Dynamic Integration of 3D Augmented Reality Features with AI-Based Contextual and Personalized Overlays in Asset Management

Kessel Okinga Koumou[a] and Omowunmi E. Isafiade[b]
Department of Computer Science, Faculty of Natural Sciences, University of the Western Cape, Bellville, Cape Town 7535, South Africa
fi

Keywords: Immersive Technology, Augmented Reality (AR), Artificial Intelligence (AI), DeepLink, Long Short Term Memory (LSTM), Three Dimensional (3D), Asset Management (AM).

Abstract: This study addresses the challenges of manual implementation of 3D models in AR and the scalability limitations of AR applications in asset management. It proposes a framework for the dynamic integration of 3D models into the AR environment, incorporating AI to enhance textual content and personalized user engagement. The study presents a system architecture comprising three layers: (i) The bottom layer, which handles the interactive capabilities of 3D models, including collision detection, mesh manipulation, dataset preparation, and model training; (ii) The middle layer, which facilitates communication between the web asset management platform and mobile application developed; and (iii) The topmost layer, which focuses on user interaction with the 3D models via the web platform. To evaluate the framework, two 3D models (microscope and centrifuge) were used as case studies for dynamic integration. The AI component was trained using a dataset based on the microscope information obtained with web scrapping. The model was trained using both Standard LSTM and BiLSTM architectures, with the dataset split into 60% for training, 20% for testing, and 20% for validation, over 50 epochs with a batch size of 64. The BiLSTM outperformed the Standard LSTM, achieving a test accuracy of 94.35% and a test loss of 0.51. This research is significant in revolutionizing asset management and promoting personalized content for quality education through technological innovation.

1 INTRODUCTION

Over the past decades, immersive technology and artificial intelligence (AI) have tremendously been integrated into many domains such as education, construction, retail, tourism, and military (Limna et al., 2022; Koumou and Isafiade, 2024). These two revolutionary technologies have transformed the way we interact with information and utilize systems. Immersive technology enhances the visualization of information through interactive and engaging virtual or three-dimensional (3D) elements or objects, and AI has revolutionized data analysis, enabling accurate predictions and personalized user experiences (Rampini et al., 2022; Datta et al., 2024).

Immersive technology is a technology that blurs the boundary between physical and virtual worlds by providing high quality or quantity of sensory information, enabling users to experience a profound sense of immersion (Suh and Prophet, 2018). There are three types of immersive technology, which are: (i) augmented reality (AR), which superimposes 3D elements or objects onto the user's physical world (Cipresso et al., 2018); (ii) virtual reality (VR), which fully immerses users in a virtual environment (Koumou and Isafiade, 2024); and (iii) mixed reality (MR), which provides users with a blended environment (Karaaslan et al., 2019). These technologies improve user satisfaction (Koumou et al., 2023), enrich learning experiences (Sepasgozar, 2020), and provide safe training environments (Braun et al., 2022).

AI is the ability of a computer program to learn and think like a human being (Zhang and Lu, 2021; Limna et al., 2022). AI development often uses pre-trained models, which are trained on large datasets for specific problems and can be fine-tuned for various applications (Han et al., 2021). Research by Abdulhamied et al., (Abdulhamied et al., 2023) proposed a system that can recognize and interpret American Sign Language (ASL) using long short term memory (LSTM) networks and hand detection techniques.

[a] https://orcid.org/0009-0003-4776-2360
[b] https://orcid.org/0000-0002-3028-6180

The authors used the MediaPipe tool to identify hand movements and the LSTM model to predict which signs and movements are being presented. However, AR was not used in the paper; instead, the authors suggested that superimposing features on the skeleton's bones using AR mechanisms would be ideal.

Moreover, the development of this immersive technology such as AR can be expensive. For example, retailers such as IKEA's e-commerce platform introduced an AR feature to its e-commerce platform to provide shoppers with a more flawless presentation of what a piece of furniture might look like once placed in its intended spot. The company integrated over two thousand (2000) 3D rendering models into the application, which was developed using ARKit (Alves and Luís Reis, 2020; Ozturkcan, 2021). The innovative solution demonstrated substantial investments in time and financial resources to enhance customer shopping experience. Similarly, in a Southern African institution, a web-based application was developed to manage assets. This application incorporated AR features to create virtual representations of assets, allowing users to preview and understand their functionality before making a request. This feature aimed to reduce the mishandling of expensive assets, particularly among inexperienced users. Yet, a notable drawback of the system is the inability of end-users to upload custom 3D models when adding new assets, as it relies on pre-developed 3D-based AR, which is meant to be automatically attached to newly added assets based on corresponding matches. It is worth highlighting that different owners of assets may possess various types of assets, each potentially requiring different 3D models (Koumou et al., 2023). Research by Garzon (Garzón, 2021) highlights the scalability limitations of AR and the challenges of manually implementing 3D models across platforms. The author also notes that barriers still prevent users with special needs from fully benefiting from AR, emphasizing that special needs require tailored solutions for a range of diverse needs.

In line with the aforementioned research, this work developed a framework to facilitate the dynamic integration of 3D models into an AR environment. Furthermore, it integrates AI through an application programming interface (API) to augment content visualization based on gesture recognition of the 3D model's nodes. This work used an existing web-based platform, SciAssetHub, which is based on asset management with a limited 3D model embedded that was developed manually. The development of the framework involved three main steps: First, developing a mobile AR platform that allows users to view an uploaded 3D model in both 3D and AR modes, along with training the AI model using LSTM architecture and integrating the trained model into the 3D models. Second, communication between the web and mobile applications is enabled through pattern recognition. Third, enabling the upload of 3D models from the existing web application, SciAssetHub. Furthermore, this work aims to incorporate and address future research suggested by Dyulicheva and Glazieva (Dyulicheva and Glazieva, 2021), which emphasized the need to incorporate AI with immersive technology. Hence, this research aims to address this by recommending best practices or guidelines for integrating immersive technology with other technologies such as AI.

The remainder of the paper is organized as follows: Section II presents the literature review, the methodology is documented in Section III, Section IV presents the results and discussions, and Section V concludes the research and provides future recommendations.

2 LITERATURE REVIEW

This section provides a general overview of AR and AI, followed by case studies and proposed solutions to demonstrate how these technologies have been integrated and applied in various settings.

2.1 AR Implementation in Different Settings

The ability to superimpose a virtual element onto the physical world, and accurately interact with that virtual element makes AR a powerful tool for enhancing user experiences. The key feature of AR rests on the idea of spatial registration, where the digital object has a physical location in the real world, considering physical objects and the end-user's point of view as if they were in the physical world (Wang et al., 2013). AR technology has been integrated into various fields to overcome complex challenges, for example in railway asset management. Due to the dispersion of assets along extended rail networks, AR was introduced to enable the faster transfer of asset information directly to track workers, regardless of their location. This allows data to be presented in a digital format, overlaid onto the real-world objects the workers are working with. By providing real-time inspection and condition monitoring data, AR supports workers by displaying this information on detailed 3D models of railway assets (Garramone et al., 2022). Furthermore, in another scenario, AR has been integrated into daily operations to enhance inventory management and improve customer engagement due to the lack of interaction with physical assets in physical stores. Research by Asta et al., (Asta et al., 2024) demonstrated that AR significantly improves data visualization, daily sales, and inventory management efficiency in the retail sector through the developed AR application that allows retail managers to view data in a more interactive format, facilitating faster and more accurate decision-making. Furthermore, the authors pointed out that the study participants reported high levels of satisfaction with using these applications, indicating that AR can overcome the limitations of traditional data visualization methods and enhance user satisfaction.

2.2 Artificial Intelligence and Text Generation

Recurrent neural network (RNN) is a type of neural network architecture within the field of AI. RNN consists of layers of interconnected nodes with looped connections, allowing them to use memory to process sequences and generate text by predicting the next word based on previous inputs (Hussein and Savaş, 2024). Types of RNN, such as LSTM and gated recurrent units (GRU) have been widely used for text generation. Various researchers have considered these types of artificial neural network (ANN) architectures to address the limitations of traditional RNN, which struggle with long-term dependencies due to vanishing gradients (Lipton, 2015). Research by Abujar et al., (Abujar et al., 2019) proposed a model that can generate text based on the Bangla language using bi-directional LSTM (BiLSTM). The authors used a sequence-to-sequence technique to predict the next word in a sentence based on the previous words. The model was trained over 100 epochs, and two activation functions such as rectified linear unit (ReLu) and Softmax. Similarly, Ibne Akhtar et al., (Ibne Akhtar et al., 2021) developed a solution for generating text in the Bangla language using a BiLSTM, optimized GRU network. the model was trained with 75% of the dataset and 25% for testing, the batch size was 256, and the number of epochs 150. The model achieved an accuracy of 97%. Additionally, research by Li and Zhang (Li and Zhang, 2021) compared the quality of generated text produced by LSTM (with peephole), GRU, and standard (without peephole) LSTM, using bidirectional encoder representations from transformer (BERT) and bilingual evaluation understudy with representations from transformer (BLEURT) evaluation metrics. The authors concluded that LSTM performed better.

2.3 AI-Integrated User Interaction with Immersive Technology

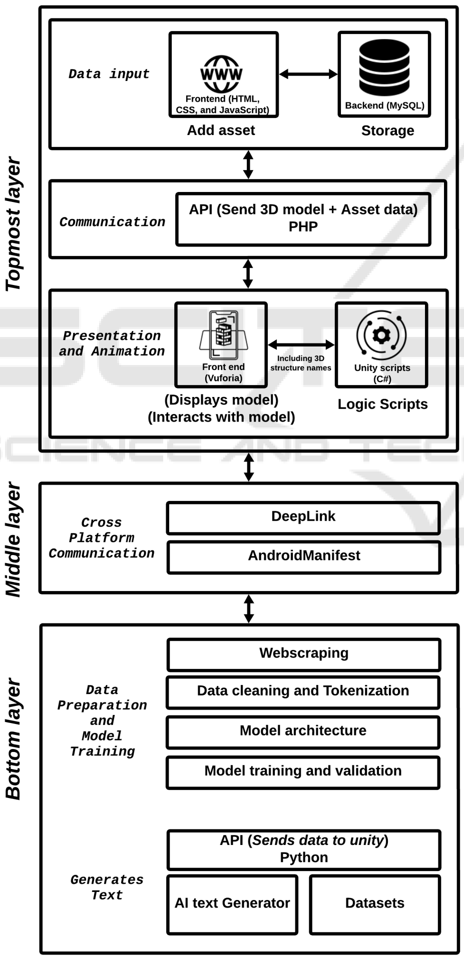
Delving into the existing literature on user interaction within the context of AI and immersive technology is crucial for understanding the key aspects linking these two technologies. Bassyouni and Elhajj (Bassyouni and Elhajj, 2021) highlight that AR can serve as an interface for visualizing AI algorithm outputs in real-time. They also note that AI contributes to making AR applications or interfaces more accurate and reliable. This integration extends to MR as well. For instance, Karaaslan et al., (Karaaslan et al., 2019) demonstrate how combining AI with MR technology can improve infrastructure inspection. Their approach involves attention-guided semi-supervised deep learning (DL) and human-AI collaboration. AI models, which are computational representations of a real world process, are trained on extensive text data from sources like books, articles, reviews, and online conversations (Alessio et al., 2018). The potential of AI in immersive environments is further illustrated by Duricica et al., (Duricica et al., 2024) research. Where an AI assistant is developed to elevate immersive training, which leverages multimodal AI and VR technology to support task execution within industrial environments. The study presents a case of a VR environment simulating a juice mixer setup, where the VR setup replicates the juice mixing process similar to machinery used in pharmaceutical and chemical industries. According to the authors, this setup immerses users in understanding operational principles and functionalities. The multimodal AI assistant, powered by a large language model (LLM), incorporates a speech-to-text model, such as OpenAI's generative pretrained transformer four (GPT-4), to convert audio into text. For example, a user can ask, “What should I do next?” and receive step-by-step guidance.

For this study, the LSTM model was chosen in combination with AR technology due to its clear advantages over traditional RNN. According to Li and zhang (Li and Zhang, 2021), LSTM outperforms other models, particularly in tasks such as text generation and sequence prediction. Based on this, this work considered both Standard LSTM and BiLSTM architectures for training the model, with the high-performance model being selected for integration into the 3D models. The following sections elaborate on the proposed framework.

3 DESIGN AND METHODOLOGY

This work introduces a framework for the dynamic integration of 3D models into an AR environment and incorporates an AI model, to enhance content by augmenting textual information. In developing the AI component, we trained on two LSTM architectures: Standard LSTM and BiLSTM, and then selected the one with the highest accuracy and lowest loss.

The framework consists of three components, as shown in Figure 1: (a) Data preparation and model training (Bottom Layer); (b) Cross-platform communication (Middle Layer); and (c) Data input, API communication and presentation, and animation (Topmost Layer).



Topmost layer
Data input
WWW
Frontend (HTML, CSS, and JavaScript)
Add asset
Backend (MySQL)
Storage
Communication
API (Send 3D model + Asset data) PHP
Presentation and Animation
Front end (Vuforia)
Including 3D structure names
Unity scripts (C#)
(Displays model) (Interacts with model)
Logic Scripts
Middle layer
Cross Platform Communication
DeepLink
AndroidManifest
Bottom layer
Data Preparation and Model Training
Webscraping
Data cleaning and Tokenization
Model architecture
Model training and validation
Generates Text
API (Sends data to unity) Python
AI text Generator
Datasets

Figure 1: Conceptual Framework of the proposed solution.

3.1 Bottom Layer

This layer encompasses key components that constitute the embedded processes required for text generation. The model trained was integrated into the 3D model to generate textual content that enhances the immersive 3D visuals, with content dynamically adapting based on user interactions with various nodes in the 3D model's structure. The following sections outline the steps taken to develop and train the AI model.

3.1.1 Data Collection and Preparation

A. Data Collection. This was done using web scraping such that an asset's (e.g., microscope) textual data were collected from at least 18 URLs using the requests library. The HTML content was parsed using BeautifulSoup to extract relevant textual information from tags such as `<p>`, `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, and `<h6>`. Additionally, four PDF URLs were downloaded through Gdown library and processed with python PDF version 2 (PyPDF2) library. The size of the extracted data was 1.43 MB.

B. Data Preparation. The raw text retrieved was cleaned by removing non-alphabetical characters, symbols, and unwanted formatting. This was then augmented to enhance the diversity of the dataset by considering the synonym replacement technique while preserving meaning. Additionally, a text tokenizer was employed to divide the final dataset into chunks of 100 words, and further split into $n-grams$ (a sequence of n words) to form sequences that were used as inputs to the model to be trained.

- *Text Tokenization:* The text is divided into chunks of 100 words. Furthermore, the chunks are processed to create a word index, which maps each unique word to a unique integer. The vocabulary size was then created.
- *Sequence Creation:* The text is split into $n-grams$ to form sequences using the algorithm implement as shown in algorithm 1, which generates the $n-grams$ for each tokenized chunk, where for each index i, a sequence n_gram_seq is created which includes all tokens from the start of tk_list up to index i.
- *Dataset Splitting:* The final dataset 3.24 MB (388085 words excluding spaces). Lastly, the custom dataset is split into training (60%), test (20%), and validation (20%) sets.

C. Model Architecture. The LSTM model was used as a model architecture because it introduces an intermediate type of storage via the memory cell, which overcomes a key limitation of traditional RNN (i.e., the vanishing gradient problem, by preventing the network from forgetting information throughout a sequence (Lipton, 2015).

Algorithm 1: Sequence Creation from Text Chunks.

```
Data: chunks of text, tokenizer
Result: sequence of input tokens
initialization: input_seq = [ ];
for text in chunks do
    tk_list = tokenizer.text_to_seq([text])[0];
    for i = 1 to len(tk_list) - 1 do
        n_gram_seq = tk_list[:i + 1];
        n_gram_seq to input_seq;
    end
end
return input_seq
```

This section presents two neural network architectures developed to train models for text generation, which are (a) Standard LSTM and (b) BiLSTM. The default structure used in the architectures is as follows:

- *EmbeddingLayer:* This layer maps tokens to dense vectors, making it easier for the network to learn relationships and patterns within the data (Boykis, 2023).
- *LSTMLayers:* This processes sequential captures long-term dependencies in the data.
- *DropoutLayer:* This was incorporated to prevent overfitting (Özgür and Nar, 2020).
- *DenseLayer:* Softmax layer was used for predicting the next word based on input sequences (Chen et al., 2018).

(i) Sequential LSTM

Algorithm 2: Standard LSTM Architecture.

```
Data: total_words, max_sequence_len
Result: Compiled LSTM model
initialization: model = Sequential();
Add Embedding(total_words, n,
  input_length=max_sequence_len - 1);
Add LSTM(n, return_sequences=True);
Add Dropout(0.2);
Add LSTM(n);
Add Dense(total_words, activation =
  'Softmax');
return model
```

(ii) BiLSTM

It is important to highlight that, in algorithm 2, the line 2 layer returns sequences, and the layer in line 4 passes the final output, after processing the sequence forward, to the next layer. In algorithm 3, the layer line 2 processes the input data forward and backward to capture dependencies in both directions of the sequence, and in line 4, the layer passes the final output to the next layer.

Algorithm 3: BiLSTM Architecture.

```
Data: total_words, max_sequence_len
Result: Bidirectional LSTM model
initialization: model = Sequential();
Add Embedding(total_words, n,
  input_length=max_sequence_len - 1);
Add Bidirectional(LSTM(n,
  return_sequences=True));
Add Dropout(0.2);
Add Bidirectional(LSTM(n));
Add Dense(total_words, activation =
  'Softmax');
return model
```

D. Model Training. The training process used an Adaptive Moment Estimation (Adam) optimizer with a default learning rate of 0.001. The model is trained on 60%, tested on 20%, and validated on 20% of the overall data, over 50 epochs with a batch size of 64. Furthermore, the accuracy and loss metric is considered to evaluate training and validation performance.

E. Model Evaluation. The model is evaluated on the test set to determine its generalization capabilities. The evaluation metrics used include test loss and accuracy. Moreover, the perplexity is used to evaluate how strong the model is about its predictions and is calculated from the test loss using the formula in equation (1), where e is the exponential transformation of the average loss.

$$\text{Perplexity} = e^{\text{Loss}} \tag{1}$$

3.1.2 Text Generation Process

To generate text, the implementation of the function algorithm (4) was used. The function generates text by iteratively predicting the next word based on the input (from the 3D Model interaction). Thus, the initial input is iterated by adding predicted words considering 15 as the limit. However, each iteration tokenizes the initial text into numerical sequences and pads them for model input. The temperature sampling 1.0 controls the randomness of the probability distribution for the next word.

Moreover, to facilitate the communication between the text generation based on the model trained, an API was created, where only one parameter is required to process the text generated aspect. Figure 2, illustrates the flow process of how the API interacts

Algorithm 4: Text Generation Process.

```
Data: initial_text_input, model, tokenizer,
      max_sequence_len
Result: Generated text sequence
Parameters: num_words_to_generate = 15,
  temperature = 1.0;
for i = 1 to num_words_to_generate do
    token_seq =
      tokenizer.texts_to_seq(initial_text_input);
    padded_seq = pad_sequence(token_seq,
      max_sequence_len);
    prediction = model.predict(padded_seq,
      temperature);
    next_word =
      tokenizer.index_to_word(prediction);
    initial_text_input += next_word;
end
return initial_text_input
```

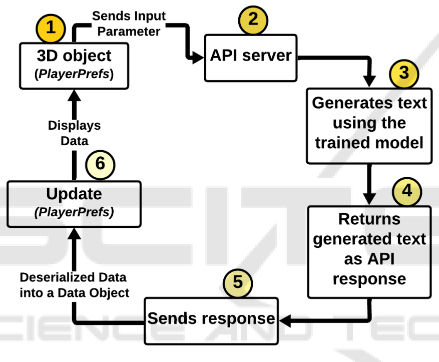

1
Sends Input Parameter
2
API server
3D object (PlayerPrefs)
3
Generates text using the trained model
Displays Data
6
Update (PlayerPrefs)
4
Returns generated text as API response
Deserialized Data into a Data Object
5
Sends response

Figure 2: API interaction with 3D model nodes.

with the 3D model and Text Generation based on the model trained. In this process, the 3D model requests services from the API-based Text Generation based on the model trained. When a user interacts with a node in the 3D model, the system allows the interaction to trigger the extraction of the node name, the action initiates the API call, passing the node name as a parameter. The API subsequently sends a request containing this node name, which is used as *InitialTextInput* on the Text Generation function (illustrated in Algorithm 4). The API submits the request and the server processes it before returning a response in JavaScript Object Notation (JSON) format. This JSON data is then deserialized into a data object, which is stored locally using the PlayerPrefs method (a Unity class for saving small amounts of data locally) and displayed on the user's screen using the TextMesh component. PlayerPrefs was incorporated to help reduce the number of API calls.

3.2 Cross-Platform Communication

This entails the implementation of a feature that enables interaction between the SciAssetHub web application and the Android mobile application using DeepLink (Ma et al., 2018), AndroidManifest, and a dynamic Quick Response (QR) code, as shown in Figure 3. Since Unity does not inherently provide this functionality, these methods are employed to allow the end-user to experience an immersive 3D model from their preferred two-dimensional (2D) representation of the asset on the web asset management platform. The dynamic QR code, which is embedded with preview asset information (3D reference), is scanned to trigger the DeepLink in the web application. This DeepLink sends a request via a Uniform Resource Identifier (URI) to the mobile app containing the downloaded AR-based application. The app with the AndroidManifest feature, processes this request, launching the app and navigating the end-user to the correct section which displays the downloaded 3D model.

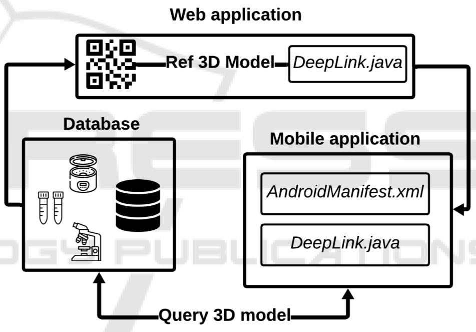


Figure 3: Interaction between web and android platform.

3.3 Topmost Layer

A 3D virtual representation of a physical asset can be uploaded when a user adds a new asset. This can occur either by the end-user directly uploading a relevant 3D model if they have one, or through a system-assisted matching process. If the end-user does not have a 3D model, the system, which has a list of existing 3D models, offers an option to 'attach a 3D model' during the asset addition process. When this option is selected, the system conducts name matching using user name matching (Ren et al., 2021), by comparing the name of the added asset and the names of the existing AR models with different 3D models in the system. If there is a match, the owner of the asset is notified via email with the QR code containing a 3D model of the asset matched and requested to confirm whether the 3D model corresponds to the added asset. The attachment of the representation of the asset added in the form of the 3D model is only authorized upon the owner's confirmation of the match.

When the end-user triggers the immersive mode to view an asset in 3D or AR, the following steps unfold to enable the visualization of the virtual 3D aspect: 3D model retrieval, model instantiation, interactivity setup, touchscreen control capabilities, and selection and highlighting.

3.3.1 3D Model Retrieval

This is where the retrieved model is downloaded onto the end-user's device. The downloaded file is checked to ensure it is in Filmbox (FBX) format and then moved to a specific folder location for effective dependencies management, as other functions depend on this 3D model.

3.3.2 3D Model Instantiation

In the AR mode, virtual 3D models are visualized in the physical environment through the Vuforia SDK, with features such as Plane Finder, which aims to detect and track plane surfaces in the user's real-world environment. To enable this feature in the application, we configured the use of Ground Plane and Plane Finder. Plane Finder detects flat surfaces, and Ground Plane initiates the 3D object in the physical environment.

The AR camera position is set by default to x = 0, y = 0, z = 0, and the same default coordinates (x = 0, y = 0, z = 0) are applied to the position of any 3D object in the physical environment. If the 3D object were placed at a different position, such as x = -0.98, y = 0, z = 0, the user would find that when attempting to place the 3D model in their environment, it would appear misaligned according to spatial registration, showing up at the offset location of x = -0.98, as illustrated in Figure 4, which illustrated the reality of the fact in the real environment. Furthermore, in 3D mode, we considered two options for the camera position, which are as follows:

(i) **Camera Positioning Options**
Option (1): A static view was captured using camera coordinates that visualized 10 different 3D models at varying sizes on two simulator devices (Samsung Galaxy S10e and Apple iPhone 12). The results showed that the 3D models were well displayed, leading us to consider these coordinates (x = 0, y = 5, z = -25) as a universal camera position for all 3D models, with the models instantiated at the default position (0, 0, 0).

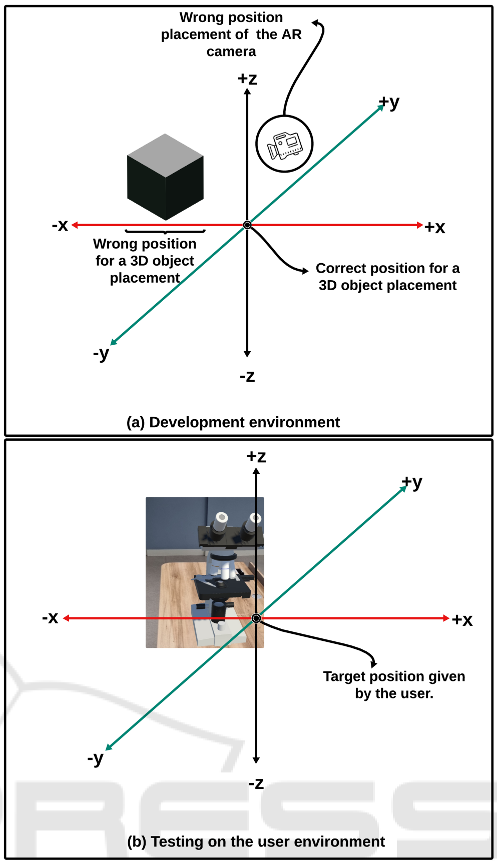

Wrong position placement of the AR camera
+z
+y
-x
+x
Wrong position for a 3D object placement
Correct position for a 3D object placement
-y
-z
(a) Development environment
+z
+y
-x
+x
Target position given by the user.
-y
-z
(b) Testing on the user environment

Figure 4: Spatial registration and positioning of 3D model.

Option (2): This approach combines static and dynamic positioning. Initially, the same static position as in Option (1) was used. Then, its position is dynamically adjusted based on the size and position of the 3D object in world space, specifically accounting for its extent along the z-axis through a developed algorithm. In simpler terms, the camera is automatically positioned along the depth of the 3D object's z-axis, with its height corresponding to the object's width, measured by the distance along the x-axis from the camera.

(ii) **Evaluation of the Ideal Position for the Camera**
The ideal position that was ultimately chosen is option (2) because option (1) did not accurately display some 3D models, leading to inconsistent visualization where some models appeared too far or too close. Option (2) dynamically positions the camera based on calculations using a static scaling factor of 2.5, determined through experimental testing as summarized in Table 1. Table 1 presents the evaluation process used to determine a feasible scaling factor between the camera and

the 3D model. In the Table, "World Transform 3D Object (x, y, z)" represents the world transform vector3 of the 3D model. Additionally, "Bounds: Extent/Midpoint Size 3D Object (x, y, z)" represents the half-dimensions from the center to the edges of each axis of the 3D model. Both "World Transform 3D Object (x, y, z)" and "Bounds: Extent/Midpoint Size 3D Object (x, y, z)" are dynamically retrieved for each 3D model, while the scaling factor remains static. The display output in Table 1 uses values 0.5, 1.0, and 1.5 to represent the camera's position relative to the 3D model:

- 0.5: Camera is too close, making rotation infeasible.
- 1.0: 3D model is well-represented, and rotation is feasible.
- 1.5: Camera is slightly further from the 3D model, but rotation remains feasible.

3.3.3 Interactivity Setup

The main idea behind the interactivity setup is to allow end-users to touch a specific point on the 3D model with their finger, the application should be able to identify and highlight, and then display the name of the structure along with a short description that was touched based on pattern interaction. As illustrated in Figure 5, the application detects touch events on the screen, it then determines which object the end-user intends to tap by mathematically casting a ray from the screen's XY position into 3D space using the camera pose (Linowes and Babilinski, 2017). If the ray intersects a detectable object, the application responds to the tap by, for example, modifying the geometry. To enable this functionality, method functions are automatically attached to the model, to facilitate collision detection between meshes and geometric shapes. The following highlights the relevant functions:

(a) ***MeshCollider.*** This allows collision detection between meshes and geometric shapes.

(b) ***Mesh.MarkDynamic.*** This enables the manipulation of a mesh (a collection of vertices, edges, and faces that define the shape of the 3D model).

(c) ***Transform.LookAt.*** This auto rotates the camera to face the 3D model.

(d) ***TagName.*** This is where tagnames are assigned to organize the model components, such as parent and child nodes or parts.

3.3.4 Touchscreen Control Capabilities

The concept of the touch input method is added to allow the screen to facilitate interaction with the model by detecting and responding to user interactions based on different touch input features. We used the $Input.GetTouch(n)$ function, where n indicates the number of fingers on the screen, and applied the logic statement to manage the number of fingers required for triggering interactions with the 3D model.

3.3.5 Selection and Highlighting

As illustrated in Figure 5, selection and highlighting allow a node within or part of a model to be selected and visually highlighted. When a node is selected, it changes to a default color set in the system. If another node is selected, the previous node reverts to its original color, and the newly selected node is highlighted in the default color. This process repeats with each new selection.

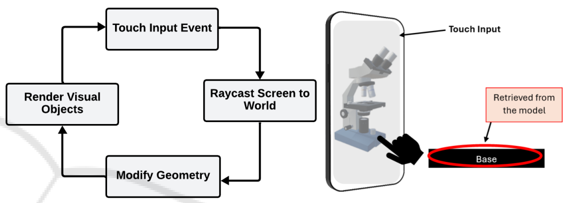


Figure 5: Touch interaction and ray casting in 3D (Best viewed in color mode).

4 RESULTS AND DISCUSSION

The proposed framework was incorporated into the SciAssetHub asset management web-based system to assess its performance. The development process involved the use of Vuforia software development kit (SDK) version 10.21.3, Unity version 2020.3.15f1, PHP 8.2.13, and Python 3.12.1. Furthermore, in this study, the 3D models were purchased from Turbosquid, and optimized using 3ds Max 2024. The entire system was developed on a desktop running Windows 11, with 16GB of RAM, 453GB of storage, and an Intel Core i7 processor. Development tools included Visual Studio Code version 1.90.0 and Unity 3D engine version 2023.2.7f1.

4.1 Performance of Trained Model

The model was trained using neural network architecture including an embedding layer, LSTM recurrent layers, and dense layers. The training was configured for 50 epochs with a batch size of 64.The optimizer used was Adam, with a learning rate initialized at 0.001, and a dropout rate of 0.2. During the experiment, both training and validation accuracy and loss were captured. The following section presents

Table 1: Evaluation vector3 representation of the camera and 3D model.

World Transform 3D object (x,y,z)	Extent / midpoint size 3D object (x,y,z)	Midpoint: Camera z position (height)	Generated: Center Camera (x,y,z)	Scaling factor	Display output
-0.55, 6.57, 1.07	3.45, 6.60, 4.36	4.36	-0.55, 6.57, -3.29	0.5	0.5
-0.55, 6.57, 1.07	3.45, 6.60, 4.36	4.36	-0.55, 6.57, -7.65	1	0.5
-0.55, 6.57, 1.07	3.45, 6.60, 4.36	4.36	-0.55, 6.57, -12.01	1.5	0.5
-0.55, 6.57, 1.07	3.45, 6.60, 4.36	4.36	-0.55, 6.57, -16.37	2	0.5
-0.55, 6.57, 1.07	**3.45, 6.60, 4.36**	**4.36**	**-0.55, 6.57, -20.73**	**2.5**	**1**
-0.55, 6.57, 1.07	3.45, 6.60, 4.36	4.36	-0.55, 6.57, -25.09	3	1.5

the result of the performance of standard LSTM and BiLSTM models based on the training outcomes.

4.1.1 Standard LSTM

Figure 6 shows the training loss, it can be observed that the starting loss at Epoch 1 was quite high at 7,09; which indicates that the model struggled to learn at first. However, as the model progressed, the loss steadily decreased, reaching 0.12 by Epoch 50, which suggests that the model learned from the dataset.

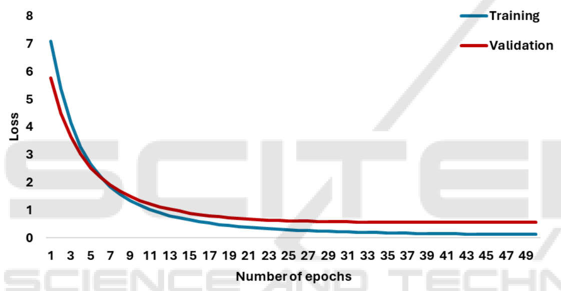


Figure 6: Standard LSTM: Training and validation loss over epochs.

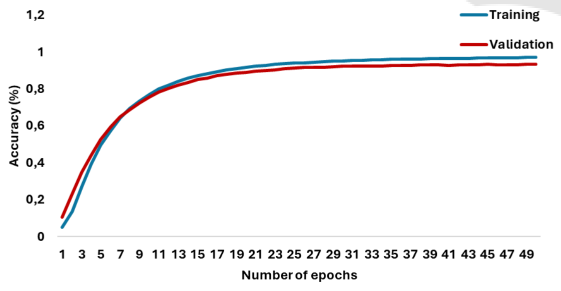


Figure 7: Standard LSTM: Training and validation accuracy over epochs.

Furthermore, Figure 7 presents the validation loss, which followed a similar trend, starting at 5,76 and ending at 0,57. The training accuracy rose from 0,05 in the early epoch to 0,97 in the final epoch. The validation accuracy started at 0,10 and ended at 0,93. It can be said that the model was learning and adapting to the training data. Additionally, the perplexity score stood at 1.69, with a test accuracy of 93.55% and a test loss of 0.52.

4.1.2 BiLSTM

Figure 8 presents the training loss for the BiLSTM architecture. At the start, the loss is relatively low, at 6.09, compared to the Standard LSTM in Figure 6, indicating that the model began with slightly better learning ability.

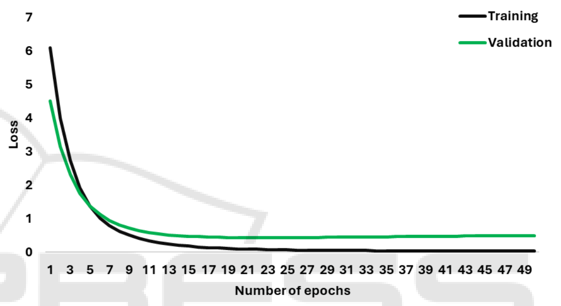


Figure 8: BiLSTM: Training and validation loss over epochs.

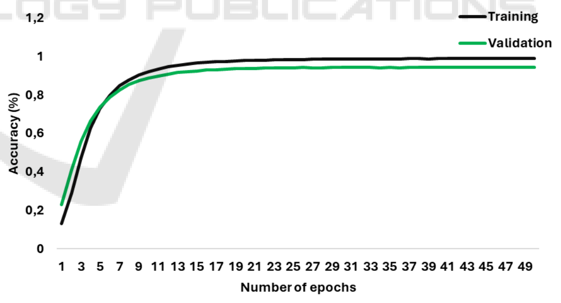


Figure 9: BiLSTM: Training and validation accuracy over epochs.

The loss steadily decreased as training progressed, reaching 0.04 by epoch 50, suggesting that the model effectively learned from the dataset. Similarly, the validation loss followed this trend, starting at 4.51 and ending at 0.49, which is notably better than the results from the Standard LSTM. The training accuracy, shown in Figure 9, rose from 0.13 early on to 0.99 at the final epoch. The validation accuracy also improved, starting at 0.23 and ending at 0.94. Additionally, the test accuracy is 94.35% and a test loss of 0.51.

Due to the superior performance demonstrated by the model trained with BiLSTM architecture, in terms of training and validation loss, accuracy, and test accuracy. The model was selected for integration into the 3D models in this work.

4.2 3D Preparation and Interaction

The core functionality of the framework also involves its capability to retrieve and visualize various 3D models in two main modes: 3D and AR. However, the development of the AR mode is similar to the 3D mode but requires access to the device's camera to detect a flat surface for placing the 3D model in the real world. This means that the development is slightly different. In 3D mode, the first step involves implementing the script for touch input and the script to instantiate the 3D model. Following this, scripts for MeshCollider, Mesh.MarkDynamic, TagNames, selection and highlighting, and rotation are created to enable interactivity with the 3D models. In AR mode, Vuforia is installed first, followed by setting up the Plane Finder and Ground Plane functionalities. A script was then created to instantiate the downloaded 3D model as a child of the ground plane. Following this, the same scripts are used in 3D mode, such as touch input, MeshCollider, Mesh.MarkDynamic, TagNames, highlighting, and rotation, were also applied in AR mode.

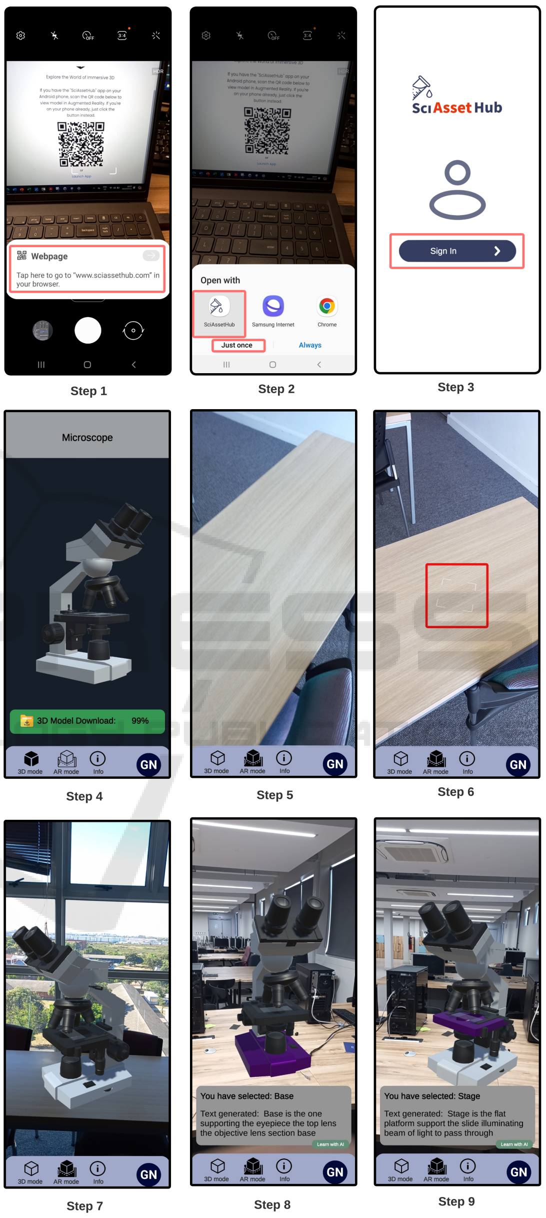
4.2.1 Data Collection Layer

The option to allow end-users to upload and store 3D models, with necessary asset information, was integrated into the asset management system. The system accepts only the FBX file format extension.

- *3D Node Structure:* This involved partitioning 3D models into sub-parts based on their names, assigning or renaming essential components, following the approach used by Manith et al. (Manith et al., 2019).

4.2.2 Presentation and Animation Layer

The immersive aspect aims to provide better visualization of the virtual representation of physical assets with interactive capabilities. Figure 10 outlines the steps to activate the immersive element using the mobile SciAssetHub application. Before starting, users should ensure they have downloaded the mobile immersive SciAssetHub application by opening the QR code reader on their device, such as the Android camera app. Users can scan the QR code displayed on their devices to download the SciAssetHub mobile application, or they can simply click the "Download App" button if they have accessed SciAssetHub through their mobile devices. Once downloaded, users should proceed to install the application on their devices.



You have selected: Base

Text generated: Base is the one supporting the eyepiece the top lens the objective lens section base

You have selected: Stage

Text generated: Stage is the flat platform support the slide illuminating beam of light to pass through

Figure 10: Steps to engage the immersive elements in the platform (Best viewed in colour mode).

Next, as shown in Step 1 of Figure 10, an end user can scan the QR code on the right as displayed on the desktop website using a preferred QR code reader, or by simply clicking on the "Launch App". To retrieve the information on the immersive 3D model to visualize the application, the end-user can click on the pop-up model displayed, as shown in Step 2, and select the "SciAssetHub" application as an option and choose "Just once" to launch and access the immersive element based on the equipment they want to view. The 3D model will auto-download to the end user's device, and the necessary elements will be attached to the 3D model. In Step 3, this is the welcome screen; the screen provides the option to experience the immersive feature in either "3D mode" by clicking on "3D mode," which will redirect them to the respective screen of the feature. Alternatively, clicking "AR mode" will take them to the screen for the AR feature. The AR mode application accesses the device's camera. Once the camera is open (Step 5), it detects a flat surface (plane detection). When a flat surface is detected (Step 6), the user should tap on the screen to place the 3D object in the virtual environment. The model will then superimpose on the environment based on the spatial information (Step 7). To interact with the 3D model and trigger the AI-augmented textual information, the user must tap or touch the 3D model's node structure. The touched node will be selected and highlighted in purple, as shown in Steps 8 and 9 in Figure 10, and information about that part will be displayed.

5 SUMMARY AND FUTURE RECOMMENDATIONS

This paper presents a framework that facilitates the dynamic integration of 3D models into an AR environment in asset management system, enhanced by AI (trained model using BiLSTM architecture) to augment textual information based on the interaction with the node structure of the 3D model. In this study, the Standard LSTM and BiLSTM architectures were used and demonstrated promising results in terms of training, validation performance, and test accuracy. However, the BiLSTM model showed superior performance, with a higher test accuracy (94.35%) and lower validation loss (0.51) compared to the standard LSTM, which had a test accuracy of 93.55% and a loss of 0.52. The proposed AR-based framework was successfully developed and assessed for its effectiveness. The framework produced a good result, which means that it addressed scalability challenges in the asset management system. It ensures that the 3D models are accurately framed within the camera's view in both AR and 3D modes, enhancing the overall visualization. This framework can be applied across various domains that involve the visualization of virtual assets or objects, such as education, retail, construction, healthcare, and more. In the future, a larger and more structured dataset could be considered, potentially using supervised or semi-supervised learning approaches. Moreover, the use of AI is expected to further enhance 3D image processing.

ACKNOWLEDGMENTS

This work was supported in part by UWC and MasterCard Foundation.

REFERENCES

Abdulhamied, R. M., Nasr, M. M., and Abdulkader, S. N. (2023). Real-time recognition of american sign language using long-short term memory neural network and hand detection. *Indonesian Journal of Electrical Engineering and Computer Science*, 30(1):545556.

Abujar, S., Masum, A. K. M., Chowdhury, S. M. H., Hasan, M., and Hossain, S. A. (2019). Bengali text generation using bi-directional rnn. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–5. IEEE.

Alessio, H. M., Malay, N., Maurer, K., Bailer, A. J., and Rubin, B. (2018). Interaction of proctoring and student major on online test performance. *International Review of Research in Open and Distributed Learning*, 19(5).

Alves, C. and Luís Reis, J. (2020). The intention to use e-commerce using augmented reality-the case of ikea place. In *Information Technology and Systems: Proceedings of ICITS 2020*, pages 114–123. Springer.

Asta, N. P. R. N., Setiawan, S., Saputra, M., Najmuddin, N., Bedra, K. G., et al. (2024). Integrating augmented reality with management information systems for enhanced data visualization in retail. *Journal of Social Science Utilizing Technology*, 2(2):191–201.

Bassyouni, Z. and Elhajj, I. H. (2021). Augmented reality meets artificial intelligence in robotics: A systematic review. *Frontiers in Robotics and AI*, 8:724798.

Boykis, V. (2023). What are embeddings. *10.5281/zenodo*, 8015029(1–13).

Braun, P., Grafelmann, M., Gill, F., Stolz, H., Hinckeldeyn, J., and Lange, A.-K. (2022). Virtual reality for immersive multi-user firefighter training scenarios. *Virtual reality & intelligent hardware*, 4(5):406–417.

Chen, P. H., Si, S., Kumar, S., Li, Y., and Hsieh, C.-J. (2018). Learning to screen for fast softmax inference on large vocabulary neural networks. *arXiv preprint arXiv:1810.12406*.

Cipresso, P., Giglioli, I. A. C., Raya, M. A., and Riva, G. (2018). The past, present, and future of virtual and augmented reality research: a network and cluster analysis of the literature. *Frontiers in psychology*, 9:2086.

Datta, P., Kaur, A., Sassi, N., Gulzar, Y., and Jaziri, W. (2024). An evaluation of intelligent and immersive digital applications in eliciting cognitive states in humans through the utilization of emotiv insight. *MethodsX*, 12:102748.

Duricica, T., Müllnera, P., Weidingera, N., ElSayeda, N., Kowalda, D., and Veasa, E. (2024). Ai-powered immersive assistance for interactive task execution in industrial environments. *environment*, 28:2.

Dyulicheva, Y. Y. and Glazieva, A. O. (2021). Game based learning with artificial intelligence and immersive technologies: An overview. *CS&SE@ SW*, pages 146–159.

Garramone, M., Tonelli, E., Scaioni, M., et al. (2022). A multi-scale bim/gis framework for railways asset management. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 46(W1):95–102.

Garzón, J. (2021). An overview of twenty-five years of augmented reality in education. *Multimodal Technologies and Interaction*, 5(7):37.

Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., Qiu, J., Yao, Y., Zhang, A., Zhang, L., et al. (2021). Pre-trained models: Past, present and future. *AI Open*, 2:225–250.

Hussein, M. A. H. and Savaş, S. (2024). Lstm-based text generation: A study on historical datasets. *arXiv preprint arXiv:2403.07087*.

Ibne Akhtar, N., Mohimenul Islam Shazol, K., Rahman, R., and Abu Yousuf, M. (2021). Bangla text generation using bidirectional optimized gated recurrent unit network. In *Proceedings of International Conference on Trends in Computational and Cognitive Engineering: Proceedings of TCCE 2020*, pages 103–112. Springer.

Karaaslan, E., Bagci, U., and Catbas, F. N. (2019). Artificial intelligence assisted infrastructure assessment using mixed reality systems. *Transportation Research Record*, 2673(12):413–424.

Koumou, K. O. and Isafiade, O. (2024). Asset management trends in diverse settings involving immersive technology: A systematic literature review. *IEEE Access*, 12:141785–141813.

Koumou, K. O., Isafiade, O., Kotze, R. C., and Ekpo, O. E. (2023). Fostering research asset management and collaboration using publish-subscribe and immersive technologies. In *Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2023*, pages 119–124.

Li, L. and Zhang, T. (2021). Research on text generation based on lstm. *International Core Journal of Engineering*, 7(5):525–535.

Limna, P., Jakwatanatham, S., Siripipattanakul, S., Kaewpuang, P., and Sriboonruang, P. (2022). A review of artificial intelligence (ai) in education during the digital era. *Advance Knowledge for Executives*, 1(1):1–9.

Linowes, J. and Babilinski, K. (2017). *Augmented reality for developers: Build practical augmented reality applications with unity, ARCore, ARKit, and Vuforia*. Packt Publishing Ltd. ISBN-13: 978-1787286436.

Lipton, Z. C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv Preprint, CoRR, abs/1506.00019*, pages 1–38.

Ma, Y., Hu, Z., Liu, Y., Xie, T., and Liu, X. (2018). Aladdin: Automating release of deep-link apis on android. In *Proceedings of the 2018 World Wide Web Conference*, pages 1469–1478.

Manith, E., Park, C., and Yoo, K.-H. (2019). A hierarchical structure for representing 3d respiration organ models. In *Big Data Applications and Services 2017: The 4th International Conference on Big Data Applications and Services 4*, pages 23–36. Springer.

Özgür, A. and Nar, F. (2020). Effect of dropout layer on classical regression problems. In *2020 28th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE.

Ozturkcan, S. (2021). Service innovation: Using augmented reality in the ikea place app. *Journal of Information Technology Teaching Cases*, 11(1):8–13.

Rampini, L., Cecconi, F. R., et al. (2022). Artificial intelligence in construction asset management: A review of present status, challenges and future opportunities. *Journal of Information Technology in Construction*, 27:884–913.

Ren, J., Xia, F., Chen, X., Liu, J., Hou, M., Shehzad, A., Sultanova, N., and Kong, X. (2021). Matching algorithms: Fundamentals, applications and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(3):332–350.

Sepasgozar, S. M. (2020). Digital twin and web-based virtual gaming technologies for online education: A case of construction management and engineering. *Applied Sciences*, 10(13):4678.

Suh, A. and Prophet, J. (2018). The state of immersive technology research: A literature analysis. *Computers in Human behavior*, 86:77–90.

Wang, X., Love, P. E., Kim, M. J., Park, C.-S., Sing, C.-P., and Hou, L. (2013). A conceptual framework for integrating building information modeling with augmented reality. *Automation in construction*, 34:37–44.

Zhang, C. and Lu, Y. (2021). Study on artificial intelligence: The state of the art and future prospects. *Journal of Industrial Information Integration*, 23:100224.