatial API offers similar functionality to Scape's Vision Engine but boasts significantly broader coverage. Initially available in 87 countries, Google now claims that the VPS coverage extends to any area covered by Google Street View. The Geospatial API has been integrated into the ARCore SDK, described in Section 2.5.1, and is also referred to as the ARCore Geospatial API.

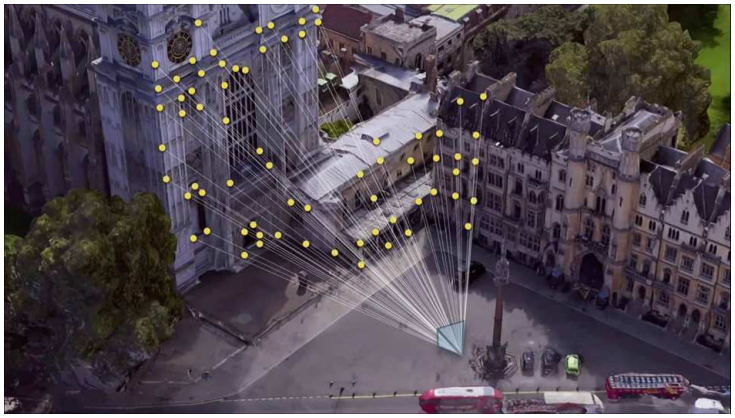

Figure 2.9: Google's Visual Positioning System using Street View images and Google Earth 3D model [16]

Apple hasn't felt behind and has also created a localization service that utilizes visual landmarks. The service incorporated into their ARKit SDK, more information in Section 2.5.2.

2.3 Visual Enhancement Techniques in AR

Rendering techniques are essential for improving the visual realism of AR applications, helping digital content integrate more smoothly with the real world. These techniques help to minimize the visual differences between digital and real objects [22]. In this section, I describe methods such as image noise, antialiasing, and motion blur, which are commonly used in AR applications. While these methods are broadly applicable, this AR application is primarily designed for outdoor use. Rendering for outdoor AR systems demands more advanced effects such as shadow mapping, light estimation, and occlusion to achieve a more realistic and immersive experience.

2.3.1 Image Noise

Each digital camera produces a distinctive type of image noise. Computing an exact noise model is very complicated and time-consuming. Therefore, a simplified noise model is used to create a noise texture that is then overlaid over the virtual objects. Adding noise [22] to digital content helps it match the graininess of real-world imagery captured by cameras, especially under lower light conditions. This technique prevents the digital objects from appearing too smooth or artificial.

2.3.2 Antialiasing

This technique smooths out the edges of digital objects, reducing the "jagged" lines that occur when high-resolution objects are rendered. Antialiasing [22] helps digital objects blend more naturally with the background, improving their perceived quality.

2.3.3 Motion Blur

Motion blur [22] mimics the blurring that occurs in a camera when capturing fast-moving objects. Applying motion blur to moving digital objects in AR creates a more realistic visual effect, as it aligns with

the user's expected perception of motion in the real world. The motion blur vector is approximated using the available camera pose information and sensor data.

2.3.4 Shadows

To make sure that the virtual object looks realistic, all the shades from the light or any other objects should fall on the right sides. Shadows ground objects to their environment, making them appear as part of the real world. Shadows are often directional and tell viewers where the light sources are coming from. Therefore, it is important to accurately estimate light sources, see Section 2.3.5.

2.3.5 Light Estimation

Light estimation is a key element for creating realistic AR experiences. Understanding of Light Estimation difficulties, it is necessary to understands fundamental light components. Ambient Light refers to the diffuse light that permeates the environment, providing general illumination that makes objects visible. It affects how all elements in a scene are lit, contributing to the overall visual coherence [23].

Shading is the variation in light intensity seen on different parts of the same object within a scene. Shading adds a depth and dimension to the scene [23]. Specular Highlights are the bright spots on surfaces that reflect light sources directly. These spots shift based on the viewer's position and light position relative to the object, enhancing the realism of the object's texture and material properties [23].

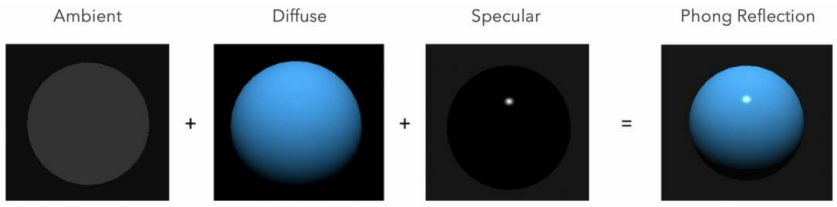

Figure 2.10: Phong reflection model combines three types of reflection: ambient, diffuse, and specular [24]

2.3.6 Environmental HDR

Environmental HDR [23] uses machine learning and advanced algorithms to estimate multiple lighting aspects simultaneously, including the intensity, color temperature, and direction of the incoming light across the scene etc. The models used are typically trained end-to-end, meaning they are designed to operate from input to output directly . This approach allows Environmental HDR to dynamically adjust to changing light conditions in real-time with high fidelity.

2.3.7 Environment Probes

Environment probes [26] are a technique that involves capturing real-world imagery through the device's camera and organizing this data into environment textures, such as cube maps or spherical maps. These textures, which capture views in all directions from a specific point, are used to realistically light virtual objects by mimicking their real-world counterparts. For example, a virtual metal ball can reflect the room around it using these textures. While environmental probes may be less demanding in terms of immediate computational power compared to Environmental HDR, they still require high-quality camera inputs for accurate environment texturing and mapping.

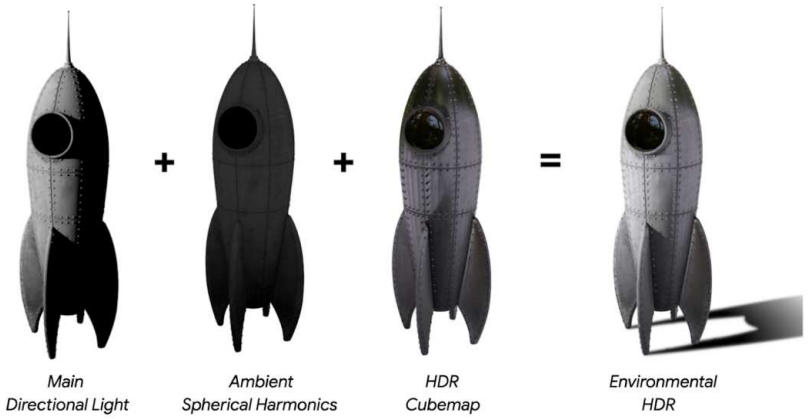

Figure 2.11: Google's Lightning API with mode Environmental HDR providing more realistic light estimation [25]

Figure 2.12: A sphere reflecting a surrounding environment using Probes [27]

2.3.8 Occlusion

Occlusion is a method that represents the visual hierarchy between virtual and real-world objects. This involves ensuring that virtual objects are correctly obscured by real-world objects when positioned behind them from the viewer's perspective. For example, if a virtual cat is behind a real plant, the plant should obscure part of the cat, just as it would obscure a real object (Figure 2.13).

Implementing occlusion can involve various methods, such as depth estimation using sensors like Time-of-Flight, LiDAR, or Kinect cameras. Alternatively, more advanced systems may predict the depth map (Figure 2.14) using machine learning techniques. Unfortunately, due to technical limitations and suboptimal imaging conditions, depth images are often low-resolution and noisy. There is extensive research, including studies like the Discrete Cosine Transform Network for Guided Depth Map Super-Resolution, aimed at achieving high-quality depth maps to enhance realism. However, computing a higher-quality depth map can be very time-consuming and resource-intensive.

Figure 2.13: The demonstration showcases [28] the occlusion effect by projecting a digital image onto a wall. On the left, with occlusion disabled, the image overlays the vase, which is closer to us in the scene. On the right, with occlusion enabled, the virtual image is displayed correctly and does not overlap the vase.

Figure 2.14: Estimation of a depth map [29] from input image, illustrating the estimated distance to the object at each pixel, with colors ranging from yellow (closest) to dark violet (farthest)

2.4 Scene Understanding

2.4.1 Scene semantic

Scene semantics enable the understanding of the surrounding environment in augmented reality. The implementation utilizes a combination of depth maps, object recognition, and semantic segmentation of images captured by the camera. The result is an image composed of labeled pixels, as it is illustrated in Figure 2.15, which helps in interpreting the scene accurately.

2.4.2 Object Detection

Object detection in scene understanding involves identifying and pinpointing various elements like vehicles, animals, and trees within an image or video frame. Detection is not limited to searching for specific objects; it also encompasses the specific detection of entire images, human bodies, or flat surfaces. Body detection targets human bodies or their parts, such as hands or faces, which is instrumental in applications like gesture recognition and human-computer interaction. Meanwhile, plane detection focuses on identifying flat surfaces within a scene, such as floors, walls, or tables, facilitating the accurate placement of virtual objects. There are much more features which can help to understand scene.

Figure 2.15: An example image of Scene semantic image created by Google Scene Semantics API [30]

2.5 Development Frameworks for AR

Creating AR applications from scratch is no longer necessary. Developers can leverage existing development frameworks that provide comprehensive libraries and tools tailored for AR. These frameworks not only simplify the development process but also improve functionality and reliability, allowing developers to focus on creating innovative and immersive AR experiences without reinventing fundamental components. This section will explore the most commonly used various AR development frameworks. Each of frameworks provides unique features and capabilities, making them suitable for various types of AR development projects depending on the specific requirements, target platform, and level of realism needed.

2.5.1 ARCore

ARCore [31] is Google's augmented reality SDK that offers cross-platform APIs for developing immersive experiences across Android, iOS, Unity, and the Web. ARCore equips developers with essential tools for crafting augmented reality experiences, including motion tracking, geospatial tracking, covered in Section 2.2.5, environmental understanding, depth perception, light estimation, and more.

2.5.2 ARKit and RealityKit

ARKit and RealityKit [32] are Apple's development kits for creating AR applications on iOS devices. ARKit utilizes Apple's camera and motion technologies to facilitate easy-to-use AR experiences. RealityKit, a newer AR SDK from Apple, enhances these experiences by processing information from ARKit. It offers advanced features that allow for greater control and customization. RealityKit includes APIs for custom rendering, metal shaders, post-processing, and object capture, utilizing state-of-the-art photogrammetry algorithms.

Feature	ARKit	ARCore
World Tracking	●	●
Image Tracking	●	●
Face Tracking	●	●
Geo Tracking	●	●
Body Tracking	●	●
Plane Detection	●	●
Scene Depth	●	●
Object Detection	●	●
Shared AR	●	●
Mashing (LiDAR)	●	●

● The feature is presented on the platform and is effective

● The feature is less effective or not available for developers

● The feature is not presented on the platform

Figure 2.16: A table showing differences features of ARKit and ARCore in 2024 [33]

2.5.3 Vuforia

Vuforia Engine [34] is another popular AR framework that supports augmented reality features across phones, tablets, and headsets. Known for its advanced computer vision capabilities, Vuforia is user-friendly and compatible with iOS, Android, and UWP (Universal Windows Platform). It provides a range of features, including image recognition, 3D object detection, smart terrain, and virtual buttons, making it versatile for both straightforward and sophisticated AR applications.

2.5.4 Kudan

Kudan [35] integrates Simultaneous Localization and Mapping (SLAM) technology, enabling machines to determine their location by analyzing their movements and the surrounding environment. It offers unlimited map scaling and cross-platform compatibility, along with enhanced robustness and accuracy.

2.5.5 AR Foundation

[36] AR Foundation [36] is a Unity [37] framework that enables developers to build rich and interactive Augmented Reality (AR) experiences across multiple platforms through a variety of plugins. These include the Google ARCore XR Plugin for Android, Apple ARKit XR Plugin for iOS, Apple visionOS XR Plugin on visionOS, OpenXR Plugin on HoloLens 2, and Unity OpenXR for Meta Quest. AR Foundation provides interfaces for AR functionalities without implementing the features directly. To utilize AR Foundation on a specific platform, developers need to choose the appropriate plugin package for that platform.

2.5.6 8th Wall

8th Wall [38] is a prominent platform for creating web-based augmented reality (WebAR) experiences. It enables developers to create AR applications directly on the web browser, removing the need for users to download any apps. This technology supports a wide range of devices including smartphones, computers, AR/VR headsets, and even smart glasses, making it extremely universal.

For developers, 8th Wall provides extensive tools and resources to create engaging AR content. This includes everything from hosting AR experiences on cloud platforms to integrating multimedia elements like 3D models and animations. Local testing and cloud deployment are supported, which is crucial for developing and scaling AR applications effectively.

2.6 Artificial Intelligence (AI)

Initially, it is needed to specify what Artificial Intelligence (AI) means. For this purpose, I'll refer to the definition provided in IBM's article [39]. "Artificial intelligence, or AI, is technology that enables computers and machines to simulate human intelligence and problem-solving capabilities." AI makes it possible for a machine to learn from experience, adjust to new inputs and solve the task.
The phrase "Artificial Intelligence" first emerged in 1956. The initial implementation of Neural Networks occurred between the 1950s and 1970s. The period from the 1980s to the 2010s saw a rise in the public's understanding of Machine Learning. From 2011 to the 2020s, Deep Learning advancements have fueled a surge in AI development. Currently, the most notable progress is observed in Generative AI, which has gained immense popularity [40]. This rise in AI is primarily attributed to tools like generative pre-trained models, including ChatGPT, as well as various text-to-image applications such as Stable Diffusion, Midjourney, and DALL-E. Despite ChatGPT's widespread popularity leading many to associate it directly with AI Nevertheless, it represents only a small portion of the current and potential future applications of AI technology.

AI, covers a range of fields including game playing, expert systems, neural networks, natural language processing, robotics and so on. Currently, no computers exhibit full artificial intelligence (that is, are able to simulate human behavior). However, the best AI system [41] are now capable of beating humans in various game, for example AlphaGo, Chess, Poker, or Dota 2.

AI works by combining large amounts of data with fast, iterative processing and advanced algorithms, which enables the software to independently learn from the patterns and features present in the data. There are three kinds of artificial intelligence based on capabilities, narrow, general and super AI [42].

Narrow AI

Narrow AI, also known as Weak AI or Artificial Narrow Intelligence (ANI), is trained to perform specific or narrow tasks. Often, ANI can solve a single task much faster than a human mind. Focusing on a specific spectrum offers the advantage of creating very robust applications; however, it is limited as it can't perform tasks outside of its defined scope. Applications such as ChatGPT, Alexa, and Siri are considered forms of Narrow AI. ANI drives most of the AI surrounding us today, as any other form of AI is theoretical. It is not yet possible to achieve these forms.

General AI

AGI, or General AI, is a theoretical form of AI where a machine would possess intelligence equivalent to that of humans. General AI would be capable of using previous learnings and skills to perform new tasks in different contexts, all without any human interference.

Super AI

Super AI, or ASI, is, much like General AI, a purely theoretical concept. This form of AI is envisioned to possess thinking, reasoning, learning, and judgment capabilities, along with cognitive abilities surpassing those of humans. These machines would surpass us in capability, evolving to a level that goes beyond our understanding. Although ASI and AGI currently exist only in theory, with no practical implementations to date, this does not deter AI researchers from investigating their potential development.

Artificial intelligence is often mentioned together with neural networks, machine learning and deep learning. Therefore, I will provide a brief introduction to these subfields of AI.

2.6.1 Artificial Neural Networks (ANNs)

A neural network is a computational model inspired by the way biological neural networks in the human brain process information. It's a key technology in the field of Artificial Intelligence (AI) and machine learning, and it's particularly adept at handling tasks that involve recognizing patterns, making predictions, or decision-making.

The fundamental units of a neural network are neurons, also can be called nodes. A basic neural network has interconnected artificial neurons in three layers, as you can see in the Figure 2.17.

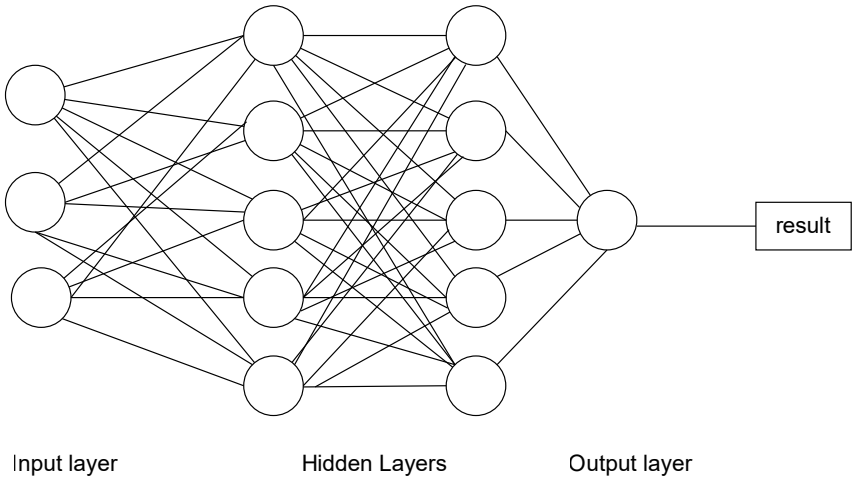

Figure 2.17: Illustration of a neural network, highlighting its distinct layers. The initial layer on the left is labeled as the input layer. Situated in the middle are the hidden layers. The rightmost layer is designated as the output layer. Output layer in this instance contains a single node but could potentially consist of several nodes.

Data enters an artificial neural network through the input layer, where the input nodes handle initial data processing, analysis, or categorization before passing it to the hidden layers [43]. The network's hidden layers, which may be numerous, receive their inputs either from the input layer or other hidden

layers. Each of these layers further processes the data received from its predecessor and then transfers it to the next layer.

Finally, the output layer delivers the end result of the neural network's data processing. This layer might have a single node or multiple nodes, depending on the task. For example, in a binary classification problem (yes/no), there would be one output node producing a result as either 1 or 0. In contrast, a multi-class classification problem would necessitate multiple output nodes in the output layer. The operation of these networks is a simulation of biological neuronal behavior, achieved through mathematical functions. Neural networks are used in fields such as machine learning and deep learning to analyze and 'learn' from large volumes of data.

The process of training neural networks refers to the phase where a new model learns from a dataset. The model's weights are typically initialized randomly. The main goal of training is to adjust the model's weights based on the error between its predictions and the actual outcomes, using an algorithm such as gradient descent. Training is stopped when the model's performance is deemed satisfactory, i.e., when the error is minimized.

Initial training focuses on generalizing models; however, the model can further optimized by tailoring it to a more specific task. The optimization process of a trained model is called fine-tuning. Fine-tuning [44] involves starting with a pre-trained model and continuing the training process to adapt it to a specific, often more limited task, and typically using a smaller dataset.

2.6.2 Machine Learning (ML)

Machine learning [45], a branch of artificial intelligence, allows systems to derive knowledge from data instead of depending on explicit programming. It uses methods from neural networks, statistics, operations research and physics to extract features from the data. Machine learning encompasses a variety of techniques, visualized in Figure 2.18, each suited for different types of problems and data. Not all machine learning techniques cover neural networks, there are other approaches and algorithms used in machine learning as well.

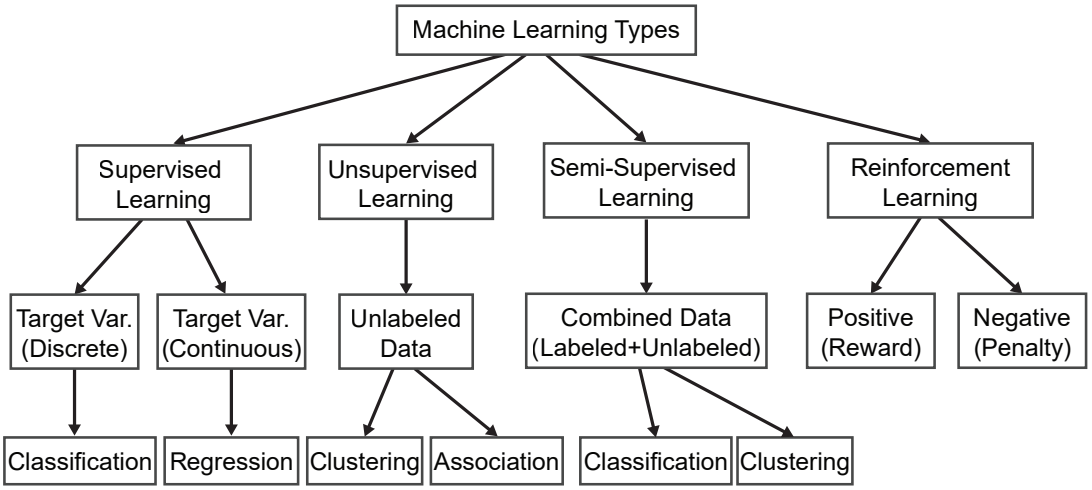

Figure 2.18: Various types of machine learning techniques[46]

2.6.3 Deep Learning

Deep learning [47], a technique in machine learning, uses layered neural networks to iteratively learn from data. The term "deep" refers to the number of hidden layers in the neural network, often called deep neural networks (DNNs). While the number of these layers can range from a few to thousands, and in theory, there's no upper limit, standard DNNs typically reach hundreds of layers. The exact number depends on the complexity of the task and the architecture of the network.

A Convolutional Neural Network (CNN or ConvNet) [39] is a specialized field within deep learning that excels in pattern recognition. Typically, CNNs are used for processing visual imagery, as well as speech and audio signals.

In certain research papers, the term MLP is used, standing for Multi-layer Perceptron [48]. MLP describes a type of modern feedforward artificial neural network characterized by a relatively small number of hidden layers. The quantity of these layers is directly related to the training duration; smaller neural networks with fewer layers typically train faster than larger, deep neural networks.

Zero-shot learning (ZSL) [49] is a machine learning technique that allows models to recognize and classify objects, concepts that they have not explicitly seen during training. This capability is particularly useful in scenarios where labeled data is scarce or when it's impractical to obtain training samples for every possible category.

2.7 Generative AI

Generative AI encompasses deep-learning models capable of processing raw data, like the entirety of Google's content or Picasso's complete works, and learning to produce outputs that are statistically likely when given prompts. In essence, generative models encode the simplified form of training data and then use them to produce new content. This newly generated content bears a similarity to the original data, but it is not identical.

Generative models [50] have been used for years in statistics to analyze numerical data. However, the rise of deep learning, has enabled their extension to images, video, music, speech, text, software code and product designs, and other complex data types.

Let's define a few key terms 2.7.1, 2.7.2 before exploring generative techniques in detail.

2.7.1 Generative Adversarial Networks (GANs)

Generative Adversarial Networks [51], or GANs for short, are an approach to generative modeling using deep learning methods, such as convolutional neural networks. GAN is unsupervised learning, and it effectively supervises itself. The GAN architecture was first described in the 2014 paper by Ian Goodfellow, et al. titled "Generative Adversarial Networks."[52]

I will borrow the description of a GAN from [53], page 669:

> 'It is based on a game theoretic scenario in which the generator network must compete against an adversary. The generator network directly produces samples. Its adversary, the

discriminator network, attempts to distinguish between samples drawn from the training data and samples drawn from the generator.'

GAN consists of two neural networks which stands against each other: the generator and the discriminator. The generator's role is to produce new examples, while the discriminator's task is to learn distinguish whether a given sample is fake (generated) or real (from the domain set), depicted in Figure 2.19. If the discriminator recognizes a fake sample, then the generator updates its model to generate fake samples that look like those from the training dataset. If the discriminator classifies a generated sample as real, then the discriminator updates its network's weights. In this way, the networks are trained together, continually improving over time.

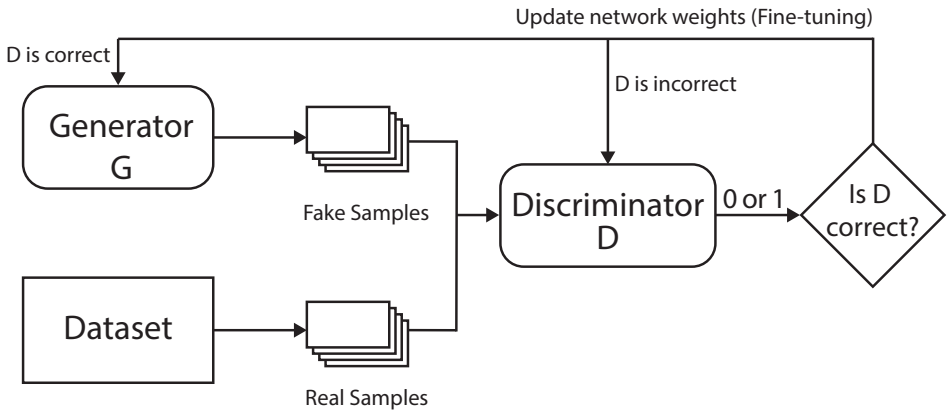

Figure 2.19: Overview of a GAN scheme [54]

StyleGAN [55] architectures have shown an unprecedented quality of RGB image generation. They are, however, designed to generate single RGB view rather than 3D content. StyleSDF [56], a method for generating 3D-consistent 1024x1024 RGB images and geometry, trained only on single-view RGB images.

2.7.2 Neural Radiance Fields (NeRF)

Neural Radiance Fields (NeRF) [57] is a technique that produces excellent results in creating new views of complex scenes. The method uses a specific deep network to represent a scene.

The deep network is trained with images of the scene taken from different angles. The number of images required for training is significantly less than what is typically needed for standalone 3D reconstruction. This approach is particularly useful in 3D reconstruction, where it's necessary to compute an unrepresented view angle in cases where a mesh lacks detail. The network processes a 5D coordinate, returning the volume density and view-dependent emitted radiance at that point (Figure 2.20). We create views by inputting 5D coordinates along camera rays as it is shown in Figure 2.21, employing traditional volume rendering methods to translate the resulting colors (Figure 2.22) and densities into an image. Since volume rendering is inherently differentiable, the only input required to optimize the model is a collection of images with known camera poses.

Neural Radiance Fields enable us to not only render an image of the scene from any viewpoint but also to export the scene as a mesh.

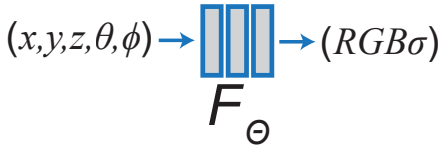

Figure 2.20: The NeRF algorithm's scheme, network F processing a 5D coordinate and outputting color and density [57]

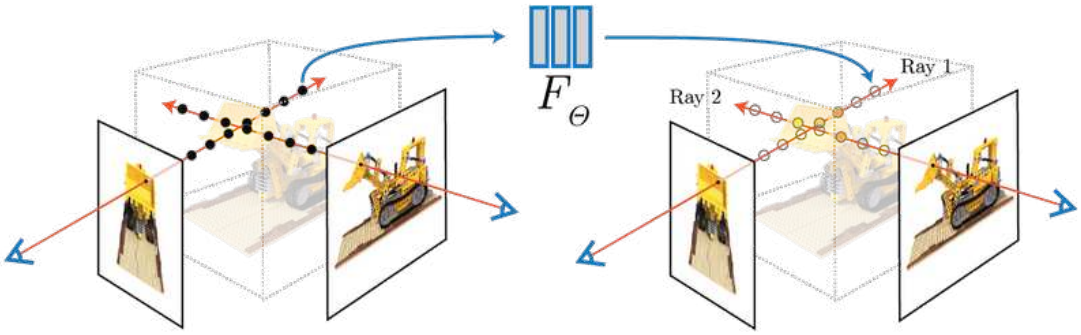

Figure 2.21: Computing the view of the scene using volume rendering [57]

2.8 Image-to-text (Image captioning)

This section does not involve scanning an image containing text and converting that image to text format. It is about brief description of the image captioning, where an algorithm predict a textual description of a scene inside an image. One of the method how to achieve this task is CLIP, which is used in many other image captioning algorithms (CLIPMatrix, Clip-Forge, CLIPCap, zero-shot CLIP, StyleCLIPDraw) and even in some generative algorithms. Another used method is BLIP, which is developed more recently.

2.8.1 Standard Image Captioning Models

Most image captioning models utilize a combination of Convolutional Neural Networks and Recurrent Neural Networks (RNNs) or more recent architectures like Transformers. These models are consisted of a well-established encoder-decoder network architecture (Figure 2.23). Firstly, we extract visual features from images using CNN, this part of the model acts as an encoder, converting the raw image data into numerical or vector representations. Then we use RNN or a Transformer, which serves as a decoder, converts encoded data into text format by generating a sequence of words. This process translates the visual features into a coherent sentence. These models are typically trained end-to-end with a large dataset of images and corresponding captions, learning to predict the next word in the sequence given the image and the previous words.

2.8.2 Contrastive Language-Image Pre-training (CLIP)

In January 2021, OpenAI introduced research titled Learning Transferable Visual Models From Natural Language Supervision [59], where presents neural network, which effectively learns visual concept from natural language supervision, learning visual representation from text paired with images. This idea was described much earlier, however all of approaches were unsupervised, self-supervised, weakly supervised, and supervised supervised respectively, and they don't work as well as CLIP.

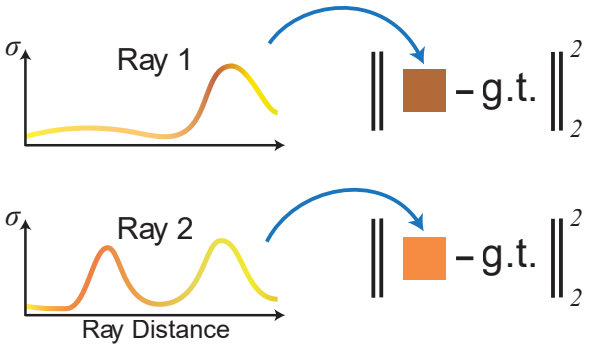

Figure 2.22: Scheme for obtaining the resulting color for each ray [57]

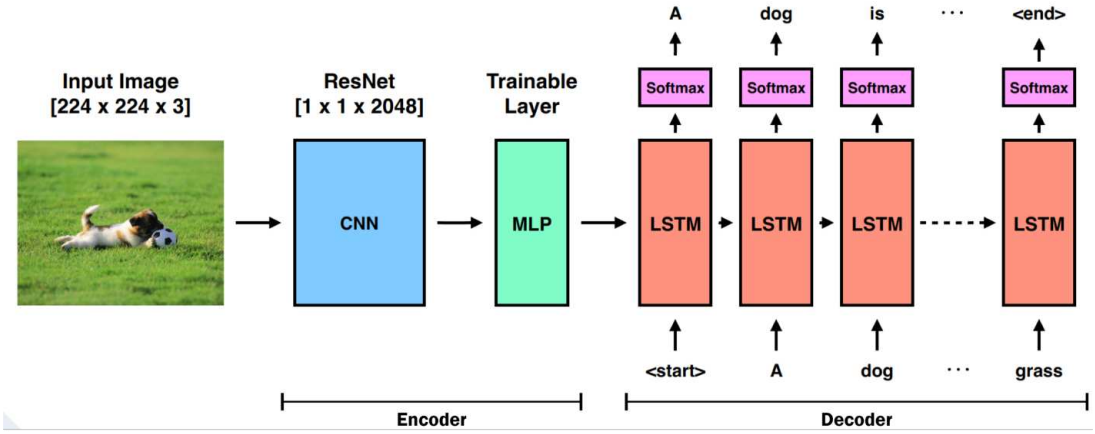

Figure 2.23: Encoder-decoder architecture consisting of an encoder transforming the image into a latent space and an LSTM (Long short-term memory) based decoder for generating the captions[58]

In this methodology, an image dataset is processed through an Image Encoder that generates a vector representation for each image. Similarly, a text dataset is subjected to a Text Encoder, which likewise produces a vector representation for each textual entity. Commonly, the Image Encoder utilizes architectures such as ResNet or Vision Transformer, while the Text Encoder employs models like CBOW or Text Transformer. We then establish correspondences, identifying which image vector matches each text vector. The goal is to maximize the similarity of the correct image-text pairs and minimize the similarity of incorrect pairs. This is often visualized in the diagram 2.24 in a matrix, where the highlighted diagonal cells indicate the correct pairings.

2.9 Text-to-image generation

The concept of text-to-image generation was first introduced in in 2015. Since then, the field has evolved significantly, leading to a variety of innovative methods. In this section, I will discuss two primary approaches: the GAN-based method and the Diffusion-based method.

2.9.1 GAN-based method

The first method used for text-to-image generation is based on the GAN architecture [60]2.26, as described in the previous sections 2.7.1. The Generator receives a noise vector and a textual description, which is encoded using a text encoder, denoted as φ. The Generator combines this noise vector with the

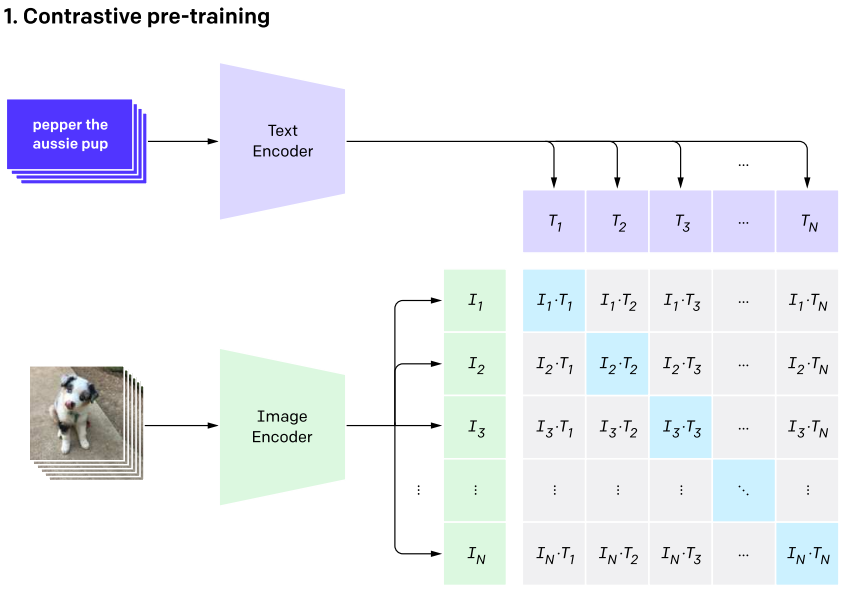

Figure 2.24: Pre-training process of CLIP method [59]

text vector to generate an image that reflects the content of the text.

The Discriminator receives an image and the corresponding text as input. Its task is to determine whether the image is a real image from the dataset that matches the text or a fake image created by the Generator.

At the end of the training process, we obtain a deep convolutional generative adversarial network that is conditioned on text features.

2.9.2 Diffusion-based method

Another approach involves using diffusion models, which operates in two distinct steps [50], which are forward diffusion and reverse diffusion (Figure 2.27). The forward diffusion process progressively transforms an image x_0 into uniform Gaussian noise x_T over T time steps by incrementally adding noise at each step. Conversely, the backward process involves the sequential removal of this noise. By identifying the reverse distribution, it becomes possible to reconstruct the original image from the Gaussian noise $x_T \sim N(0,I)$. We will train a model to simulate distribution. Diffusion models [61] are a type of generative model that transform Gaussian noise into samples from a trained data distribution through an iterative denoising process. One such model is the Denoising U-Net, which is used in many diffusion models, including Stable Diffusion.

Training objectives for image-text models show that text encoders capture meaningful visual representations, essential for text-to-image tasks. Text encoders in current text-to-image models are typically trained on paired image-text data (CLIP) and can either be developed from scratch. Large language models such as BERT and GPT are also excellent choices for encoding text in text-to-image generation tasks. Their recent advancements have greatly enhanced text understanding and generative abilities. The language models are usually much larger than the text encoders used in contemporary image-text models.

Diffusion models typically generate lower-resolution images due to the computational complexity, training requirements. Techniques like super-resolution can be applied to enhance the image quality

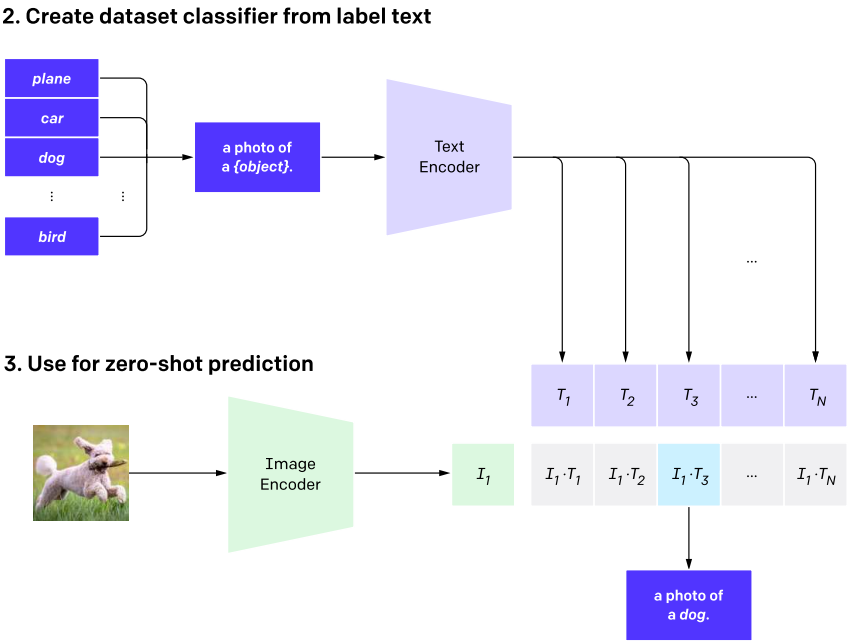

Figure 2.25: During the testing process of the CLIP model, the text encoder synthesizes a zero-shot classifier by embedding the descriptions of the target dataset's classes [59]

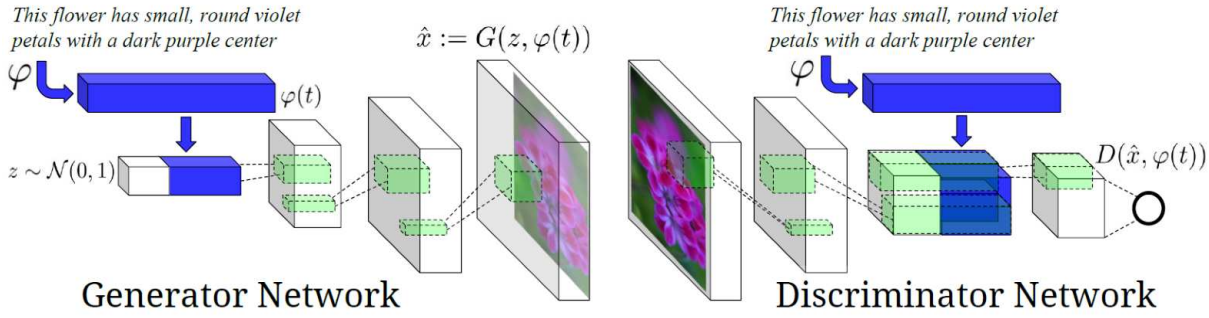

Figure 2.26: Text-conditional convolutional GAN architecture using text encoding $\phi(t)$ [60]

post-generation or training methods can be adjusted to gradually scale up the resolution, this is also implemented in the Imagen model (Figure2.28). Imagen, a text-to-image diffusion model created by Google, is known for producing highly realistic and detailed images.

Diffusion models have become popular because they can produce results that are often more realistic and detailed compared to other types of generative models, such as (GANs).

There is an overview (Figure 2.29) of works focuses on the text-to-image task over time.

2.10 Text-to-3D model generation

Generating 3D objects is significantly more complex than synthesizing 2D images. 3D generative models can be trained on explicit structural representations such as voxels and point clouds. Although the availability of 3D data is generally less extensive than that for 2D images, the principles of 3D generation often involve adapting methods originally developed for 2D image generation, among other techniques.

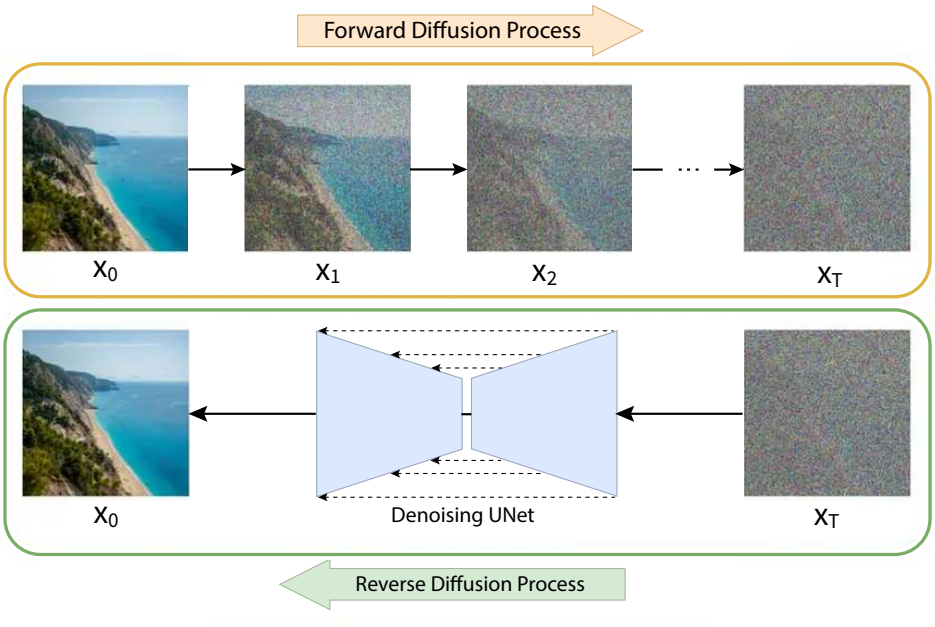

Figure 2.27: The diffusion and denoising process [62]

This approach allows for the generation of 3D models without the need for explicit 3D data.

DreamFusion is the first work that successfully applies diffusion models to 3D object synthesis. Inspired by Dream Fields which applies 2D image-text models (i.e., CLIP) for 3D synthesis.[63]

2.10.1 Dream Fields

DreamFields [64] is a method that starts by initializing a Neural Radiance Fields (NeRF) model. The process involves rendering images from the NeRF at random camera angles, shown in Figure 2.30. A critical aspect of DreamFields is its integration with CLIP, a text-image matching model trained on a large dataset of images and captions. CLIP evaluates whether the images rendered from the NeRF match the input text prompt. If the images and the text prompt do not correspond well (as determined by CLIP), the NeRF model is updated accordingly.

The procedure is iterative: images are rendered from various viewpoints, assessed by CLIP, and the NeRF is fine-tuned. This cycle continues until CLIP consistently verifies a strong correspondence between the rendered images and the text prompt. When this alignment is consistently achieved, the NeRF model is considered effectively tailored to the specific text prompt.

Dream Fields has several limitations, including the requirement for iterative optimization, which can be resource-intensive and costly.

2.10.2 Dream Fusion

Dream Fusion [65] is a method inspired by Dream Fields, mentioned in Section 2.10.1. But this system replaces CLIP system by diffusion model. Dream Fusion uses non modified diffusion model called Imagen.

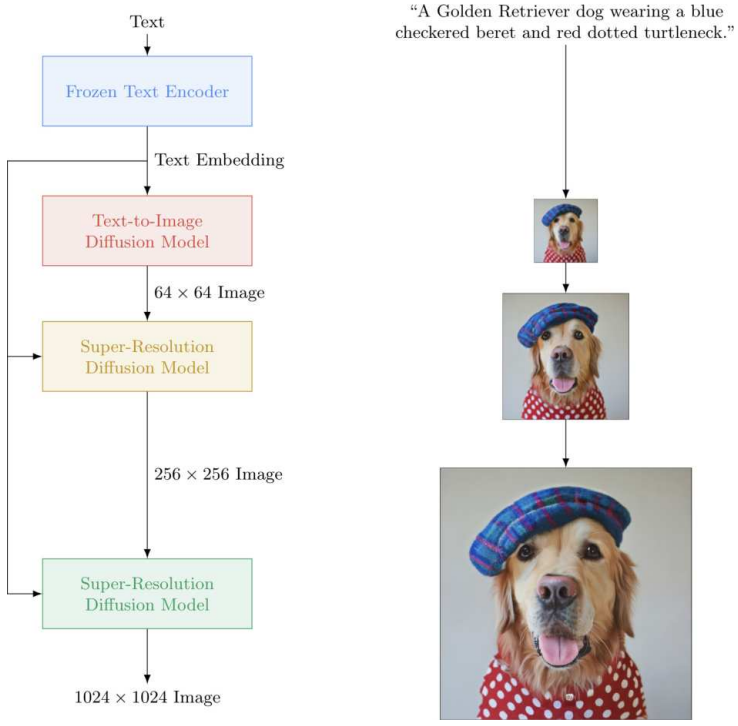

Figure 2.28: Pipeline of the Imagen diffusion model, which creates high resolution images using Super-Resolution models [61]

DreamFusion trains a randomly initialized NeRF Multi-layer Perceptron (MLP) per scene from scratch with the distillation of a pretrained 2D diffusion model (i.e., Imagen). Score distillation is a common method to update 3D parameters by lifting and backpropagating scores that are averaged over different views. Training [66] contains a process which is looped to cycle and it stops when convergence is reached. The process illustrated in Figure 2.31 is described by following steps:

1. Set a random camera pose and cast rays through scene
2. Compute density and color along rays using NeRF
3. Compute shape normals using autodiff
4. Compute Lambertian shading model with a random light direction
5. Shade the scene
6. Render a 2D image, alpha compositing along rays
7. Diffuse rendering using a random noise and noise level
8. Run Imagen diffusion model to predict a denoised image
9. Compute noise content, a linear combination of the rendering, noise, denoised prediction
10. Compute low variance update direction in pixel space
11. Backpropagate update to NeRF weights, and take an optizer step

Once the model is satisfactory, we can either maintain this NeRF representation to render any view of the scene or convert it into a mesh.

2.10.3 SteinDreamer

One of the latest advancements in text-to-3D generation is SteinDreamer [67] 2.32, which enhances Score Distillation through variance reduction using the Stein identity.

The Stein identity has been designed to minimize variance in Monte Carlo estimations for text-to-3D

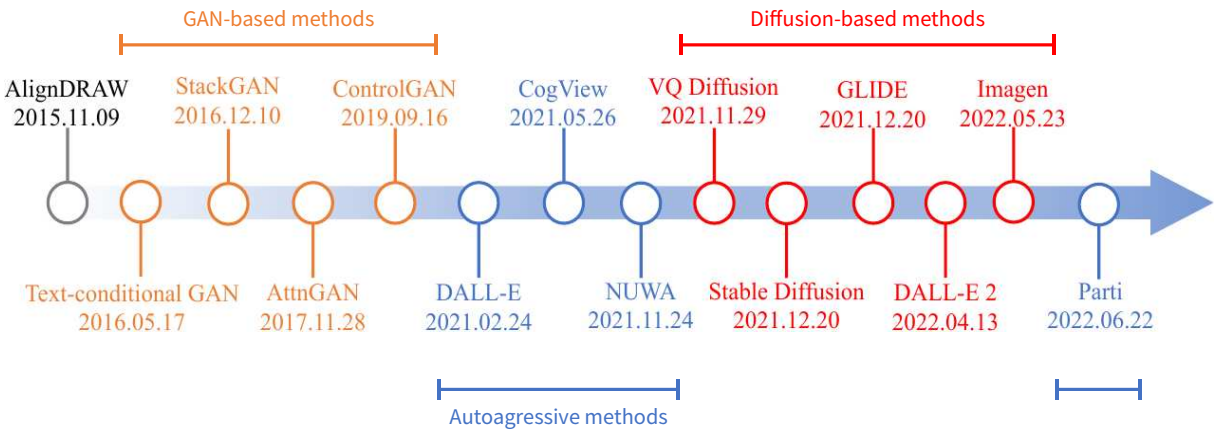

Figure 2.29: Timeline of works focuses on text-to-image task over time [63]

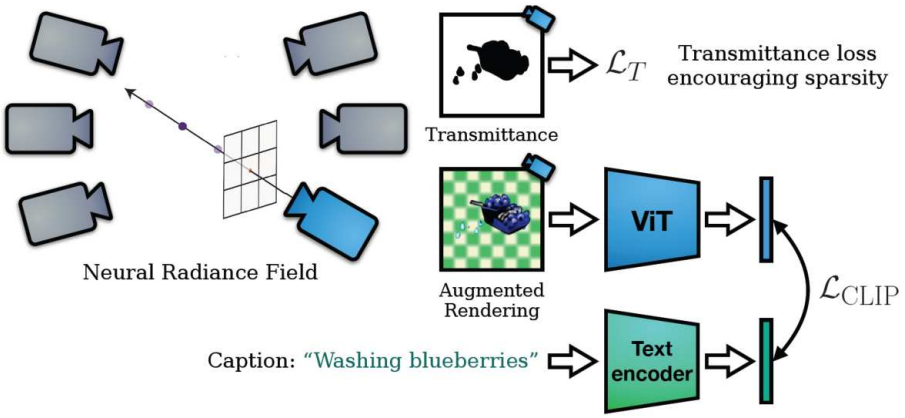

Figure 2.30: DreamFields's pipeline: Rendering images of the object from random camera poses, CLIP evaluates the relevance of renders to a given text using frozen pretrained image and text encoders (ViT - Visual Transformers)[64]

score distillation (i.e., Stein Score Distillation - SSD). SSD [66] effectively reduces variance by using flexible guidelines and network architectures specifically optimized for this purpose. Research findings indicate that the SSD method not only significantly lowers variance during distillation but also enhances the visual quality of generated objects and scenes. Additionally, a key benefit of using SteinDreamer, which employs SSD, is its faster convergence compared to traditional methods.

2.11 Available Data

To effectively analyze the wealth of data accessible from user devices and locations, it is essential to incorporate various sensor outputs available on mobile phones. These sensors provide critical data that serve as inputs for model generation. Notably, camera and location data are among the most valuable. For instance, a mobile camera can capture images of the surrounding environment or the specific location of

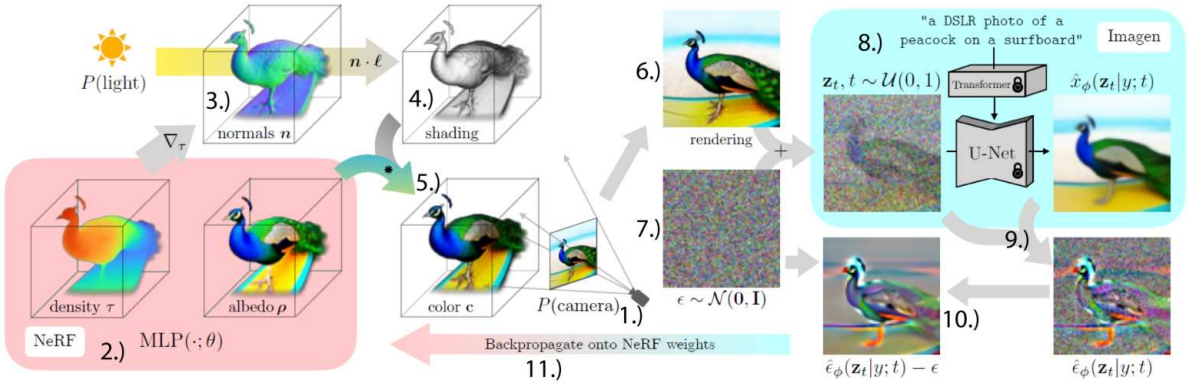

Figure 2.31: Diagram of DreamFusion process, each step is numbered [65]

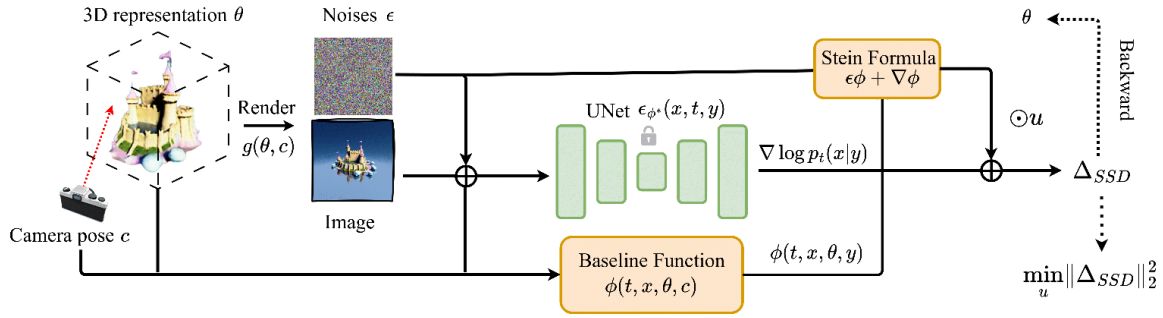

Figure 2.32: Pipeline of SteinDreamer [67]

a plant object. Location data, derived through several methodologies (Section 2.2) including GPS coordinates, can reveal detailed geographic information such as the name of the country, city, street, or nearby street names. Similar to how users locate nearby restaurants, we can identify points of interest within a certain radius. Additionally, requests for street views of specific locations are feasible. Various mapping applications such as Google Maps, Apple Maps, and Kakao Maps offer these functionalities through their APIs. Moreover, platforms like Google Cloud [68] provide extended data services, including air quality indices, pollen counts, solar energy estimates on rooftops, and local time zone information.

2.12 Related work

In this section, I will present some projects and applications related to this thesis. Augmented reality is breaking educational boundaries, and we can meet with many educational applications. One of them, Virtual Veda, aims to connect users with nature. The next two sections focus on the AR techniques used in each tool.[21]

PokémonGo

PokémonGo [69] is a widely known mobile game developed and published by Niantic in collaboration with Nintendo and The Pokémon Company in 2016. In Pokémon Go, players use their smartphones to

capture, battle, and train virtual creatures, called Pokémon, which appear on the device's screen as though they exist in the real world. The game utilizes GPS to map players' real-world movements to the in-game environment, encouraging players to travel to different physical locations to find new Pokémon species and participate in game-related activities like Gym battles and Raid fights. Pokémon Go is designed to be interactive and social, promoting physical activity and exploration by connecting the virtual and physical worlds. In the year 2018, PokémonGo announced an extension of application with AR mode. AR mode was initially launched for iOS devices that support Apple's ARKit technology (covered in Section 2.5.2) and later expanded to include Android devices compatible with Google's ARCore (Section 2.5.1).

Figure 2.33: Screenshots from PokémonGo game [70] gameplay, showing different aspects of the game in action: on the left, a user catches a Pokémon in the virtual world; in the middle, a view of a Pokémon in AR+ mode; on the right, the user's location projected into the game environment

IKEA Place

In 2017, the IKEA Place app [71] was introduced to assist users in virtually positioning furniture in their homes, demonstrated in Figure 2.34. Featuring over 2,000 IKEA products—from sofas and armchairs to coffee tables—all items in the app are displayed as 3D models that are true to scale, ensuring that each piece fits perfectly in terms of size, design, and function. Developed using Apple's ARKit technology (Section 2.5.2), IKEA Place represents a significant step in IKEA's digital transformation. IKEA was among the first home furnishing brands globally to offer this advanced technology to consumers.

Virtual Veda

"Virtual Veda – Visualize Plants through Augmented Reality" [72] is an innovative project that uses augmented reality to enhance educational experiences about flora. It allows users to interact with and learn about medicinal plants within ecosystems, understanding their significance and conservation needs. The project includes audio explanations to make the content accessible to visually impaired individuals, aiming to deepen the connection between users and nature, promoting environmental responsibility.

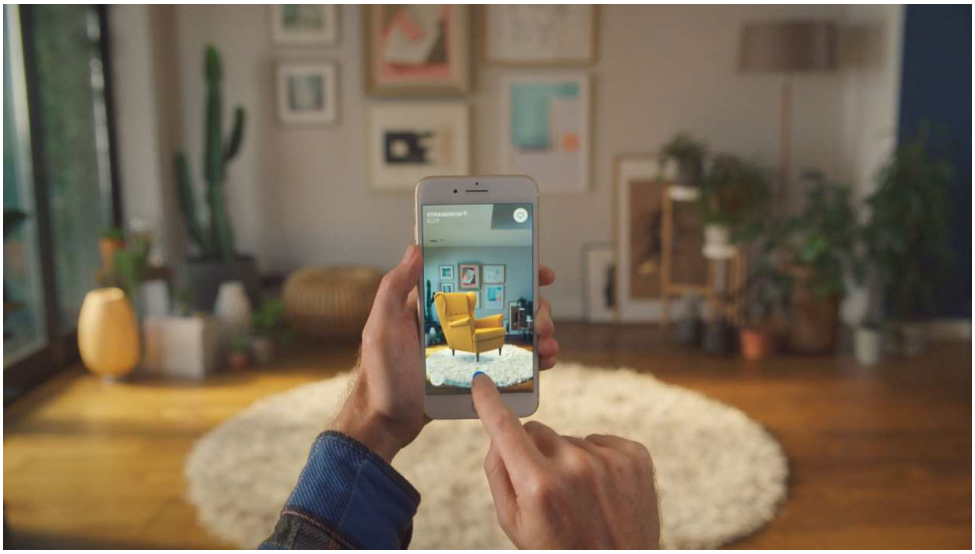

Figure 2.34: A demonstration image of Ikea Place application [71]

Quiver

Quiver - a 3D Coloring App [73], is an AR application for children that blends education with entertainment. Developed in 2017 by Quiver Vision, this app motivates children to color in a coloring book while acquiring new knowledge. When a child finishes coloring, they can scan the page, and the application displays a 3D model of their artwork through augmented reality (Figure 2.35). The coloring books are downloadable from coloring packs available in the Quiver app.

Figure 2.35: A graphic illustration showing a 3D model of a bird using the Quiver application, where the model is ”generated” from a scanned image [73]

2.12.1 Specialization in Localization

The following application utilizes modern localization techniques such as VPS that was described in Section 2.2.5.

Google Maps Live View

Google Maps Live View [74] is directly integrated into the Google Maps app and is the first feature from Google to utilize the Visual Positioning System. Google Maps Live View is an AR feature that enhances navigation by overlaying digital directions, such as arrows and markers, onto the real-world environment through the user's smartphone camera (Figure 2.36). This assists users in navigating complex spaces by providing intuitive visual cues in real-time. Initially launched for outdoor use, it has since expanded to support indoor navigation in specific locations where Street View is available, helping users find points of interest like restrooms or lounges in unfamiliar settings such as airports.

Figure 2.36: Display of the Google Maps Live View feature in action [75]

Pocket Garden

Pocket Garden[76] is a playful demo application from Google designed to demonstrate the capabilities of the Geospatial API, which allows for the attachment of content to any location covered by Google Street View. This API combines a user's local coordinates with the global coordinates from the Visual Positioning System, ensuring that 3D assets are connected to a global coordinate system rather than just the device's local coordinates. This open-source project is available on GitHub and serves as a foundational tool for developers looking to conduct their own experiments. In the app, users plant seeds and water them to promote growth. The plants then grow, blossom, and eventually bear fruit (Figure 2.37). This fruit can be harvested for more seeds to expand the garden. However, the variety of plants available is limited to a fixed set of plant models.

2.12.2 Specialization in Generative AI

There are numerous generative tools, such as 3dfy, Sloyd, Luma Labs, 3D CSM AI, Stable Projectorz, and others, that generate models from text inputs. However, these tools often require the user to take extra steps, such as selecting the category of the generated object. Consequently, I chose not to detail these tools extensively. Instead, I focused on a select few tools that are capable of generating 3D assets directly from a given prompt without requiring any additional inputs.

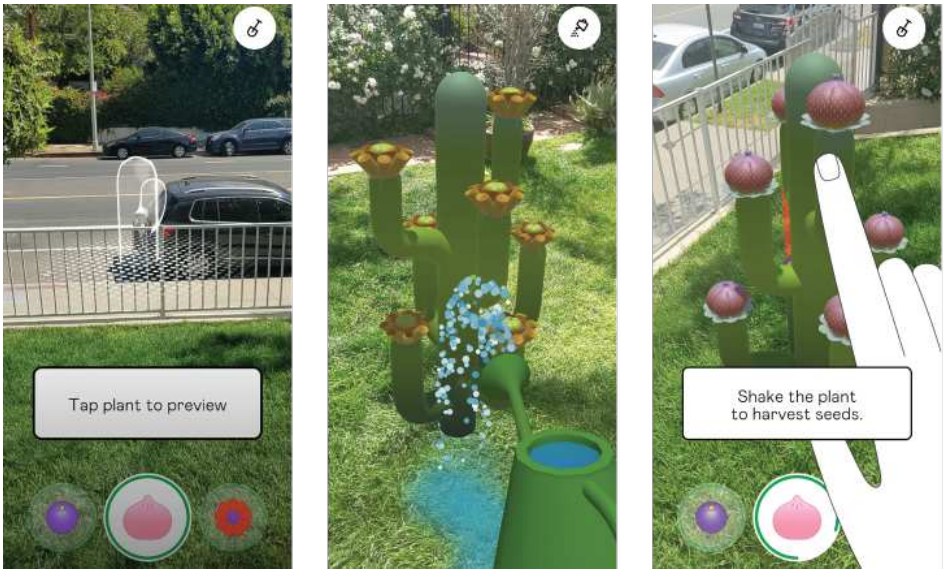

Figure 2.37: Screenshots from the demo application Pocket Garden [76]

Meshy

Using Meshy [77], we can generate some nice models2.38 for free, although we are limited to 200 credits. Generating one model costs either 10 or 20 credits, depending on the model version used. Meshy also allows for the generation of textures, 3D models from text or images, or voxel volume models from text. Meshy has introduced a new beta version of its generative model for 3D modeling that offers much improved texture quality and precision. The beta model enables the generation of a preview model and additionally a refine model. The differences between these generated models can be seen in the Figure 2.39 However, I believe that the 'imagination' of this model is not as good as that of the alpha model. One of the greatest advantages of this tool is the API access it provides with the free subscription.

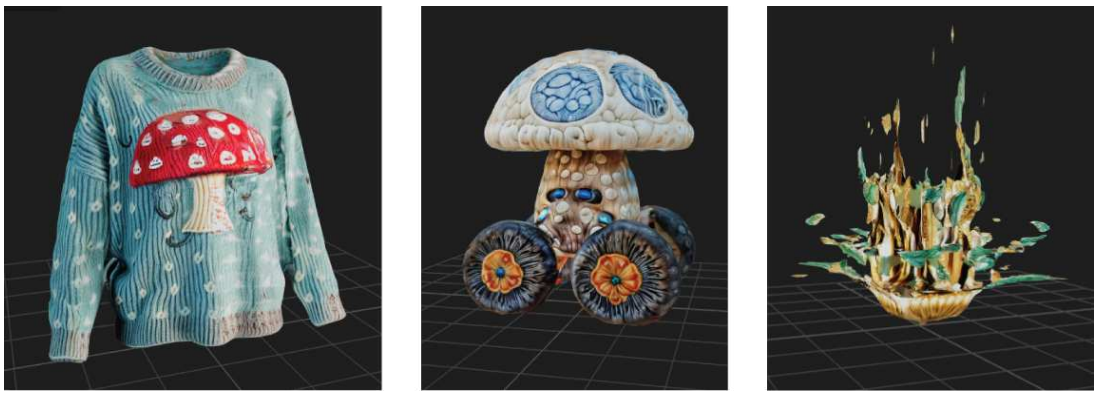

Figure 2.38: Example models generated by Meshy from given text prompts, from left to right: a cozy sweater with a stylish mushroom pattern, a robot mushroom, and a plant symbolizing the Olympic Games

Stable Diffusion

Stable Diffusion [78] is well-known for its text-to-image diffusion model that generates photo-realistic images from any text input. In addition to this, it offers a Text-to-3D API, which enables the creation of 3D assets based on text descriptions. This feature can produce a diverse range of 3D models, including

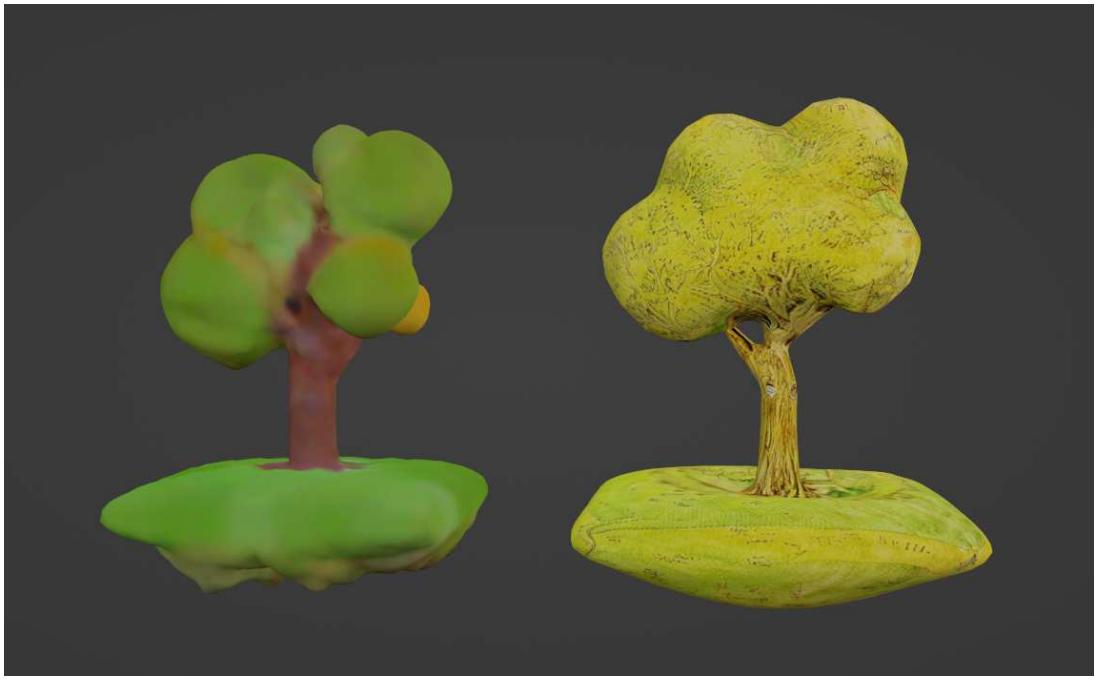

Figure 2.39: An illustration showing the differences between a preview and a refine models generated using the Meshy's beta generative model

simple objects, characters, and entire scenes. The API also supports the creation of 3D models from images (Figure 2.40). However, access to this model is not free; it requires a subscription. The cost for the basic subscription is $27 per month.

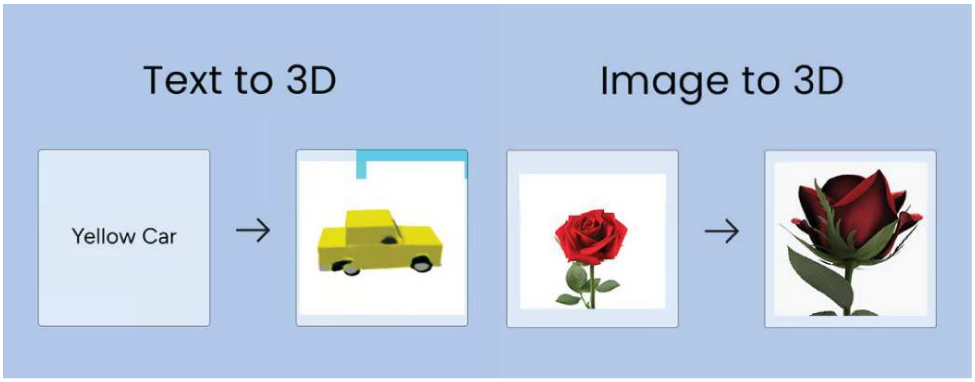

Figure 2.40: Stable diffusion AI tools for 3D model generation [78]

Stable DreamFusion

If we don't want to rely on any external service, we can use an open-source implementation of DreamFusion, introduced in Section 2.10.2. This implementation [79] uses the open-source text-to-image model Stable Diffusion instead of Google's Imagen. While the generated models 2.41 may not match the quality of other tools, they are sufficiently good for mobile applications 2.42. The project, named Stable DreamFusion, was created by Jiaxiang Tang and is accessible on GitHub. Due to the necessity for specific packages and their ongoing updates, modifications to the code are required, which can be a significant challenge for inexperienced users. For advanced users with knowledge of neural networks, this project

serves as an excellent starting point for creating your own text-to-3D generative model.

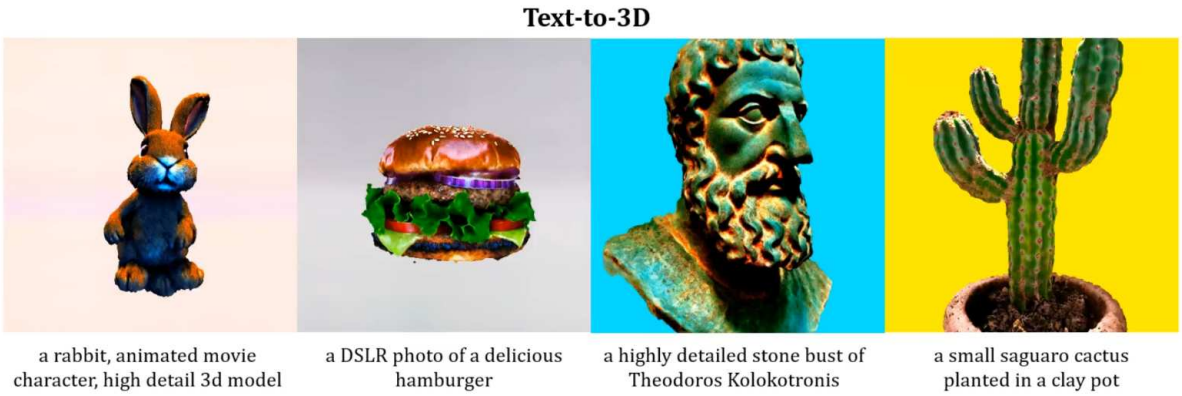

Figure 2.41: Demonstration of 3D models generated by Stable DreamFusion [79]

Figure 2.42: Generated models with a given prompt (from left to right): a huge blossoming flower, a small blossoming plant with square leaves, a tree with square leaves and pink blossoms (NeRF resolution 64, 128, 128 + 1000 iterations)

2.13 Summary

Augmented Reality utilizes localization features and visual enhancement techniques to smoothly integrate digital content with the real world. These technologies enable AR applications to accurately position and realistically render virtual objects based on scene understanding, which interprets objects, surfaces, and spatial relationships. To efficiently develop these sophisticated AR applications, developers use various frameworks that provide essential tools and modules.

Research in text-to-3D generation is rapidly advancing and continuously changing. While all presented researches provide intriguing methods of 3D generating, they represent just a fraction of the broader challenge in creating realistic and efficient 3D models from text descriptions. As algorithmic enhancements progress and computational power expands, I expect to witness more significant breakthroughs in this fascinating area of artificial intelligence. Analysis of generative models also confirms the feasibility of generating 2D or 3D digital content for various applications.

The development of fields such as AR applications and generative AI tools is experiencing significant growth. It is very important to thoroughly investigate the functionality of each tool because their descriptions can be misleading. I've presented only a few selected applications that might inspire or influence the thesis. Every listed application comes with a pre-selected set of 3D assets and does not generate any additional 3D content. I believe the thesis could introduce a unique AR application featuring AI-generated models.

CHAPTER 3

Design

The primary objective of the thesis is to develop an Augmented Reality (AR) application for smartphones that allows users to create models using generative AI. It's important to note that the computational power of a mobile phone is insufficient for running generative AI models, as they require at least a mid-tier GPU and substantial memory—a requirement that is increasingly being met. Due to this limitation, the AR application and model generation are split into two components: a mobile application and a server.

The development of generative AI is advancing rapidly, with new AI models emerging every month. To maintain flexibility in the architecture and avoid being tied to a specific 3D generation process, I decided to separate the communication features from the computation features of the server. Additionally, it is necessary to store information about 3D models in a database. In summary, I designed the system, illustrated in the Figure 3.1, with four components that need to communicate with each other. These components will be described in the following sections.

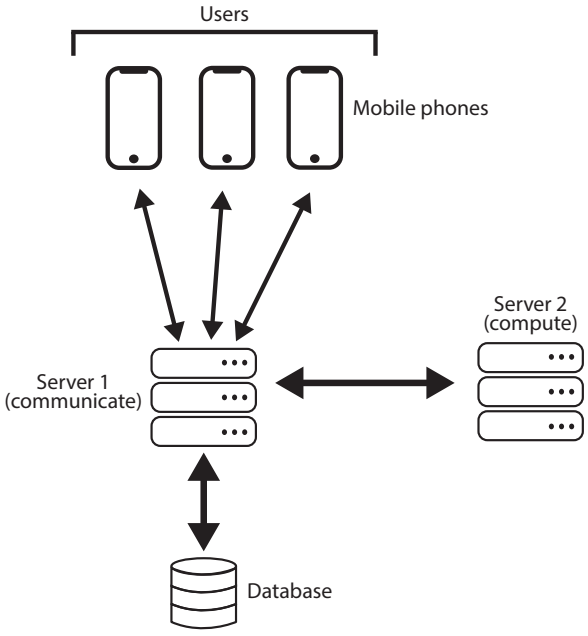

Figure 3.1: A communication diagram of the system

Prior to discussing the individual components in detail, it is essential to first outline the functional and non-functional requirements of the system.

Computational server's requirements:

- independence from generative models
- get data for given location
- create text prompts from available data
- output of generative model
- let know communication server about newly created model

Communication server's requirements:

- access the database
- send information from the database
- create a new user
- login a user
- logout a user
- create a new object
- submit a model
- get a list of the nearest object at the specific location
- request the computation server to generate a new model

Application's requirements:

- intuitive
- user authentication: register, login, and logout users
- creating a object
- loading objects around the user
- viewing other users' objects
- downloading models from the server
- updating the models at runtime
- updating objects' information
- viewing the models' details
- rating models
- updating settings
- responding to the visitor count

3.1 Communication Server and Database

The primary functions of the server are to facilitate the connection between the application Section 3.3 and the computational component Section 3.2, and to maintain the data within the database. The database stores all information from AR applications and the computational server. Regardless of who created the model, ensuring visibility of all objects, imposes certain requirements for communication and database operations. The database is designed to store information about each user, the location of the plants, and their corresponding models. To meet these requirements, I have developed a draft of the database model shown in Figure 3.2.

The domain model comprises four tables. The first table, the User table, represents users logged into the application. The Plant table contains information about each plant, such as versions of the generated model, and metrics like the number of visitors, likes, and dislikes. The plant's location is captured in the Anchor table, which stores GPS coordinates including latitude, longitude, and altitude. Additionally, using the VPS system, I am able to record model rotations, which are also stored in the Anchor table. The final table, the 3DModel table, represents the generated models within the domain model. The server is responsible for providing information about each table, but it also processes and filters database data to meet specific requirements. For example, it should be capable of generating a list of objects within a specified distance.

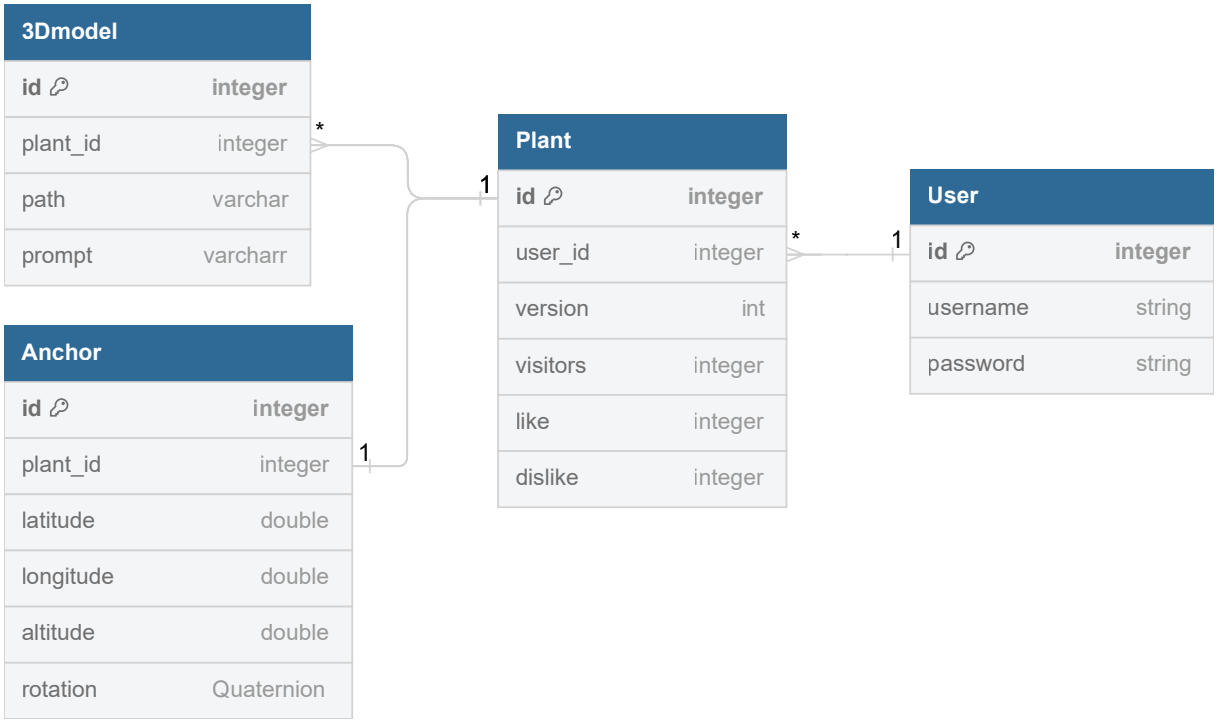

Figure 3.2: A draft of the domain diagram

3.2 Computation Server

The computation server processes the location data to generate the output model. Based on the analysis Section 2.7, I designed a generative component whose core is the conversion from the text-to-3D model. Initially, however, it was necessary to develop a method for creating text prompts that correspond to the user's location.

The process of creating prompts is managed by the prompt engineering method, established to optimize interactions with AI models. The quality of the output is directly influenced by the input prompt, making the creation of effective prompts a critical element of the system. The creation of text prompts is driven by information retrieved from GPS coordinates Section 2.11. To craft the most accurate textual description of a user's location, GPS is used to obtain detailed address data, including the country, city, suburb, street, and nearby points of interest, as well as street views. These street views are processed through an image-to-text model to provide a concise description of the visible surroundings, capturing dominant colour schemes and significant elements. The address information is further utilized in other models, such as the text expansion model and the question-answer model. The text expansion model can generate imaginative stories or sentences related to the specific address. Meanwhile, the question-answer model can provide insights into the types of plants that can thrive at the location, the local vegetation, the typical art styles of the area, and more. Ultimately, I got lists of texts and words (Figure 3.3), which are combined into one cohesive text prompt. The final text prompt is assembled by random selections from each textual output.

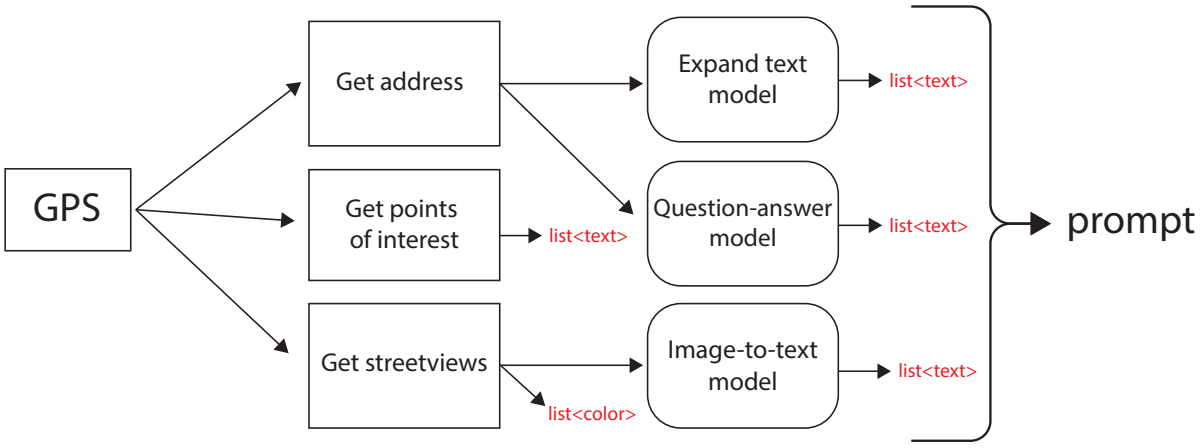

Figure 3.3: A design process of creating text prompt from GPS information

3.3 Mobile Phone Application

The mobile phone application serves as the sole ”access point” for the entire system for users. Therefore, it is crucial to develop an intuitive application where users can create new objects and view all generated models. This involves satisfying all application requirements, including login, logout, loading objects, downloading models. Additionally, the application must accurately determine the location of placed objects using the VPS system Section 2.2.5. While communication between the application and the server is crucial, it's not solely about connectivity. Effective updates are essential, as they enhance the functionality provided by this communication.

One of the use cases entails finding nearby objects and accessing their details. The process involves several steps necessary to complete this task, as illustrated in the activity diagram Figure 3.4. Initially, the user launches the application and authenticates. Upon successful authentication, the user scans the surroundings to detect any models or creates a new one. If a model is spotted, the user selects it and interacts with the model to explore its detailed attributes.

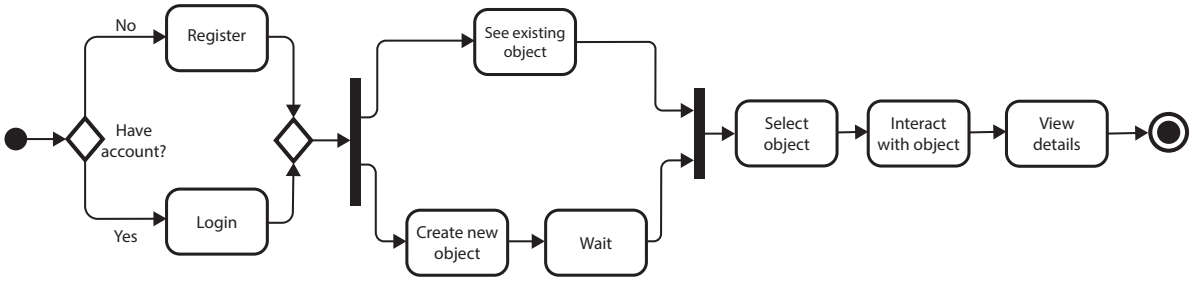

Figure 3.4: The application's activity diagram illustrates the use case of viewing details of a generated model, the process begins with the user launching the application and concludes with viewing the details of the selected model

CHAPTER 4

Implementation

The previous chapter 3 provided a draft of the system, which was divided into four components. Similarly, this chapter will also be organized into sections describing each component in detail.

4.1 Communication Server

The Communication Server is a REST API server that enables client-server communication. Clients send requests to the server, which then responds back to the clients. Having personal knowledge of the Python programming language, I chose to implement the REST API server using the open-source Django web framework, with the aid of the Django REST framework. Several endpoints were implemented, primarily utilizing the HTTP methods GET and POST. The GET method requests data from the server without including additional information in its body. In contrast, the POST method sends data to the server in its body, which can be used for creating, updating, or even retrieving data that requires more information than can be included in the HTTP address's query.
Endpoints:

1. plants/[primary-key]
2. plants/like/[primary-key]
3. plants/dislike/[primary-key]
4. plants/visit/[primary-key]
5. users/[primary-key]
6. models/[primary-key]
7. anchors/[primary-key]
8. relogin/
9. logout/
10. models/
11. anchors/
12. nearest/
13. register/
14. login/

Endpoints 1 through 8 are implemented using the GET method, while the remaining endpoints utilize the POST method. Endpoints 1, 5, and 7 specifically request attributes of an object, using a primary key. Endpoint 6 is designed to retrieve the actual 3D model; this request downloads the model from the server. For incrementing certain plant's attributes, endpoints 3, 4, and 5 facilitate this via GET requests.

POST methods are used to create new objects for each model type. However, the "nearest" endpoint is an exception. It is the only POST request that does not create any new object but instead requests a list of the nearest objects within a specified distance and location. It also retrieves information about the plant model's version, which is useful for loading and updating objects. Notably, this POST request could alternatively be structured as a GET request with the parameters included in the URL query.

The entire system uses Token Authentication. Communication between the Application and the Communication Server is secured by the Django REST framework's Token Authentication. Since the

servers run on the same device, it is not necessary to create a sophisticated authentication system. A static token, which is generated before each start, is saved in the configuration file on the device. If the servers were to run on different machines, they would use the same authentication method as that used for communication with the application. The reason why the servers run on the same device is explained in the Section 4.3. Currently, the server is running on the CTU's computer, accessible via the public IP address 147.32.81.228 on port 6363.

4.2 Database

The database is an integral part of the Communication Server, which is implemented using the Django REST framework. Django's default database is SQLite3. SQLite is a simple, lightweight database typically used in small to medium-sized projects. It does not require a separate server process or system to operate, as the SQLite library accesses its storage files directly. Additionally, it does not need any configuration and is designed to be easy to install and use. For my purposes, this database is sufficient. In the Section 3.1, there is a design of the database from which the final structure is derived, as shown in the Figure 4.1. The database includes four tables, each enriched with several attributes. The "User" table, titled "CustomUser," is derived from Django's "AbstractUser" model. Let's explore the relationships between the tables and the logic behind them, as also depicted in the Figure 4.1. Each user can create several plants, but each plant is assigned to only one location, represented by the "Anchor" table. A single plant can be associated with multiple 3D models. For instance, if the location's attendance is high, a new model may be generated, replacing the previous one.

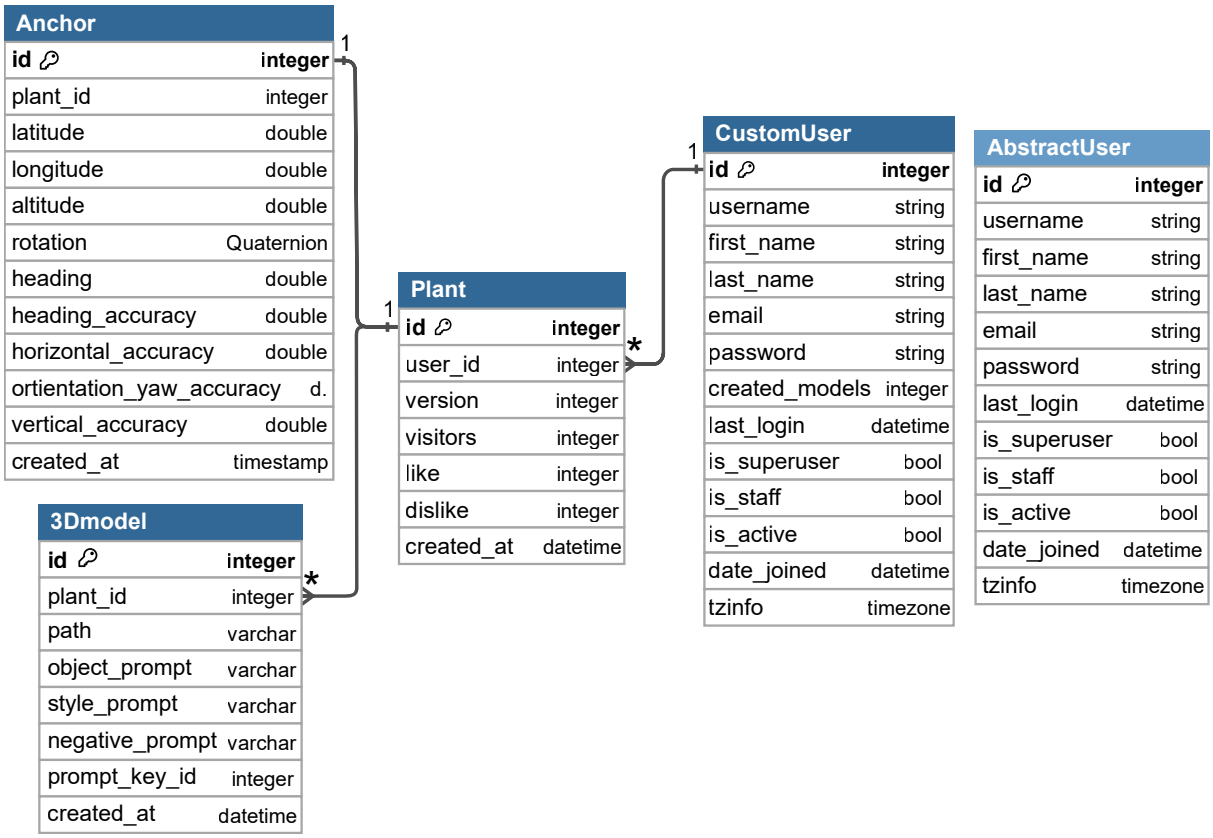

Figure 4.1: The final form of a domain diagram

4.3 Computation Server

The Computation REST API server employs the same web framework as the Communication Server, as detailed in Section 4.1. It features two primary endpoints: ”/anchor/” and ”/regenerate/.” Initially, the ”/anchor/” endpoint generates a text prompt utilizing location information alongside a three-dimensional model. Subsequently, the ”/regenerate/” endpoint is utilized to generate a new model based on the provided text prompt.

The Django REST framework does not support multitasking processing. Due to extended generation times while the server responds to other requests, it was necessary to implement a solution for multitasking. Celery, one of the most commonly used task-scheduling frameworks, was chosen. This open-source framework manages asynchronous task queues through distributed message passing. Communication between the server and Celery is facilitated by a message broker, with Redis being a recommended option. However, Redis only operates on Linux distributions, necessitating the activation of the Linux subsystem on Windows to install and run Redis.

4.3.1 Text Prompt Generation

The process of generating text prompts, previously introduced in Section 3.2 and depicted in Figure 3.3, begins with the input of GPS location data to obtain an address, a list of points of interest, and street views. I primarily use Google Cloud Services: the Geocoding API [80] for addresses, the Places API [81] for points of interest, and the Street View Static API [82] for getting street views. From these street views, I extract dominant colors, and create a list of colors. These color values are converted to textual representations using the Python package Webcolors [83]. To access all mentioned APIs, it is needed to create Google Cloud account with credentials. Then you have to enable selected API and paste provided API key in the servers configuration file.

Additionally, street views are processed using the BLIP Image Captioning large model, available on the HuggingFace website [84]. This model, based on the theory described in a paper [85] in 2022, converts a set of images into textual descriptions of the user’s surroundings. Beyond using the pre-trained BLIP model, I also employ the GPT-2 model for generating textual content. Despite the newer GPT-3 model being known for its capabilities in chatbots, due to its requirement for significant computational power and memory, I opted for the GPT-2 model. Specifically, I use the full GPT-2 XL, which contains 1.5 billion parameters and is also available on the HuggingFace website. For comparison, the GPT-3 model features approximately 175 billion parameters, while the latest GPT-4 has escalated to one trillion parameters. Moreover, Google’s Gemini model boasts an unprecedented 175 trillion parameters.

For expanding the input sentence structured as ”On [place] there is a [plant] which,” the ’place’ is determined using the address from the Geocoding API, and ’plant’ is randomly selected from a set of static plant names, augmented with words obtained from the question-answer model. This model, known as Flan-T5-XL, is a derivative of the pre-trained T5 model, fine-tuned to enhance zero-shot and few-shot performance. It was utilized to respond to several static questions that were completed by the ’place,’ thus providing the address again. Static questions which are input to Flan-T5 model:

- What can grow up on [place]?
- Which plant can grow on [place]?

- Which shape can be on [place]?
- Which style can be on [place]?
- Which art style can be on [place]?

The last trained pipeline used in prompt generation is en_core_web_sm. It is designed to extract nouns from terms and sentences. This pipeline is utilized for processing larger strings, such as those expanded by GPT-2. The final prompt is created by randomly combining all the strings from the formed lists. The style and negative prompts are randomly created from static set of words. For summarise all used tools for text prompt generation, there is an updated design image 4.2. You may notice that the computational power required is proportional to the use of generative models.

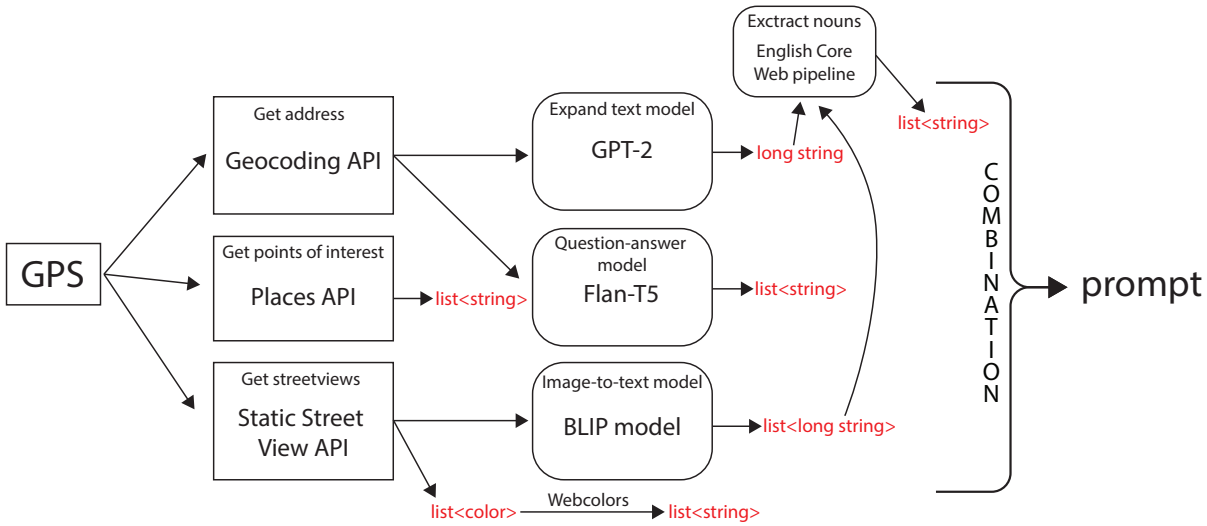

Figure 4.2: The illustration of used tools for text prompt generation

4.3.2 3D Model Generation

The most power-intensive generative model in the thesis is the text-to-3D model. As previously mentioned in Section 2.12.2, I attempted to run the open-source project known as Stable DreamFusion. This implementation requires a good graphics card with at least 8 GB of memory. The project is also accessible via Google Colab; however, I encountered numerous limitations that prevented the completion of any tasks within the Colab environment. Specifically, the generation time for one model is approximately 40 minutes, and due to this extended duration, the process in Google Colab was frequently interrupted because of GPU usage limitations. Consequently, I began to consider the feasibility of a server equipped with substantial GPU power. This consideration led to many limitations, and ultimately, I abandoned the idea of hosting the server on any virtual machine. Further details regarding this decision are discussed in the following paragraph.

Then I tried to generate the model locally on a computer provided by CTU University. As noted, the generation process takes about 40 minutes. The results of these efforts are documented in the Section 2.12.2. However, the last successful creation was in May 2023. Since then, there have been numerous updates in the required libraries and packages, which have introduced several challenges with

specific dependencies. These issues have been reported, but bugs are still persist.

One potential solution could involve developing my own model; however, this task exceeds my current expertise, could potentially occupy an entire thesis, and faces the significant hurdle of availability appropriate training datasets.

The last possibility was to use already an existing service that facilitates text-to-3D generation. The final solution is using Meshy 2.12.2, which offers a trial subscription providing 200 credits per month, along with API access. Each 3D model generation consumes 23 credits, enabling the creation of approximately nine models per month at no cost. Currently, this is the most advantageous available free subscription. For more frequent use, the PRO subscription is priced at $20 per month, or $16 per month when billed annually, allowing for the generation of up to 43 models monthly.

A request to the Meshy API includes a JSON object containing a created object text prompt, a style prompt, and a negative prompt. The response provides the task ID, which generates a preview model. Subsequently, I request a refined model, referencing the preview model's task ID. The response to this request yields the second task ID where the final model is created. The requesting process has become more complex with the introduction of a new generative model called beta. Once the final model is completed, the textured model is downloaded. According to Meshy API documentation [86], it is possible to obtain the model in formats such as FBX, OBJ with MTL, GLB, and USDZ. The beta generative model produces textures in only one base color, whereas the previous alpha model also provided metallic, normal, and roughness textures. However, the texture quality produced by the beta model is exceptionally high and includes many details.

4.3.3 Servers hosting

Hosting servers on a cloud platform is feasible but requires financial resources. Typically, the monthly cost of server hosting with the least powerful GPU offered and a suitable amount of memory is approximately $300. However, the price can be reduced if the service is purchased for long-term use; for example, a virtual machine on Google Cloud would cost approximately $150 [87] per month under such a plan. On the other hand, it is possible to test various short-term projects and utilize the features provided by Google Cloud. The Google Cloud Free Program offers 90 days of free usage with a $300 budget. Similar services are offered by other competing companies such as IBM [88], Amazon Web Services [89], and Microsoft Azure [90].

After my Google Cloud trial period ended, I approached CTU University to inquire if it would be possible to host servers on their equipment, and they provided me with a computer for this purpose. Subsequently, my implementation focused on constructing one computing server with a local IP address and one communication server with a public IP address on the same device. The hardware specifications of the device are as follows: an Intel Core i9-10900x processor at 3.70GHz, an NVIDIA Quadro RTX 5000 graphics card, and 64GB of RAM.

Hosting two servers on a single device offers several advantages. For instance, it eliminates the need to transfer generated models from one server to another. Generated models are saved locally, and the servers reference these models using their file paths. The size of each generated model is approximately 15 MB, which means that the system does not require an enormous amount of memory to store the

models.

4.4 Application

The initial concept for implementing the AR application involved leveraging the open-source project Pocket Garden, which is developed in Unity, and adapting it into the designed application. However, transforming the Pocket Garden system into my system proved to be overly complex and time-consuming. It forced me to opt to start from scratch instead. The entire application was developed in Unity, which facilitates the creation of multi-platform applications. My focus, however, was on development for the Android platform.

4.4.1 Unity Project

First, I initiated a new Unity project using Unity's AR application template, version 2021.3.10. This startup template includes several of Unity's packages tailored for augmented reality, along with other essential packages needed to construct a basic scene containing a camera and direct lighting. Utilizing a template accelerates the development process compared to starting with an empty project. Once the project was established, it became necessary to install additional critical packages such as **AR Foundation** and **ARCore XR Plugin** via the Package Manager. In iOS development, ARCore is substituted by ARKit. These packages offer numerous features that enhance AR applications, including place detection, image tracking, and raycast management.

At this stage, I can add the two most important components, without which running an AR application is not possible. The first is the **AR Session**. The AR Session is a component that controls the lifecycle of an AR experience by enabling or disabling AR functionality on the target platform. It also verifies the availability of AR capabilities on the device. By default, this includes the ARInputManager, which enables world tracking. The second key component is the **AR Session Origin**, which in AR Foundation 5 is now referred to as XR Origin. The AR Session Origin's purpose is to transform session space (or device space, where the device's coordinates are (0, 0, 0)) into Unity world space. This component incorporates a Camera component.

Localization

Global localization with VPS is facilitated by the **ARCore Geospatial API**. To utilize this feature, it is necessary to install the **ARCore Extensions** package, which I imported from a Git repository [91]. Once ARCore Extensions are installed, this component is added to the scene. It is linked to the AR Session, AR Session Origin, and AR Camera, which have already been inserted. The last two properties are configuration files for the ARCore Extensions itself and the AR Camera. We can select the features we wish to use in the project; for instance, semantics and street geometry modes are enabled here. My configuration files activate Geospatial mode. The AR Session Origin maps the model object to Unity space coordinates, but for global localization, it is necessary to obtain the model's global coordinates in the form of GPS data. This functionality is incorporated in the **AR Earth Manager**, which is integrated into the ARCore Extension component. The AR Earth Manager can convert Unity's Pose into a GeospatialPose and vice versa. Now, I can easily create a new Anchor, which is attached to the model and manages the model's global location. The basic anchor creation is shown in a code block 1.

Algorithm 1 Creating Anchor

```
1: procedure CREATEANCHOR(Pose newPose, GameObject model)
2:     using ManagerEx = ARAnchorManagerExtensions
3:     GeospatialPose geoPose = AREarthManager.Convert(newPose)
4:     ARGeospatialAnchor newAnchor = ManagerEx.AddAnchor(ARAnchorManager, geoPose)
5:     GameObject newObject = Instantiate(model, newAnchor.gameObject.transform)
6:     newObject.transform.parent = newAnchor.gameObject.transform
```

To ensure a successful build and run the application without any issues, it is necessary to modify the project settings. Within the XR Plug-In Management tab, the plugin provider must be selected; in this case, I selected the ARCore option. The sub-tab for ARCore Extensions is dedicated to configuring the Geospatial API settings. Here, the Geospatial API key from Google Cloud is entered, and the features to be used are selected. I chose the Geospatial feature. In the Player Settings, in addition to setting application preferences such as orientation, icon, and name, it is crucial to adjust the Other Settings. Here, all Graphics APIs options are removed except for OpenGL. It is also essential to select the minimum API level, scripting backend, and target architecture for the project. For this project, the minimum API level is set to 28, the scripting backend to IL2CPP, and the target architecture to ARM64.

Visualization

The generated models are downloaded from the server and loaded into the scene at runtime. Since Unity can only import prefab models at runtime, alternative solutions were necessary. **Siccity's GLTFUtility**, a straightforward open-source GLTF importer for Unity, facilitates the runtime import of GLTF and GLB files. However, for the import to function correctly, all its shaders must be added to the 'Always Included Shaders' list under the 'Graphics' tab in the Project Settings.

To enhance the appearance of virtual objects, I utilized the AR Occlusion Manager, which adjusts the blending of digital content with the real-world view by using depth information. This information can be provided by a depth sensor (e.g., LiDAR) or estimated. The manager provides environment depth information in three modes, which vary in accuracy and computation time: Fast, Medium, and Best. I set it to the Medium mode. However, if the models in the scene are far from the user, the occlusion effect may render them invisible. This behavior hinders the user's ability to search for surrounding models. To address this, I added the option to turn the occlusion effect on and off within the application settings.

The AR Camera Manager provides a light estimation option. It is possible to estimate the light direction, its intensity, and both the color and intensity of ambient light. Ambient Spherical Harmonics can also be estimated, although this is the only feature I have disabled. All estimated information is used in Unity's HDR Light Estimation script, which is attached to the directional light object. The only consistently reliable estimation is that of ambient light. The other estimated light parameters are not available most of the time.

To make the models appear more realistic and 'anchored' to the ground, I added shadows. These shadows are actually planes equipped with a transparent material that uses a shader to modify its texture according to objects' shape. This shader was published in the article [92] in the STYLY Magazine.

Structure

This paragraph brings an overall overview of how the application works. The functionality is split into four main components (managers), see Figure 4.3. The managers are implemented as singletons, that they have only one instance, which exists entire application's lifecycle. The UI Manager is responsible for an application's user interface and its interaction with user. The second manager is Placement Manager which manages placement target's behavior and place and initialize all models in the scene. It primary coorporates with Geo Manager which converts Unity's coordination system to Global Positioning System (GPS). Geo Manager starts location services, checks localization's accuracy, and creates anchors. The last manager is designed for communication with server. It cares about independent updating corutine which asks for nearest objects, compares versions of models on the server and in the application, decides which model in needed to be downloaded and which will be loaded from memory. It also updates all model's information in the scene. Update process is launched in specified interval which can be changed by user in the application's settings options. User can also change the range of distance of loaded anchors. The default value of update interval is two minutes and default range of distance is one kilometer.

All managers collaborate with each other and together form the core of the application. Additionally, I developed several scripts, most of which handle UI interactions.

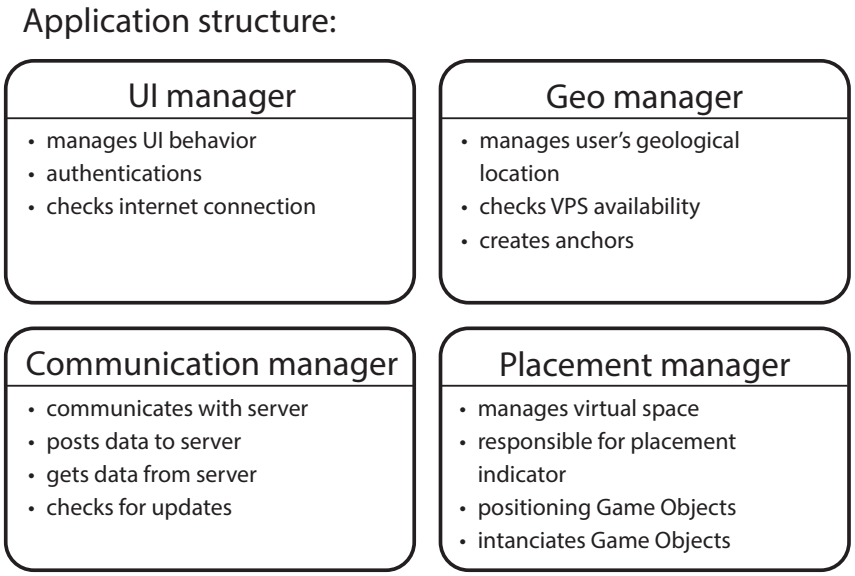

Figure 4.3: The application's structure showing each manager and their roles

4.5 Communication

I introduced the details of each part of the system, including the communication server, database, computation server, and application. To fully understand the system, I will explain their communication, review the tools used, and present the data flow diagram.

The application is developed using the Unity game engine and runs on Android mobile phones. It

sends requests to a communication server implemented with the Django REST API framework. This server's primary role is to facilitate communication between the application, the computation server, and manage data saved to an SQLite database. The computation server receives tasks from the communication server and processes them asynchronously using the Celery framework. These tasks are passed to Celery via the Redis message broker. Celery then generates the text prompt and sends it to the Meshy API. Once Meshy completes model generation, Celery downloads the textured model and sends the model's file path to the communication server, which updates the database with information about the new model. This information is illustrated in Figure 4.4.

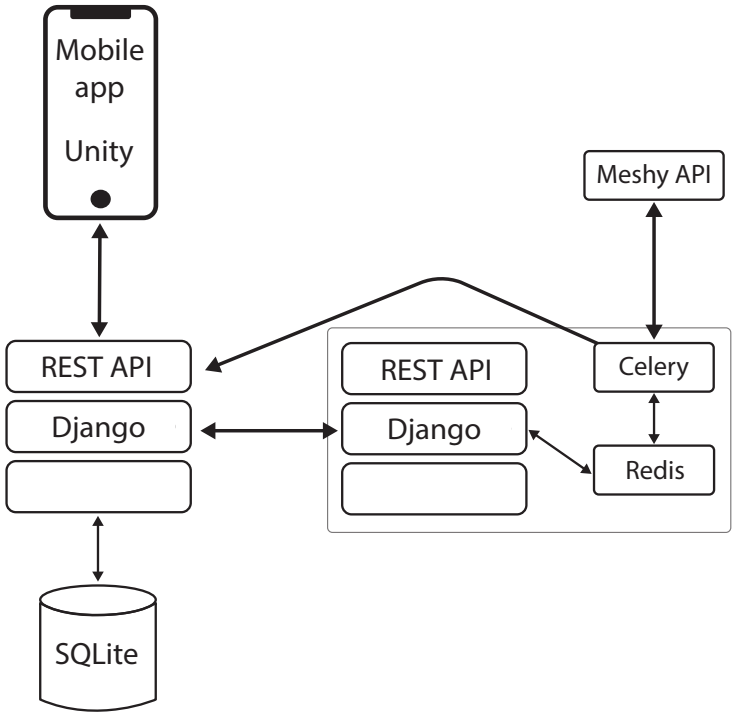

Figure 4.4: The illustration of used tools in the system

Firstly, I will demonstrate the process of creating a new model, beginning when the user places a new plant. This process is depicted in the green section of Figure 4.5. I will provide comments on each step to enhance comprehension. For simplicity, I will use the following abbreviations: Communication Server = S1, Computation Server = S2, Application = APP, and Database = DB.

1. The user creates a new plant; a new anchor is created and its location is sent to S1
2. S1 saves the location in the DB
3. DB creates a new location entity and returns the plant ID to S1
4. S1 sends a request to S2, including the location and plant ID
5. S1 sends the plant ID back to the application
6. S2 generates a model and sends it along with a prompt to S1
7. S1 stores the model and the prompt in the DB

The described process creates a prefab in the application, generates a new model, and then saves it in the database of the communication server. The generated model is loaded into the user's scene when the application queries the communication server for the nearest objects. The communication server identifies the nearest object, extracts its plant ID and model's version, compiles them into a list, and sends

it back to the application. The application then compares this list with the plant IDs of models already loaded and also generates a list of plant IDs from objects with differing model versions. Subsequently, the application requests the new model using the plant ID. The communication server consults the database, retrieves the latest model, and dispatches it to the application. This process is also illustrated in the right side of Figure 4.5.

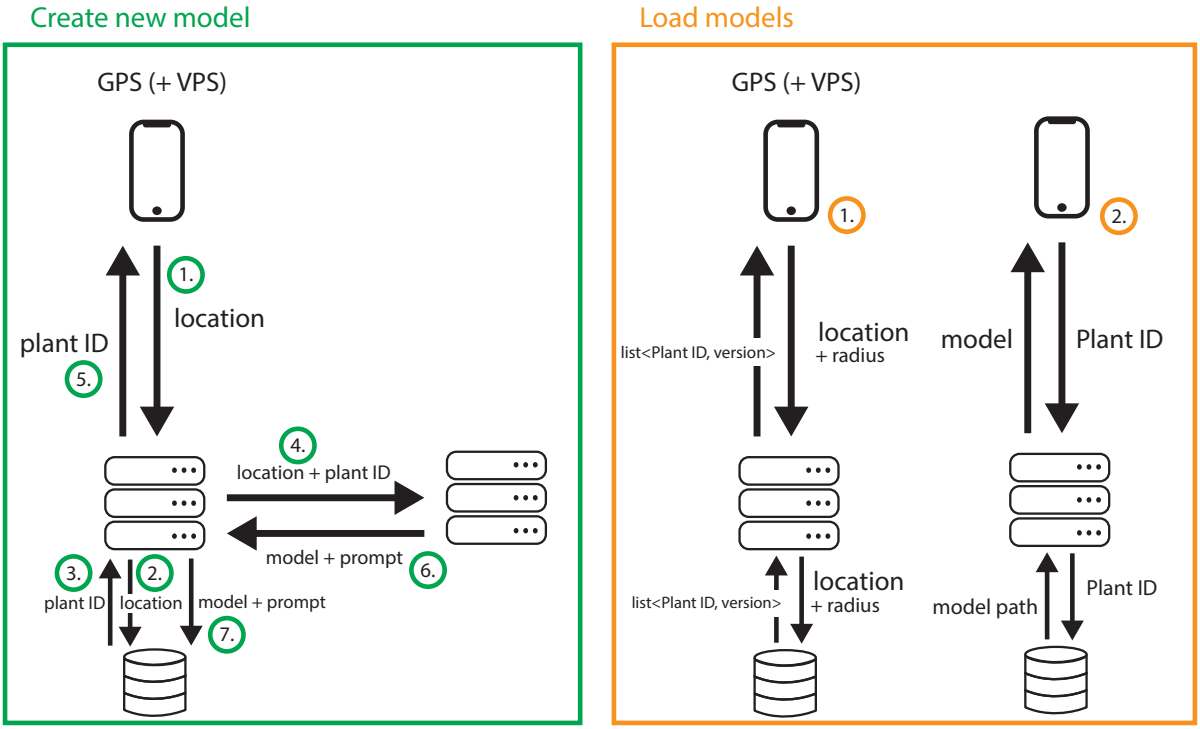

Figure 4.5: The data flow diagram of the system, the left part shows the process of a model creation, the right part illustrates the model loading process

CHAPTER 5

Testing

This chapter is organized into three sections, each focusing on a different aspect of the system's functionality. The first section addresses the testing of Localization features, followed by an evaluation of Visual Enhancements. The final section is dedicated to User Testing, which assesses the system's performance as a whole.

5.1 Localization testing

This test evaluates the system's localization capabilities, specifically the use of the Visual Positioning System (VPS) as outlined in Section 2.2.5. The VPS, provided by Google wherever Street View is available, is generally accessible. However, it is crucial to monitor its accuracy regularly, as the precision of location data can vary.

The availability of the VPS was tested in a part of Prague, with results presented in the Figure 5.1. Key findings are marked by a blue line, highlighting areas where Google Street View is available but VPS location accuracy is low. This is typically observed in areas with dense vegetation, such as parks.

The localization accuracy was also tested by examining the precision with which the model is reloaded at a specific location. The process involved placing the prefab plant at a designated spot, resetting the system, and then reloading the model to measure the distance between the original and reloaded locations. A cross mark was drawn on the pedestrian path where the model was placed, and the range of distances was highlighted in twenty-centimeter increments, as visualized in the Figure 5.2.

I selected two locations that differ in their surrounding environments. The first location is near Nádraží Podbaba in a parking area. This site is surrounded by buildings, and according to Google, the street geometry appears quite accurate, as shown in the Figure 5.3.

The prefab was placed on the cross mark as shown in the Figure 5.4. I then launched the application and, during the initialization of VPS, pointed the camera at four different points of view, as seen in the Figure 5.5. These views significantly influenced the VPS accuracy. When the camera's viewpoint matched that of the model's placement, the location error was approximately one meter. The error was slightly less when the camera pointed at a building across the road. The best accuracy was achieved when the camera faced an apartment building behind the parking area. This building, distinct from its surroundings, features many visible edges and corners, contributing to its identification. According to the Google 3D street map, this building has great geometry and precise textures. In this case, the error

Figure 5.1: A map depicting VPS availability in the selected area: green lines indicate locations where VPS is available, and blue lines highlight areas where VPS accuracy is poor despite the presence of Google Street View

distance was approximately twenty centimeters, whereas typically, the location error is around fifty centimeters.

The second location is situated in an open space near Stadium Strahov. The exact spot is depicted in the Figure 5.6, which also shows the environmental geometry and the availability of Google Street View. This location was deliberately chosen due to the high amount of vegetation surrounding it. Prior to testing, I anticipated a higher distance error compared to the first test, and this expectation was indeed confirmed. As indicated by the views in the Figure 5.7, the minimum recorded location error was 90 centimeters. During the initialization of VPS with a view at a corner of the stadium, the prefab was placed more than two meters away from the original location.

5.2 Visual Enhancements testing

This section, I will put the visual enhancement, that was mentioned in the Section 4.4.1, to the test. I describe the effects added to the project to enhance the visual appearance of the digital content. The most immediately noticeable enhancement is the implementation of light estimation. Light estimation is closely tied to shading. Accurate light estimation of the direction and intensity of direct light results in realistically visualized shadows. I will present three examples in the Figure 5.8, demonstrating different aspects of this effect. The model on the left lacks any shadow, making it appear highly unrealistic next to a car that casts a shadow. However, the ambient light estimation integrates the models into the scene effectively. Shadows also help to anchor objects, allowing us to discern whether a model is floating or

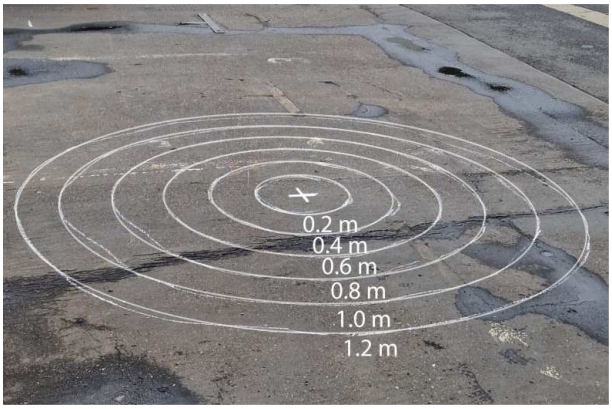

Figure 5.2: A photo showing distance ranges marked on a pedestrian pathway, starting at 0.2 meters and extending to 1.2 meters from the cross mark

Figure 5.3: The map of the place near by Nádraží Podbaba where the localization accuracy was measured. The left image shows the Google 3D street map, on the right there is a visualization of Google Street View availability

lying on the ground. The model in the middle casts a shadow, but a careful examination of the photo reveals a real shadow cast from a different direction than that of the model's shadow. This contradiction is due to incorrect directional light estimation in terms of direction and intensity. The last model appears seamlessly integrated into the real world; its shadow is oriented differently from the middle one, enhancing its realism.

All models feature excellent ambient light estimation, ensuring they blend seamlessly into the photographs. While the direction and intensity of light are sometimes accurately estimated, this consistency is not always achieved. However, the visualization of shadows provides valuable information about the position of each model, enhancing the overall realism of the scene.

Occlusion culling offers another layer of visual enhancement. Primarily focused on environmental elements, it also supports human body segmentation. This effect is demonstrated in the Figure 5.9. While the segmentation does not always yield the best results, it effectively maintains the correct order of objects within the scene. Environmental occlusion culling can create a realistic appearance, though it is subject to a distance limitation. Beyond this threshold, objects become invisible. This is illustrated

Figure 5.4: The photo of the placed prefab on the cross mark

in the Figure 5.10, where the left part of the image shows the model fully visible with occlusion turned off. The right part demonstrates what happens when the occlusion distance limit is crossed; only the nearest part of the model remains visible. I employ a middle level of occlusion culling, with the distance limitation set at approximately 3 meters.

5.3 User Testing

User testing was structured into two parts and conducted with ten users. In the first part, the tester introduces the application and guides the user through two scenarios, providing instructions for each task. As the user completes these tasks, the tester records the user's screen and notes the success rate of each task. The second part is conducted individually, where the user completes a survey reflecting on their previous experiences with the application. The application was tested on mobile phone Samsung Galaxy S22 Ultra with Android's version 13.

In the first user testing scenario, the user launches the application and completes the registration process. Following registration, the user explores the surrounding environment to search for objects. Upon finding an object, the user interacts with it and seeks a text prompt that corresponds to the generated model. The user also explores the settings and profile panels. In the second scenario, the user places a new plant and tries to find out the time of the model's generation. After logging out and shutting down the application, the user returns at the specified time for model generation, relaunches the application, logs in, and reviews the loaded models. If the generated model is identified, the user examines the models in detail and go through the individual prompt.

The results of user success in the given task are shown in the table 5.11. I grouped the users according to their phone's operating system. Android users are familiar with the interface features of their devices, such as the appearance of the keyboard, managing background applications, and checking internet connections. While this knowledge is not essential for completing the tasks, users perform better in environments they are used to. For example, Android users generally completed the registration task more successfully. The task with the lowest completion rate involved interacting with objects. Users struggled to select objects, often attempting to like a model by clicking on the like icon, which only displays the number of positive reactions. This confusion may stem from limited experience with AR applications and the poor placement of reaction icons. Some users tried to view details by zooming in on

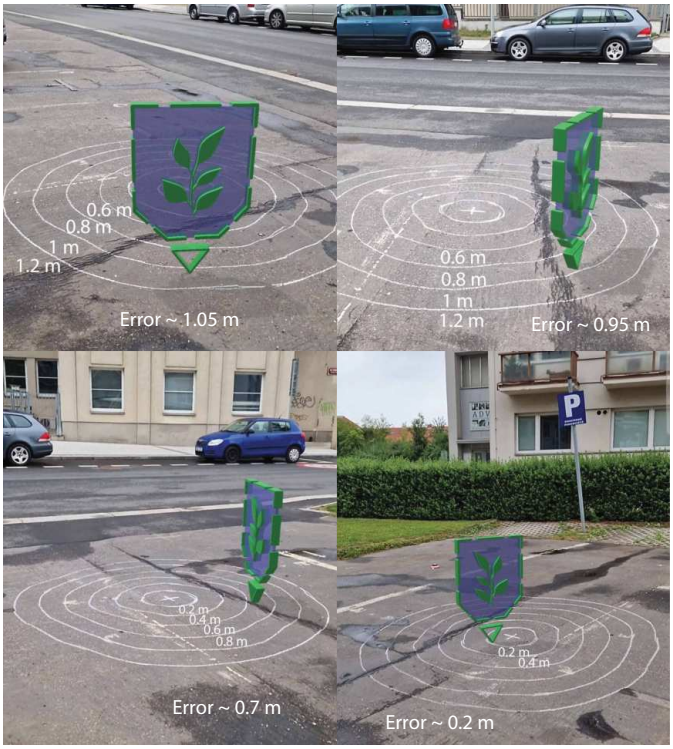

Figure 5.5: The photos of loaded prefab's location near by Nádraží Podbaba, each photo initialized the VPS from different angle

the view, rather than physically moving closer to the object. Additionally, some results were affected by technical issues, such as server shutdowns.

In summary, users completed all tasks with few small issues. During user testing, several bugs were identified and subsequently fixed. One notable bug was the ability to place many plants at the same location, which caused unexpected behavior.

Additional insights were provided through a user survey, which included the following questions:

- Did the testing proceed smoothly, and were there any problems? Did you complete all assigned tasks?
- Does the created text prompt correspond to the model's location?
- How well does the generated model match the text prompt?
- Does the generated model fit into the environment?
- How do you rate the creativity of the AI-generated model?
- How would you improve the model generation process?
- How would you enhance the application?
- Would you use this type of application, and do you see potential in it?
- How do you feel about using AI to create application content?

The results are presented in the table 5.12. A significant identified weakness was the long generation time; if users place a new plant and wish to see the model in the same session, they have to wait approximately 15 minutes, although typically the waiting time is shorter. This duration requires users to engage in other activities. In the absence of surrounded models, users may become bored, close the application,

Figure 5.6: The map of the place close to Stadium Strahov where the localization accuracy was measured. The left image shows the Google 3D street map, on the right there is a visualization of Google Street View availability

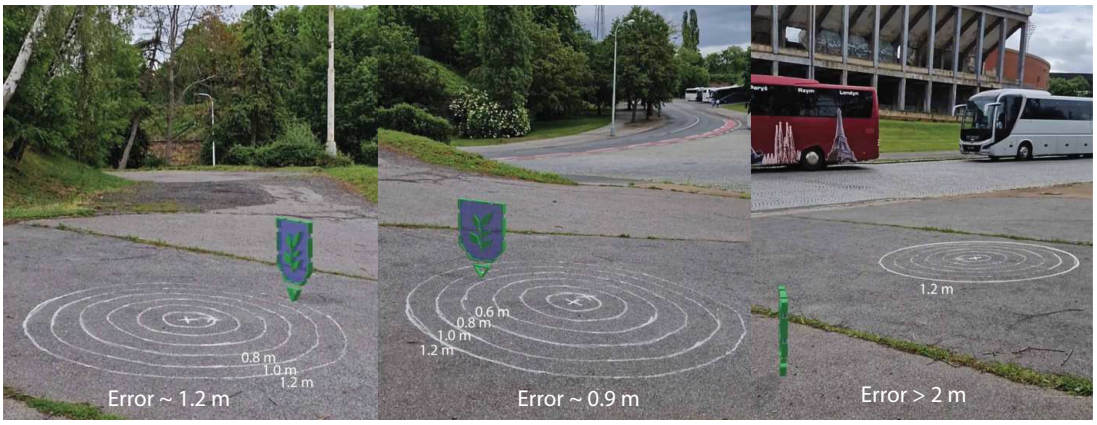

Figure 5.7: The photos of loaded prefab's location close to Stadium Strahov, each photo initialized the VPS from different angle

and leave the location.

The most informative parameters—correspondence of the prompt to the location, the model to the prompt, and the model's fit into the environment—are illustrated in individual graphs 5.13 5.14. According to the responses, 80% of users believe that the generated prompt accurately matches the location, while the remaining users think it somewhat corresponds. Half of the users confirmed that the generated model accurately reflects the prompt, while the other half felt it likely corresponds. The assessment of the model's fit into the environment is more subjective and depends on individual perspectives towards new objects. Sixty percent of participants found the model matched well within the environment. Only 10% of users considered the generated models to be overly flamboyant.

Open-ended questions provided insight into users' personal feelings about the application. Most were impressed and expressed a desire to have the application on their phones. However, some users were disappointed with the generated models, particularly when expecting a plant and seeing a statue instead, which diminished their initial excitement. Overall, the application was considered interesting and entertaining, and it could encourage people to visit new places or spend more time outdoors.

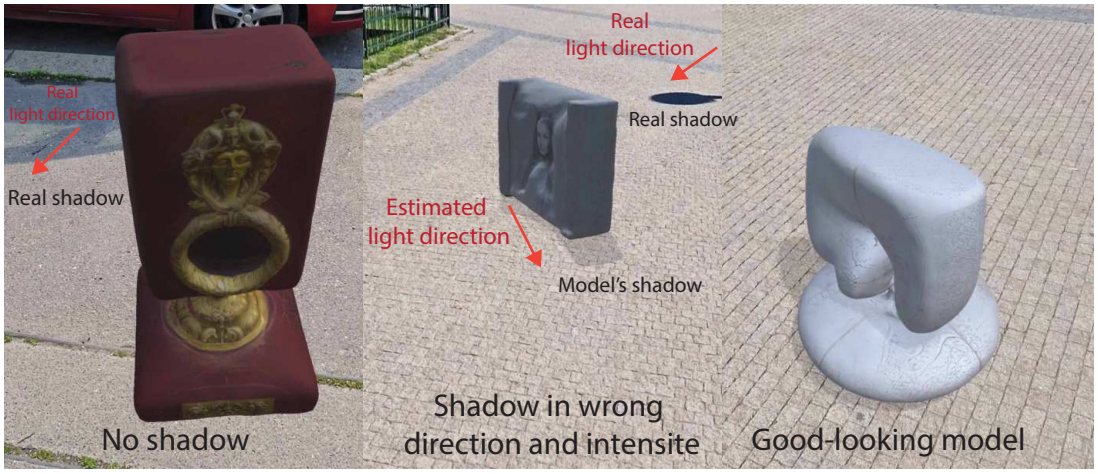

Figure 5.8: The photos of model's visual enhancement, the most left model does not have any shadow, the middle model has a wrong shadow direction, the most right model reached the best visual enhancement

Figure 5.9: A demonstration of the occlusion culling with the human body segmentation

Several improvement features were highlighted by users. The most frequently mentioned was the ability to create or add a single word to the text prompt. The system is already prepared for this feature, but the final implementation has not been completed yet. Users expressed a desire to add comments to the model, save the model to their library, or share models with friends. Enhancing the searchability of generated models could be facilitated by displaying the models' locations on a map.

Figure 5.10: A demonstration of the occlusion culling effect

	Android users						Results	iPhone users				Results	Total results
User no.	1	2	3	4	5	6		7	8	9	10		
Registration	100	50	100	100	100	100	91.67	100	75	100	75	87.50	90.00
Searching for objects	100	100	100	50	75	100	87.50	100	100	100	25	81.25	85.00
Interaction with objects	100	25	100	75	100	100	83.33	50	50	75	50	56.25	72.50
View details	100	75	100	100	100	100	95.83	100	100	100	25	81.25	90.00
Settings	75	100	75	100	100	100	91.67	100	100	100	100	100.00	95.00
Create new object	75	50	100	100	100	100	87.50	100	100	100	100	100.00	92.50
Find out the duration of generation	100	100	100	75	100	100	95.83	50	75	100	25	62.50	82.50
Log out	100	100	100	100	100	100	100.00	100	100	100	100	100.00	100.00
Looking at newly generated model	75	0	100	100	100	100	79.17	100	100	100	100	100.00	87.50

Figure 5.11: The table showing the users' success in each task

	Android users						iPhone users				Total results
User no.	1	2	3	4	5	6	7	8	9	10	
Is application intuitive?	Rather yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Rate of interaction with models?	100	100	100	75	75	100	100	100	100	100	95.00
Rating of application fluence	100	100	100	75	100	100	100	100	100	100	97.50
Rating of application updating	100	100	100	100	100	100	100	75	100	100	97.50
Rating of application loading new models	100	25	100	25	100	75	75	100	100	100	80.00
Corresponding of generated text prompt to localization, it corresponds	Yes	Yes	Yes	Rather yes	Rather yes	Yes	Yes	Yes	Yes	Yes	Yes
Corresponding of generated model to text prompt, it corresponds	Yes	Rather yes	Yes	Rather yes	Rather yes	Rather yes	Yes	Yes	Rather yes	Yes	Yes
How does the model fit into surrounded environment? It fits	Yes	Yes	Yes	Yes	Rather no	Yes	Rather yes	Rather yes	Yes	Rather yes	Yes
Rating of imagination of generative model?	75	75	100	75	75	100	100	100	100	100	90.00

Figure 5.12: The table showing the users' responses on the given questions

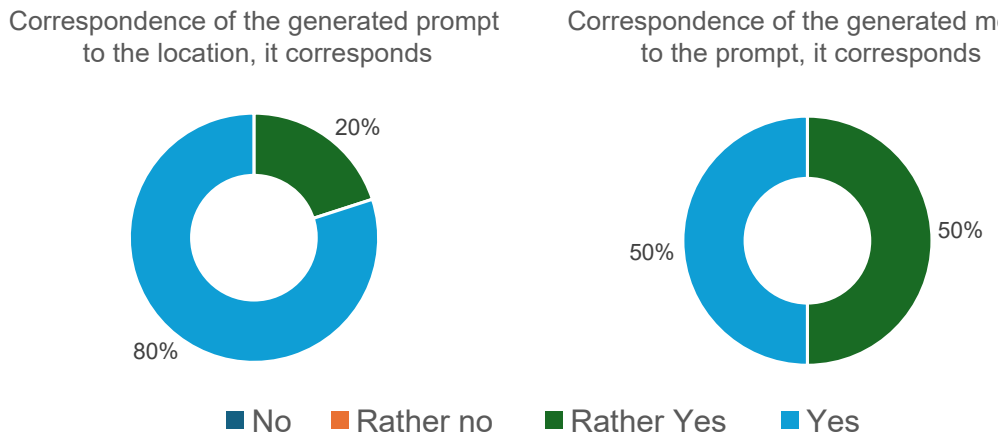

Figure 5.13: Graphs representing correspondence of the prompt to the location (on the left) and the generated model to the prompt (on the right)

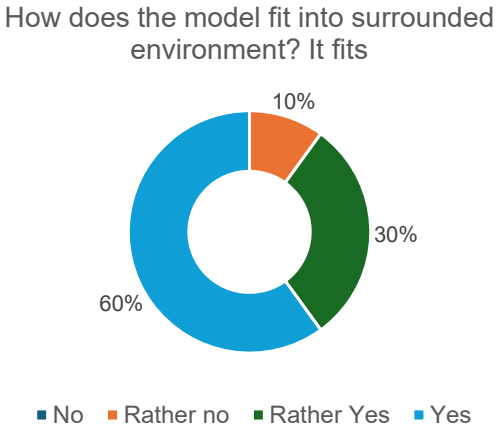

Figure 5.14: The graph representing the generated matching into surrounded environment

CHAPTER 6

Results

This chapter presents the results of the thesis. It primarily showcases the generated models placed in five distinct locations. Each location is illustrated on a map 6.3, with its own detailed map highlighting points of interest 6.4, 6.7, 6.9, 6.12, 6.14.

First, the interaction with the models and the level of detail are presented in the Figures 6.1, 6.2. A model can be selected by clicking on it and several icons show up. The user can see the like and dislike reaction counts along with the visitor count. Like and dislike icons appear, allowing users to rate the model. Each user can respond to a model once per day. The ”Model Info” panel, which shows the text prompts used in the generating process, can be opened by pressing the ”Info” icon.

If the model’s location is visited by many people, once the visitor count reach the number in which the model is replaced by a new model. The newly generated model can has the same prompts if the location has higher positive reaction than negative, otherwise the prompts are generated too.

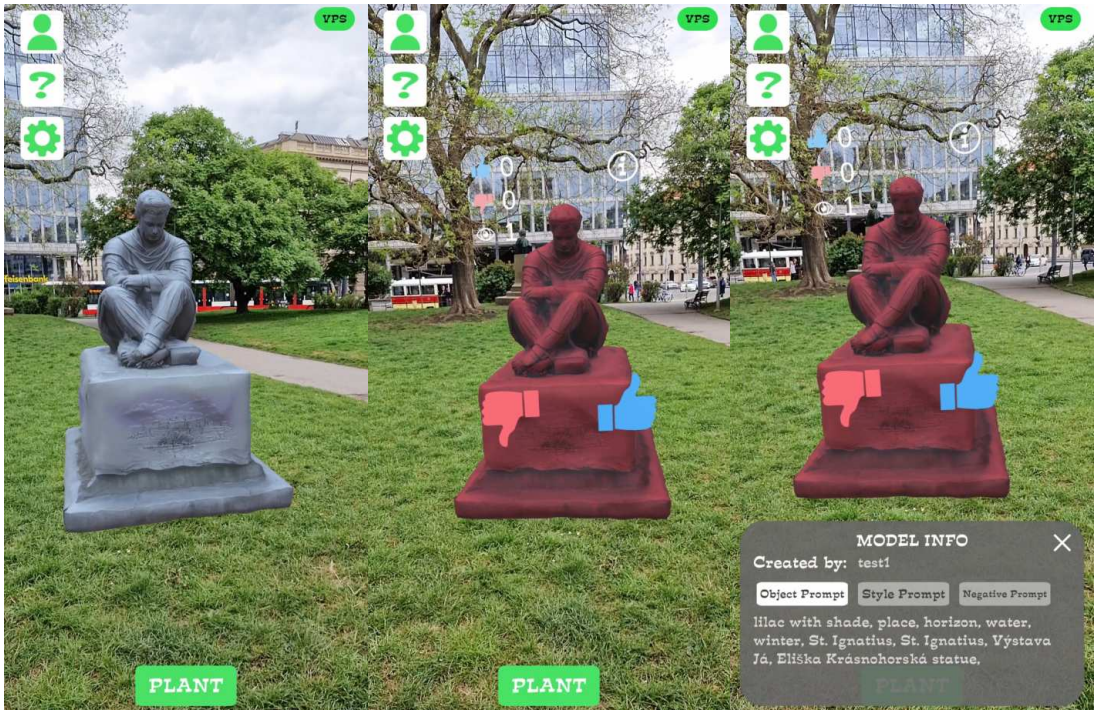

Figure 6.1: The images showing the interaction with a model. Upon selecting the model by touch, several icons appear. Pressing the ”Info” icon displays the ”Model Info” panel.

Figure 6.2: The images presenting the model's details. In the image on the right, the texture shows highly detailed towers.

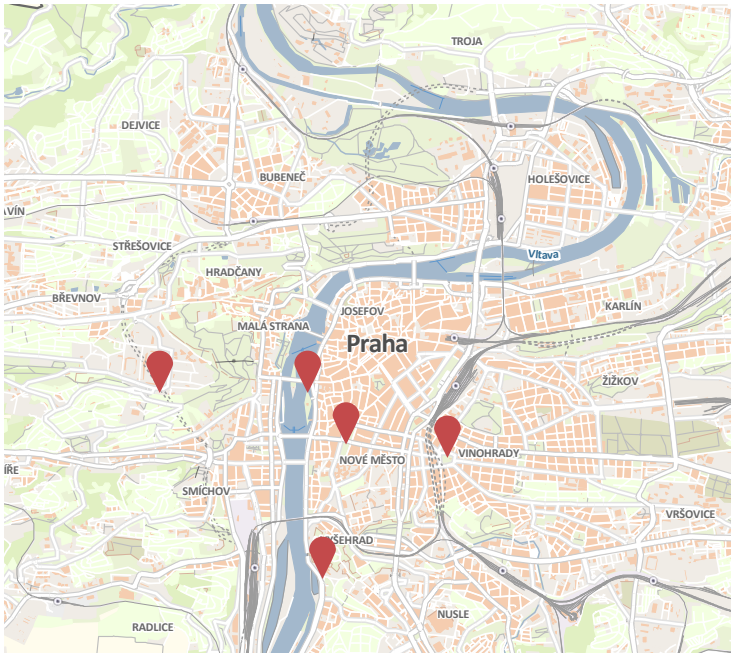

Figure 6.3: The map of all locations where models were generated

Figure 6.4: The map of Náměstí míru with the closest points of interest. The red arrow indicates the place where the models were generated.

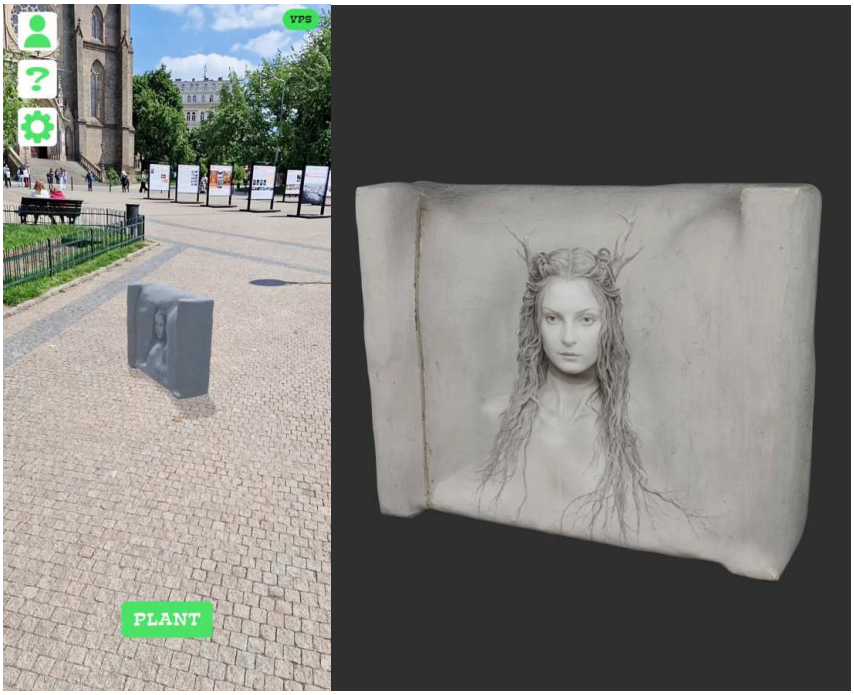

Figure 6.5: The generated model on Náměstí míru with following prompts: **Object prompt**: bush with appearance, sight, impression, trunk, root, system, fact, part, Namesti Miru, woman. **Style prompt**: gray, Performace Art, Da Vinci, metal. **Negative prompt**: low resolution, best quality, high quality, best quality

Figure 6.6: The generated model on Náměstí míru with following prompts: **Object prompt**: tree with shrine, letter, word, Century, book, World, map, animal, world, locations, monasteries, Order, abbeys, Bazilika sv. **Style prompt**: gray, Da Vinci, Symbolism, textile, wood, glass. **Negative prompt**: sharp, HD

Figure 6.7: The map of Karlovo náměstí with the closest points of interest. The red arrow indicates the place where the models were generated.

Figure 6.8: The generated model on Karlovo náměstí with following prompts: **Object prompt**: lilac with shade, place, horizon, water, winter, St. Ignatius, Vystava Ja, Eliska Krasnohorska statue. **Style prompt**: gray, black, silver, Graffiti Art, Performace Art, ceramic, brick

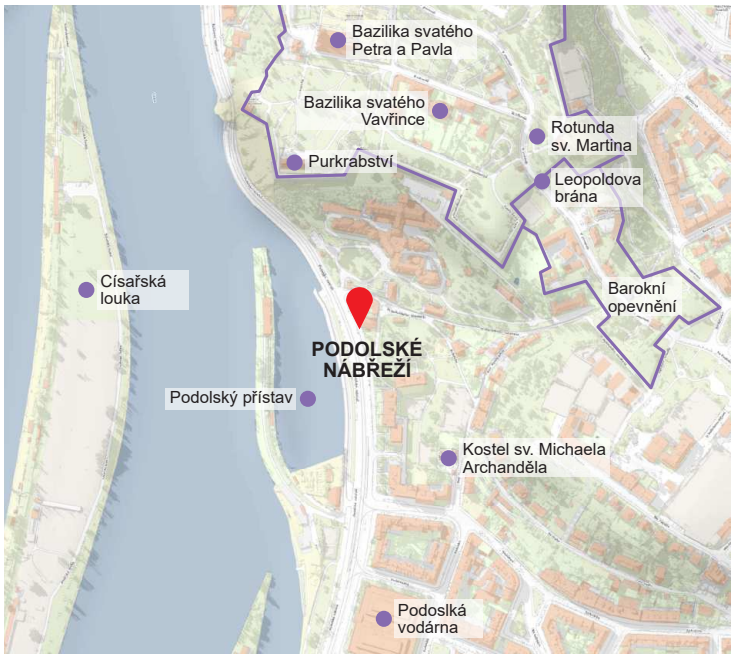

Figure 6.9: The map of Podolské nábřeží with the closest points of interest. The red arrow indicates the place where the models were generated.

Figure 6.10: The generated model on Podolské nábřeží with following prompts: **Object prompt**: vegetable with food, bread, sausage, plant, area, Libuse's Bath, Gothic Cellar - Permanent Exhibition. **Style prompt**: navy, silver, Installation Art, Classicism, Futurism, textile. **Negative prompt**: low poly, 4K, beautiful

Figure 6.11: The generated models on Podolské nábřeží with following prompts: **The model on the left. Object prompt**: statue, child, members, children, hands, prayers, tears, sorrow, minutes, witness, room, things, St. Michael Archangel. **Style prompt**: teal, Expressionism, steel, textile. **The model in the middle. Object prompt**: tree with root, example, type, times, fuel, furnace, buildings, tree, branches, branch, bark, half, Gothic Cellar - Permanent Exhibition, Paul Basilica, Park Na Topolce, traffic. **Style prompt**: gray, aqua, navy, Rococo, Precisionism, steel. **The model in the right. Object prompt**: lilac with flowers, website, plant, member, family, name, translation, word, flower, informations, species, Leaves, St. Michael Archangel, traffic, parking. **Style prompt**: silver, Symbolism, Expressionism, wood, stone, glass. **Negative prompt**: low resolution, HD, ugly

Figure 6.12: The map of Slovanský ostrov with the closest points of interest. The red arrow indicates the place where the models were generated.

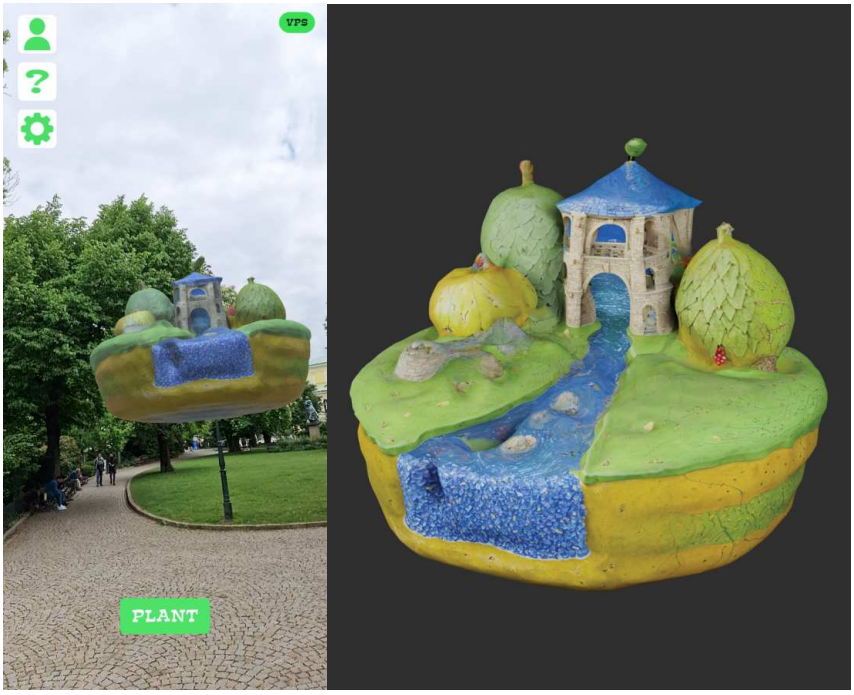

Figure 6.13: The generated model on Slovanský ostrov with following prompts: **Object prompt**: fruit with rocks, water-pit, river, Vytautovets, juice, berries, summer, vines, Legion Bridge, bench, tower. **Style prompt**: olive, Land Art, ceramic. **Negative prompt**: smooth, beautiful, highest quality, low quality

Figure 6.14: The map of Stadion Strahov with the closest points of interest. The red arrow indicates the place where the models were generated.

Figure 6.15: The generated model on Stadion Strahov with following prompts: **Object prompt**: fruit with legend, plant, thorn, ankle, girl, Ventilation tower Strahov tunnel, car. **Style prompt**: fuchsia, black, Land Art, Fauvism, Precisionism, plastic. **Negative prompt**: ugly, low resolution, HD

Figure 6.16: The generated models on Stadion Strahov with following prompts:. **The model on the left. Object prompt**: tree with branches, tree, metal, plate, swastika, numbers, legend, birthday, communist, revolution, site, casket, Ventilation tower Strahov tunnel, car, clock. **Style prompt**: maroon, silver, Dadaism, Expressionism, brick, ceramic, glass. **Negative prompt**: low quality. **The model on the right. Object prompt**: tree with name, Franz, Fels, Kraselkovska, Milan Rastislav Stefanik, Studanka Petrinka. **Style prompt**: gray, Expressionism, wood, brick. **Negative prompt**: HD, ugly, details, high poly

Conclusion

This thesis represents an explorative study into the integration of generative AI with Augmented Reality (AR) technologies, aimed at enhancing digital content creation. The central to this examination was the development of a system capable of generating 3D models tailored to specific environmental data. Throughout the research, I analyzed various aspects of AR application development and improvement, resulting to a robust design and implementation strategy for the system.

The system was designed as a conjunction of three components, each with its own role. The augmented reality application provides the environment for visualization and interaction with the generated 3D models and utilizes a localization service to provide precise location data for acquiring environmental information. The communication server facilitates access to a database storing 3D models and their essential data. The generative process runs on a separate computation server. This separation allows for easier updates of each component.

The finalized system stands as a testament to the potential of generative AI in AR settings, successfully demonstrating its capability to create digital content highly relevant to the contextual dynamics of the environment. Comprehensive testing and evaluation phases revealed that the system acquires information about the surrounding environment effectively, generates a text prompt which serves to create 3D models that integrate with their surrounding context. Based on the testing results, the text prompt creation could see some improvements to attain more precise captions, and the limitations of generative models were noted, such as the duration of generation or the correspondence of models to the text prompts. The current state of generative AI technologies is not suitable for applications requiring high precision and consistency yet. Despite these limitations, the system excels in scenarios where the primary objective is to generate and visualize results.

In summary, the outcomes of the thesis highlight the promising intersection of generative AI and AR technologies, showcasing significant advancements in digital content creation that are both innovative and applicable in real-world scenarios. The thesis paves the way for future studies for further refine these technologies, enhancing their reliability and precision for broader applications in various fields.

CHAPTER A

Application's Manual

This chapter provides a manual on how to control the application. The instructions will be primarily depicted in the following figures. The first Figure A.1 shows the initial screen displayed after the application is launched. By pressing the "Get Started" button, the game begins. If the user hasn't logged in already, he is invited to log in or register. After a successful authentication process, the user sees the main screen with a message. The message states that the application is waiting for the VPS service and the device's camera needs to be pointed at buildings around. Once the VPS is initialized, the user is allowed to seed a new plant.

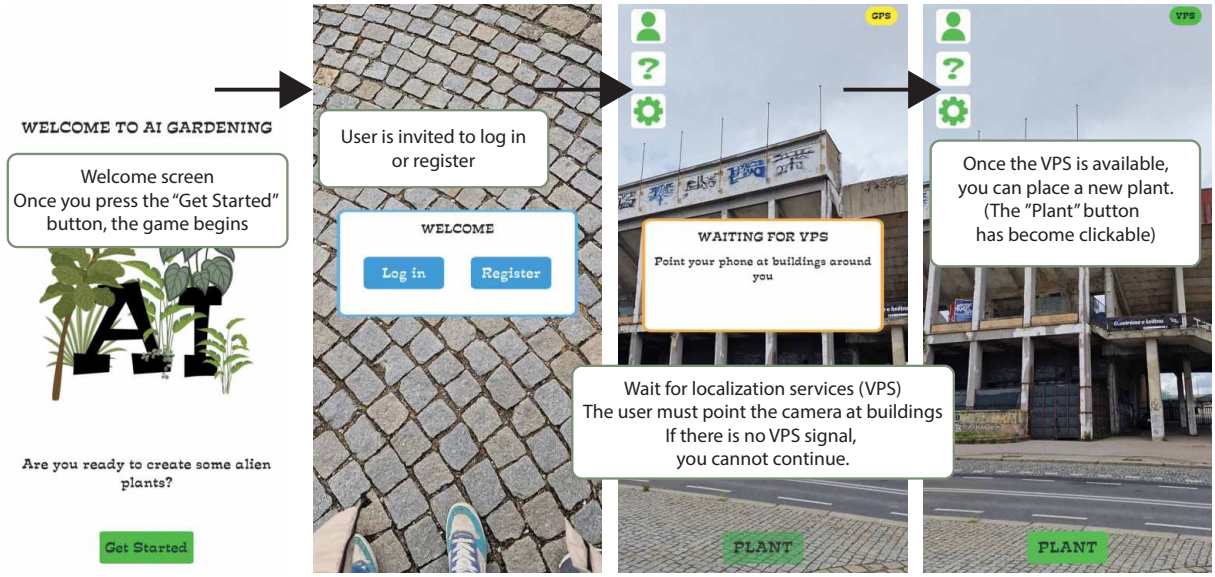

Figure A.1: The instructions to run the application

Placing the new object is illustrated in the Figure A.2. A green circle indicates where the new plant will be placed. The circle moves with the camera's centre. The plant is placed by pressing the "Plant" button. Immediately, the prefab plant model appears. The prefab model can be selected by clicking on it. Information about the generated model pops up, showing either the remaining time for generation or indicating that the model is loading into the scene. If the user wants to place another new object, it cannot be too close to other models. If it is too close, the green indicator becomes invisible, and the "Plant" button will be unclickable.

If the generated model is loaded into the scene, the user can select the model by clicking on it to view the model's information, as shown in the Figure A.3. Several pieces of information and icons are

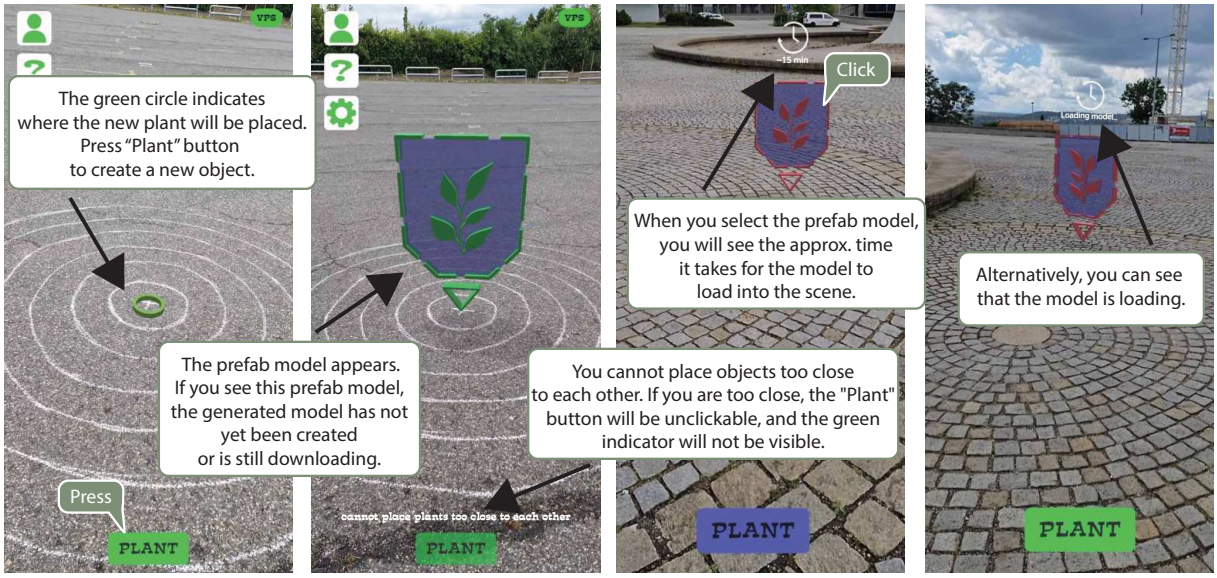

Figure A.2: The instructions to seed a new plant.

displayed. On the left side of the model, there is information about the number of visitors and like and dislike reactions. In the middle, in front of the model, there are two icons that can used to respond to the model by the user. The user can respond to each model once per day. On the right side of the model, there is an information icon that opens the model's information panel about its creation. The panel provides information about the creator and the prompts used in the generating process.

If the model is visited by many users, it will be renewed. If the number of like reactions is higher than the number of dislike reactions, the prompt is kept for the new model generation. Otherwise, the prompt and the model are generated again.

In the Figure A.4, there is an explanation of the icons on the main screen. These icons open the profile panel, help panel, and settings panel. In the profile panel, the user can see their name and can log out. The help panel provides information to assist with any problems that may arise. The settings panel allows the user to customize the application. It is possible to change the models' update interval, which sets how often the application updates plant models, model information, and requests new objects in the vicinity. The maximum distance range can also be changed, indicating the area within which models will be loaded into the scene. Occlusion culling is turned off by default but can be enabled. The user also has the option to change the user interface appearance by adjusting the theme colour and colour mode.

If you want to change the application settings such as the server's IP address or the minimal object distance, you need to open the Unity project, modify the Constants.cs file, change the keystore, and then build the application.

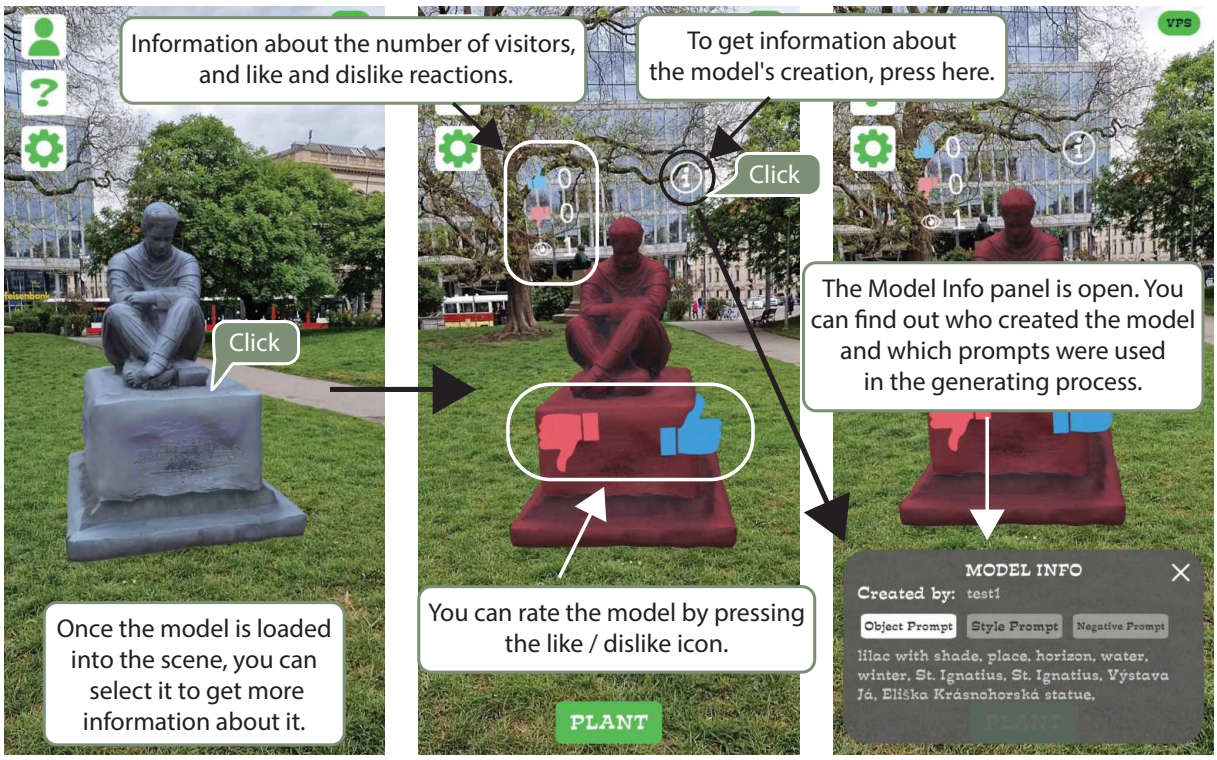

Figure A.3: The instructions to view details about the generated model.

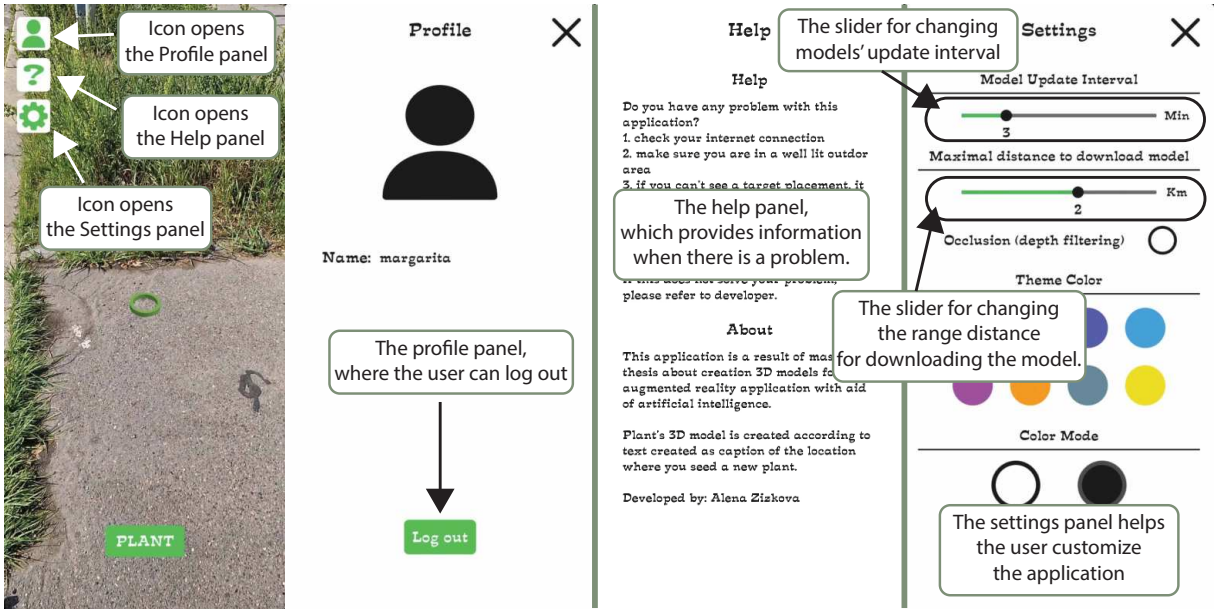

Figure A.4: The explanation of the icons on the main screen and the meaning of the showing panels.

CHAPTER B

Servers' Manual

This manual provides instructions on what needs to be done to run the implemented servers.

Downloading

First, download the project, and then download the pre-trained models (18 GB). These can be downloaded using the command line [93] or directly from the Hugging Face website:

- gpt2-xl [94]
- flan-t5-xl [95]
- blip-image-captioning-large [96]

Create the following folder structure within the folder server_computing\pretrained :

- gpt2-xl
  - config.json
  - generation_config.json
  - pytorch_model.bin
- blip-image-captioning-large
  - config.json
  - pytorch_model.bin
- flan-t5-xl
  - config.json
  - generation_config.json
  - pytorch_model-00001-of-00002.bin
  - pytorch_model-00002-of-00002.bin
- processors
  - blip-image-captioning-large
    - tokenizer.json
    - tokenizer_config.json
    - preprocessor_config.json
    - special_tokens_map.json
    - vocab.txt
- tokenizers
  - flan-t5-xl
    - tokenizer_config.json

    - special_tokens_map.json
    - spiece.model
  - gpt2-xl
    - tokenizer.json
    - tokenizer_config.json
    - special_tokens_map.json
    - vocab.json
    - merges.txt

Pretrained models can be replaced; in that case, it is necessary to modify the code in tasks.py, specifically the following functions:

- get_images_captions (image captioning)
- expand text (text generation)
- get_answers (generating answers for questions)

Installation

note: cmd = command line The size of installation packages is approximately 20 GB.

- enable windows features: Windows Subsystem for Linux, Virtual Machine Platform
- restart PC
- in BiOS enable Virtualization (Vtx)
- cmd as Administrator and run *wsl –install*
- open wsl and run *sudo apt-get install redis*
- restart PC
- install anaconda [97]
- open Powershell as Administrator
- run *Set-ExecutionPolicy -ExecutionPolicy RemoteSigned*
- run *conda init powershell*
- close powershell and open it again as Administrator
- run *conda config –set auto_activate_base false*
- run *conda create -n ai_kytky py anaconda pip* (or choose your virtual environment name, then edit powershell script servers.ps1)
- run *conda activate ai_kytky*
- run *conda install -n ai_kytky pytorch torchvision pytorch-cuda=12.1 -c pytorch -c nvidia* [98] (find version which fits to your cuda)
- run *conda install -n ai_kytky –file req_all.txt* (or *conda install –file requirements_conda.txt; pip install -r requirements_pip.txt*)
- run *python spacy -m download en-core-web-sm*

Configuration

Find the configuration file (config.json) and set the API keys exactly as needed. The servers' API key is generated automatically before starting.

- Google Cloud Api keys [99]
  - geocode_api_key - Geocoding API, Street View Static API
  - places_api_key - Places API
- Meshy Api key [100]

In the config file, set also the API addresses for the server host, including the ports. Then, open the command line and navigate to the server's folder where manage.py exists. Activate the previously created virtual environment, ai_kytky (activate ai_kytky) and execute the command *py manage.py migrate* in the servers' folders.

Running Servers

To run servers, start by opening PowerShell as an Administrator. Next, allow script execution by running the command *Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process*. Then, navigate to the folder where the servers are stored using the *cd* command followed by the path to your servers. Once in the correct directory, you can run the servers by executing the script *.\servers.ps1* with either the *pws* or *log* parameter. *pws* parameter directs the output to the PowerShell console, unlike the *log* parameter, which writes the output into a log file.

Bibliography

[1] I. Adamska, *What is mixed reality?*, NS FLOW, Jun. 2023. [Online]. Available: https://nsflow.com/blog/what-is-mixed-reality (visited on 04/10/2024).

[2] D. Pavlov, *Virtual and Augmented Reality in Hospitality Industry*, SmartTek Solutions, Sep. 2022. [Online]. Available: https://smarttek.solutions/blog/how-ar-vr-technologies-are-shaping-horeca-industry/ (visited on 04/16/2024).

[3] D. Pavlov, *Virtual Reality Training for Healthcare: The New Tool in Medical Education*, SmartTek Solutions, Jan. 2023. [Online]. Available: https://smarttek.solutions/blog/vr-training-for-healthcare-why-your-hospital-needs-it/ (visited on 04/17/2024).

[4] E. Balistrebi, *Extended reality is the future of education.* goHere, Apr. 2023. [Online]. Available: https://www.mixyourreality.com/insights/extended-reality-is-the-future-of-education (visited on 04/17/2024).

[5] J. Kessler, *What is augmented reality, and how does AR work?*, NS FLOW, Jun. 2023. [Online]. Available: https://nsflow.com/blog/what-is-augmented-reality-and-how-does-ar-work (visited on 04/12/2024).

[6] A. Gherghina, A. Olteanu, N. Tapus, *A marker-based augmented reality system for mobile devices*, Jan. 2013. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6511731/authors#authors (visited on 04/12/2024).

[7] A. Sheldon, T. Dobbs, A. Fabbri, N. Gardner, M. Hank Haeusler, C. Ramos, Y. Zavoleas, *PUTTING THE AR IN (AR)CHITECTURE: Integrating voice recognition and gesture control for Augmented Reality interaction to enhance design practice*, Distributed Systems Seminar FS2013, Apr. 2019. DOI: 10.52842/conf.caadria.2019.1.475. [Online]. Available: https://papers.cumincad.org/data/works/att/caadria2019_081.pdf (visited on 04/13/2024).

[8] A. Cebulla, *Projection-Based Augmented Reality*, Distributed Systems Seminar FS2013, Apr. 2013. [Online]. Available: https://www.academia.edu/38614182/Projection_Based_Augmented_Reality_Distributed_Systems_Seminar_FS2013 (visited on 04/13/2024).

[9] X. Wang, H. Yazdani Nezhad, *Augmented Reality (AR) Equipped Composites Repair - Postgraduate study*, Sep. 2019. [Online]. Available: https://www.researchgate.net/publication/335590182_Augmented_Reality_AR_Equipped_Composites_Repair_-_Postgraduate_study (visited on 04/14/2024).

[10] SAM, *Explore How Marker-Based AR Works for Your Business*, Zeal AR, Jun. 2022. [Online]. Available: https://zealar.com.au/explore-how-marker-based-ar-works-for-your-business/ (visited on 04/13/2024).

[11] *Location based AR - New concept of Augmented Reality*, JASH ENTERTAINMENT. [Online]. Available: https://jashentertainment.com/blog/Location-based-AR---New-concept-of-Augmented-Reality (visited on 04/13/2024).

[12] M. Akram, M. Sikander HayatKhiayl, M. Ahmad, S. Aqeel Abbas, *Decision Tree for Selection Appropriate Location Estimation Technique of GSM Cellular Network*, International Conference on Engineering & Emerging Technology, Pakistan, Mar. 2014. [Online]. Available: https://www.researchgate.net/publication/264128575_Decision_Tree_for_Selection_Appropriate_Location_Estimation_Technique_of_GSM_Cellular_Network (visited on 04/13/2024).

[13] *Visualization of the Constructed Map and Trajectory of the Robot*, MathWorks. [Online]. Available: https://it.mathworks.com/help/nav/ug/implement-simultaneous-localization-and-mapping-with-lidar-scans.html (visited on 04/13/2024).

[14] GISGeography, *How GPS Receivers Work – Trilateration vs Triangulation*, Mar. 2024. [Online]. Available: https://gisgeography.com/trilateration-triangulation-gps/ (visited on 04/13/2024).

[15] S. Anup, A. Goel, S. Padmanabhan, *Visual positioning system for automated indoor/outdoor navigation*, TENCON 2017 - 2017 IEEE Region 10 Conference, Mar. 2017. DOI: 10.1109/TENCON.2017.8228008. [Online]. Available: https://ieeexplore.ieee.org/document/8228008 (visited on 04/13/2024).

[16] M. Dhande, *Getting lost with GPS? Don't worry, Google Maps VPS is here*, Geospatial World, Jan. 2023. [Online]. Available: https://www.geospatialworld.net/prime/business-and-industry-trends/what-is-visual-positioning-system-vps/ (visited on 04/13/2024).

[17] *Build global-scale, immersive, location-based AR experiences with the ARCore Geospatial API*, Google for Developers, Mar. 2024. [Online]. Available: https://developers.google.com/ar/develop/geospatial (visited on 04/13/2024).

[18] N. Régo, *Scape Technologies Launches Visual Positioning Service That Gives Location Accuracy Far Beyond GPS*, Cool Blind Tech, Jan. 2019. [Online]. Available: https://coolblindtech.com/scape-technologies-launches-visual-positioning-service-that-gives-location-accuracy-far-beyond-gps/ (visited on 04/13/2024).

[19] E. Miller, *Vision for the Future*, Medium, Jan. 2019. [Online]. Available: https://medium.com/scape-technologies/vision-for-the-future-dcb7d058b0d3 (visited on 04/13/2024).

[20] B. Lang, *Scape Raises 8M for Visual Positioning System Which Could be a Key Component of the AR Metaverse*, Road Tovr, Jan. 2019. [Online]. Available: https://www.roadtovr.com/scape-raises-8m-for-visual-positioning-system-which-could-be-a-key-component-of-the-ar-metaverse/ (visited on 04/13/2024).

[21] B. Sidhu, *Make the world your canvas with the ARCore Geospatial API*, Google Blog, May 2022. [Online]. Available: https://developers.googleblog.com/2022/05/Make-the-world-your-canvas-ARCore-Geospatial-API.html (visited on 04/13/2024).

[22] J. Fisher, D. Bartz, W. Strasser, *Enhanced visual realism by incorporating camera image effects*, 2006 IEEE/ACM International Symposium on Mixed and Augmented Reality, Oct. 2006. DOI: 10.1109/ISMAR.2006.297815. [Online]. Available: https://ieeexplore.ieee.org/document/4079277 (visited on 04/14/2024).

[23] *Get the lighting right*, Google for Developers, Mar. 2024. [Online]. Available: https://developers.google.com/ar/develop/lighting-estimation (visited on 04/14/2024).

[24] F. Ghayour, D. Cantor, *Real-Time 3D Graphics with WebGL 2 - Second Edition*, Oct. 2018. [Online]. Available: https://www.oreilly.com/library/view/real-time-3d-graphics/9781788629690/e08d3981-4690-45d5-b2c0-8cd6158b11d7.xhtml (visited on 04/14/2024).

[25] A. Gosalia, *Updates to ARCore Help You Build More Interactive & Realistic AR Experiences*, Google Blog, May 2019. [Online]. Available: https://developers.googleblog.com/2019/05/ARCore-IO19.html (visited on 04/15/2024).

[26] *Environment Probes*, Unity Technologies, Oct. 2023. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual/features/environment-probes.html (visited on 04/15/2024).

[27] *Environment Probes*, Unigine, Apr. 2024. [Online]. Available: https://developer.unigine.com/en/docs/latest/editor2/lighting/gi/env_probes?rlang=cpp (visited on 04/15/2024).

[28] M. Alfakhori, J. Aebastián Barzallo, V. Coors, *Occlusion Handling for Mobile AR Applications in Indoor and Outdoor Scenarios*, Centre for Geodesy and Geoinformatics, Stuttgart University of Applied Sciences, Apr. 2023. DOI: 10.3390/s23094245. [Online]. Available: https://doi.org/10.3390/s23094245 (visited on 04/15/2024).

[29] Z. Zhao, J. Zhang, S. Xu, Z. Lin, H. Pfister, *Discrete Cosine Transform Network for Guided Depth Map Super-Resolution*, Apr. 2021. [Online]. Available: https://arxiv.org/abs/2104.06977 (visited on 04/15/2024).

[30] *Understand the user's environment with the Scene Semantics API*, Google for Developers, Mar. 2024. [Online]. Available: https://developers.google.com/ar/develop/scene-semantics (visited on 04/16/2024).

[31] *ARCore*, Google for Developers. [Online]. Available: https://developers.google.com/ar (visited on 04/16/2024).

[32] *Augmented Reality*, Apple Developer. [Online]. Available: https://developer.apple.com/augmented-reality/ (visited on 04/16/2024).

[33] A. Makarov, *12 Augmented Reality Trends of 2024: New Milestones in Immersive Technology*, Mobidev, Apr. 2024. [Online]. Available: https://mobidev.biz/blog/augmented-reality-trends-future-ar-technologies (visited on 04/25/2024).

[34] *Vuforia: Market-Leading Enterprise AR*, PTC. [Online]. Available: https://www.ptc.com/en/products/vuforia (visited on 04/16/2024).

[35] *Augmented Reality / Virtual Reality*, Kudan. [Online]. Available: https://www.kudan.io/blog/solution/ar-vr/ (visited on 04/16/2024).

[36] *AR Foundation*, Unity. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.0/manual/index.html (visited on 04/16/2024).

[37] *Unity*, Unity Technologies. [Online]. Available: https://unity.com/ (visited on 04/25/2024).

[38] *8th Wall*, Niantic. [Online]. Available: https://www.8thwall.com/ (visited on 04/19/2024).

[39] *What is artificial intelligence (AI)?*, IBM. [Online]. Available: https://www.ibm.com/topics/artificial-intelligence (visited on 04/08/2024).

[40] *Artificial Intelligence - What it is and why it matters*, sas. [Online]. Available: https://www.sas.com/en_us/insights/analytics/what-is-artificial-intelligence.html#:~:text=Artificial%20intelligence%20(AI)%20makes%20it,learning%20and%20natural%20language%20processing.html (visited on 04/08/2024).

[41] R. Gupta, *Research Paper on Artificial Intelligence*, International Journal Of Engineering And Computer Science, Feb. 2023. [Online]. Available: https://www.researchgate.net/publication/371426909_Research_Paper_on_Artificial_Intelligence (visited on 04/08/2024).

[42] I. Data and A. Team, *Understanding the different types of artificial intelligence*, IBM, Oct. 2023. [Online]. Available: https://www.ibm.com/blog/understanding-the-different-types-of-artificial-intelligence/ (visited on 04/08/2024).

[43] *What is a Neural Network?*, Amazon Web Service, Inc. [Online]. Available: https://aws.amazon.com/what-is/neural-network/#:~:text=A%20neural%20network%20is%20a,that%20resembles%20the%20human%20brain.html (visited on 04/08/2024).

[44] N. Buhl, *Training vs. Fine-tuning: What is the Difference?*, Encord, Nov. 2023. [Online]. Available: https://encord.com/blog/training-vs-fine-tuning/ (visited on 04/09/2024).

[45] J. Paul Mueller, L. Massaron, *Machine Learning For Dummies, IBM Limited Edition*, John Wiles & Sons.Inc, 2018. [Online]. Available: https://www.ibm.com/downloads/cas/GB8ZMQZ3 (visited on 04/09/2024).

[46] I. H. Sarker, *Machine Learning: Algorithms, Real-World Applications and Research Directions*, SN COMPUT. SCI. 2, Mar. 2021. [Online]. Available: https://doi.org/10.1007/s42979-021-00592-x (visited on 04/09/2024).

[47] *What is machine learning (ML)?*, IBM. [Online]. Available: https://www.ibm.com/topics/machine-learning (visited on 04/09/2024).

[48] baeldung, *Multi-Layer Perceptron vs. Deep Neural Network*, Bealdung, Jun. 2023. [Online]. Available: https://www.baeldung.com/cs/mlp-vs-dnn (visited on 04/09/2024).

[49] D. Bergmann, *What is zero-shot learning?*, IBM, Jan. 2024. [Online]. Available: https://www.ibm.com/topics/zero-shot-learning (visited on 04/09/2024).

[50] *What is Generative AI?*, NVIDIA. [Online]. Available: https://www.nvidia.com/en-us/glossary/generative-ai/ (visited on 04/09/2024).

[51] J. Brownlee, *A Gentle Introduction to Generative Adversarial Networks (GANs)*, Generative Adversarial Networks, Jul. 2019. [Online]. Available: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/ (visited on 04/09/2024).

[52] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, *Generative Adversarial Networks*, Generative Adversarial Networks, Jun. 2014. [Online]. Available: https://arxiv.org/abs/1406.2661 (visited on 04/09/2024).

[53] I. J. Goodfellow, A. Courville, Y. Bengio, *Deep Learning*, MIT Press, Nov. 2016.

[54] M. Müller, *Camera Re-Localization with Data Augmentation by Image Rendering and Image-to-Image Translation*, PhD Thesis, Apr. 2020. [Online]. Available: https://www.researchgate.net/publication/342379262_Camera_Re-Localization_with_Data_Augmentation_by_Image_Rendering_and_Image-to-Image_Translation (visited on 04/09/2024).

[55] T. Karras, S. Laine, T. Aila, *A Style-Based Generator Architecture for Generative Adversarial Networks*, CVPR 2019, Dec. 2018. [Online]. Available: https://arxiv.org/abs/1812.04948 (visited on 03/28/2024).

[56] R. Or-El, X. Luo, M. Shan, E. Shechtman, J. J. Park, I. Kemelmacher-Shlizerman, *StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation*, CVPR 2022, Dec. 2021. [Online]. Available: https://arxiv.org/abs/2112.11427 (visited on 03/28/2024).

[57] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng, *Representing Scenes as Neural Radiance Fields for View Synthesis*, ECCV 2020 Oral - Best Paper Honorable Mention, Mar. 2020. [Online]. Available: https://www.matthewtancik.com/nerf (visited on 03/25/2024).

[58] S. Blank, R. Pesch, *End-to-End Image Captioning*, Inovex, Apr. 2020. [Online]. Available: https://www.inovex.de/de/blog/end-to-end-image-captioning/ (visited on 04/11/2024).

[59] A. Radford, J. W. Kim, Ch. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever, *Learning Transferable Visual Models From Natural Language Supervision*, Feb. 2021. [Online]. Available: https://arxiv.org/abs/2103.00020 (visited on 04/11/2024).

[60] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, H. Lee, *Generative Adversarial Text to Image Synthesis*, May 2016. [Online]. Available: https://arxiv.org/abs/1605.05396 (visited on 04/11/2024).

[61] Ch. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. Kamyar Seyed Ghasemipour, B. Karagol Ayan, S. Sara Mahdavi, R. Gontijo Lopes, T. Salimans, J. Ho, D. J Fleet, M. Norouzi, *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding*, May 2020. [Online]. Available: https://arxiv.org/abs/2205.11487 (visited on 04/12/2024).

[62] D. Agnihotri, *End-to-End Image Captioning*, Medium, Nov. 2023. [Online]. Available: https://medium.com/@agnidhananjay/hierarchical-text-conditional-image-generation-with-clip-latents-dall-e2-ce6d5a5ce799 (visited on 04/12/2024).

[63] Ch. Zhang, Ch. Zhang, M. Zhang, I. So Kweon, *Text-to-image Diffusion Models in Generative AI: A Survey*, Mar. 2023. [Online]. Available: https://arxiv.org/abs/2303.07909 (visited on 04/12/2024).

[64] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, B. Poole, *Zero-Shot Text-Guided Object Generation with Dream Fields*, CVPR 2022, Dec. 2021. [Online]. Available: https://arxiv.org/abs/2112.01455 (visited on 04/12/2024).

[65] B. Poole, A. Jain, J. T. Barron, B. Mildenhall, *DreamFusion: Text-to-3D using 2D Diffusion*, Dec. 2022. [Online]. Available: https://arxiv.org/abs/2209.14988 (visited on 12/05/2023).

[66] *Score Distillation Sampling (SDS)*, Mindverse, Jan. 2024. [Online]. Available: https://www.mind-verse.de/en/news/score-distillation-sampling-sds (visited on 04/13/2024).

[67] P. Wang, Z. Fan, D. Xu, D. Wang, S. Mohan, F. Iandola, R. Ranjan, Y. Li, Q. Liu, Z. Wang, V. Chandra, *SteinDreamer: Variance Reduction for Text-to-3D Score Distillation via Stein Identity*, Dec. 2023. [Online]. Available: https://arxiv.org/abs/2401.00604 (visited on 04/13/2024).

[68] *Google Cloud - API Library*, Google. [Online]. Available: https://console.cloud.google.com/apis/library (visited on 05/09/2024).

[69] *PokémonGo*, The Pokémon Company, Niantic. [Online]. Available: https://pokemongolive.com/?hl=en (visited on 04/25/2024).

[70] W. Shanklin, *'Pokémon Go' developer Niantic is laying off 230 employees*, Engadget, Jun. 2023. [Online]. Available: https://www.engadget.com/pokemon-go-developer-niantic-is-laying-off-230-employees-180438129.html? (visited on 04/25/2024).

[71] *IKEA Place app launched to help people virtually place furniture at home*, Inter IKEA newsroom, Sep. 2017. [Online]. Available: https://www.ikea.com/global/en/newsroom/innovation/ikea-launches-ikea-place-a-new-app-that-allows-people-to-virtually-place-furniture-in-their-home-170912/ (visited on 04/25/2024).

[72] A. R. Thimmapurmath, BG. Shashank, S. Harshith, S. Chidaravalli, *Virtual Veda – Visualize Plants through Augmented Reality*, International Journal of Advanced Research in Science, Communication and Technology, Feb. 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267611008 (visited on 04/19/2024).

[73] *Quiver*, Quiver Vision. [Online]. Available: https://quivervision.com (visited on 04/25/2024).

[74] *Google Maps Help - Use Live View on Google Maps*, Google, 2024. [Online]. Available: https://support.google.com/maps/answer/9332056?hl&co=GENIE.Platform%3DAndroid#zippy=%2Cnavigace-s-funkc%C3%AD-live-view (visited on 04/19/2024).

[75] A. Chaturvedi, *Google Maps rolls out AR navigation for iOS and Android smartphones*, Geospatial World, Sep. 2019. [Online]. Available: https://www.geospatialworld.net/blogs/google-maps-ar-navigation-iphones-android/ (visited on 04/19/2024).

[76] *Google ARCore Pocket Garden*, BUCK, Google, 2022. [Online]. Available: https://buck.co/work/google-gracie-pocket-garden (visited on 04/19/2024).

[77] *Meshy*. [Online]. Available: https://www.meshy.ai/ (visited on 04/19/2024).

[78] *Text-to-3D API*, Stable Diffusion. [Online]. Available: https://stablediffusionapi.com/text-to-3d (visited on 04/19/2024).

[79] J. Tang, *Stable-dreamfusion: Text-to-3D with Stable-diffusion*, 2022. [Online]. Available: https://github.com/ashawkey/stable-dreamfusion (visited on 03/20/2024).

[80] *Geocoding API*, Google. [Online]. Available: https://developers.google.com/maps/documentation/geocoding (visited on 05/15/2024).

[81] *Places API*, Google. [Online]. Available: https://developers.google.com/maps/documentation/places/web-service (visited on 05/15/2024).

[82] *Steet View Static API*, Google. [Online]. Available: https://developers.google.com/maps/documentation/streetview (visited on 05/15/2024).

[83] J. Bennett, T. Komiya, A. Johnson, J. Ardell, *Webcolors*. [Online]. Available: https://webcolors.readthedocs.io/en/latest/index.html (visited on 05/15/2024).

[84] *Hugging Face*, Hugging Face community. [Online]. Available: https://huggingface.co/ (visited on 05/15/2024).

[85] J. Li, D. Li, C. Xiong, S. Hoi, *BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation*, Jan. 2022. [Online]. Available: https://doi.org/10.48550/arxiv.2201.12086 (visited on 05/15/2024).

[86] *Meshy API documentation for Text-to-3D generation*, Meshy. [Online]. Available: https://docs.meshy.ai/api-text-to-3d-beta (visited on 05/17/2024).

[87] *Google Cloud Calculator*, Google. [Online]. Available: https://cloud.google.com/products/calculator?hl=cs (visited on 05/16/2024).

[88] *IBM Cloud Pricing*, IBM. [Online]. Available: https://www.ibm.com/cloud/pricing (visited on 05/16/2024).

[89] *Amazon Web Service Cloud Pricing*, Amazon Web Service. [Online]. Available: https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all (visited on 05/16/2024).

[90] *Microsoft Azure Cloud Pricing*, Microsoft Azure. [Online]. Available: https://azure.microsoft.com/en-us/free/ (visited on 05/16/2024).

[91] *ARCore Unity Extensions package's repository*, Google - ARCore. [Online]. Available: https://github.com/google-ar/arcore-unity-extensions.git (visited on 05/17/2024).

[92] nyu, *How to Create Shadows for AR Objects*, STYLY, Inc., Feb. 2023. [Online]. Available: https://styly.cc/tips/shadowcasting-objects-ground-arscene/ (visited on 05/17/2024).

[93] *Downloading model using command line*, Huggingface. [Online]. Available: https://huggingface.co/docs/hub/en/models-downloading (visited on 05/17/2024).

[94] *Downloading GPT2 model*, Huggingface. [Online]. Available: https://huggingface.co/openai-community/gpt2-xl/tree/main (visited on 05/17/2024).

[95] *Downloading Flan-T5 model*, Huggingface. [Online]. Available: https://huggingface.co/google/flan-t5-xl/tree/main (visited on 05/17/2024).

[96] *Downloading BLIP model*, Huggingface. [Online]. Available: https://huggingface.co/Salesforce/blip-image-captioning-large/tree/main (visited on 05/17/2024).

[97] *Anaconda documentation*, Anaconda, Inc. [Online]. Available: https://docs.anaconda.com/free/anaconda/install/index.html (visited on 05/17/2024).

[98] *PyTorch Documentation*, PyTorch. [Online]. Available: https://pytorch.org/get-started/locally/ (visited on 05/17/2024).

[99] *Google Cloud - Creating credentials*, Google. [Online]. Available: https://console.cloud.google.com/apis/credentials (visited on 05/17/2024).

[100] *Meshy API key*, Meshy. [Online]. Available: https://app.meshy.ai/api (visited on 05/17/2024).