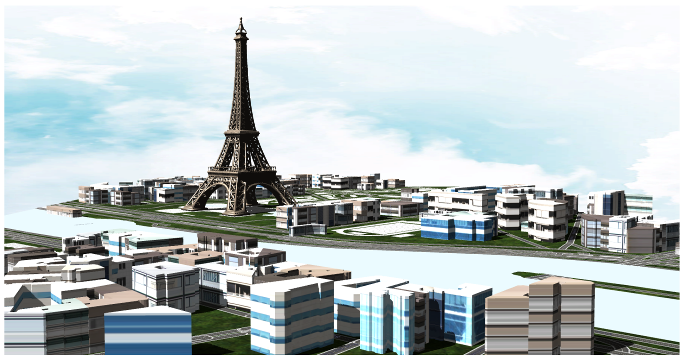
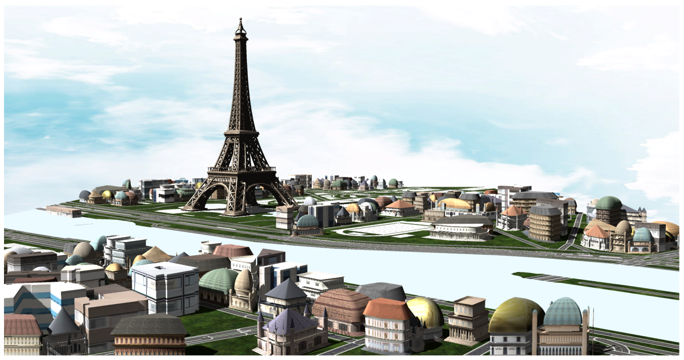
Synthesizing Virtual World Palace Scenes on OpenStreetMap

Wanwan Li
University of South Florida
Tampa, Florida, USA
wanwan@usf.edu



(a) Synthesized Modern Urban Scene on OpenStreetMap



(b) Synthesized World Palace Scene on OpenStreetMap

Fig. 1. This figure shows the experiment results from our proposed procedural modeling approach that automatically synthesizes virtual world palace scenes (b) from OpenStreetMap data (a). Location presented in this result is Eiffel Tower in Paris, France.

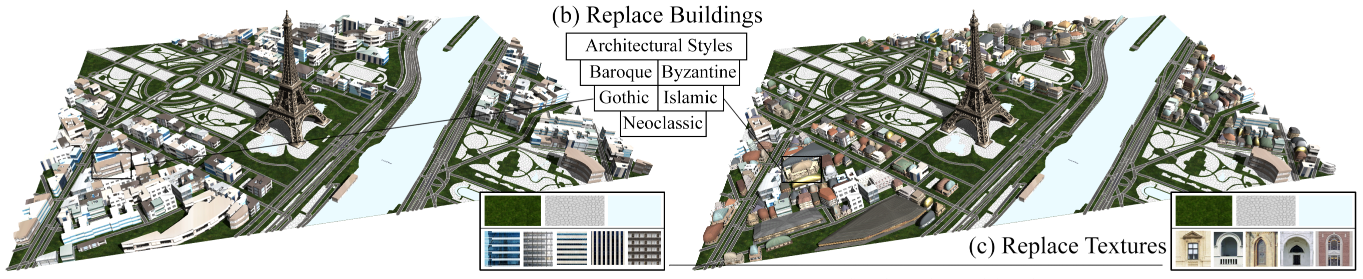
***Abstract*—Virtual worlds have gained significant popularity in recent years, offering immersive experiences and vast opportunities for various applications. With the rapid advancement of computer graphics and virtual reality technologies, the demand for realistic and immersive virtual environments has significantly increased. One critical aspect of virtual world creation is the procedural generation of realistic scenes. Recently, researchers proposed some automatic virtual urban environment synthesis approaches based on OpenStreetMap (OSM), which is a widely-used collaborative mapping platform that provides rich and diverse geospatial datasets. In this paper, we propose a novel technical approach for synthesizing virtual world palace scenes using OSM data. By applying geometry extraction, geometric modeling, and texturing techniques, our proposed approach is able to leverage OSM's geometric information to generate realistic, immersive, and highly detailed 3D palaces with different architectural styles in the world. In the end, we test our approach to convert modern urban scenes into world palace scenes with different architectural styles within the same urban layout.**

***Keywords*—Procedural Modeling, OpenStreetMap (OSM)**

I. Introduction

As the advancement and increasing popularity of Virtual Reality (VR) technologies are opening up new possibilities for creating virtual worlds for various domains including entertainment, education, tourism, architecture, etc., there is a growing demand for realistic virtual world environments. Virtual worlds have gained significant popularity in recent years, offering immersive experiences and diverse environments, and allowing users to explore and interact with computer-generated environments that replicate real-world or fictional settings. Creating visually appealing and realistic scenes within virtual worlds is crucial to enhancing user immersion and engagement. However, manually creating realistic virtual environments, such as palaces, is a nontrivial task that requires heavy manual efforts from content designers to deliver an accurate representation of 3D models, textures, and lighting. Therefore, synthesizing realistic palace scenes within virtual worlds poses a unique challenge due to the complexity and intricacy of architectural designs.

Recently, by leveraging the rich dataset and geospatial information available in the OpenStreetMap (OSM) dataset [1]–[6] and combining it with advanced computer graphics techniques, researchers have successfully proposed technical approaches to automatically synthesize virtual urban scenes on OSM. For example, in 2021, Li et al [7] synthesized virtual urban scenes on OSM for Uber schedules optimization. In 2022, Li et al. [8] synthesized virtual construction scenes on OSM, Li et al. [9] synthesized virtual urban scenes on OSM for conceptual architecture design in VR. In 2023, Li et al. [10] synthesized virtual urban scenes on OSM for VR treadmill exergaming, Li et al. [11] synthesized virtual urban scenes on OSM for optimizing Alternating Reality Games (ARG). In the same year, Li et al. [12] synthesized virtual urban night scenes on OSM. The most relevant work is the approach for synthesizing virtual Chinese palace scenes on OSM proposed by Li et al. [13]. However, no existing work has been proposed for synthesizing the virtual world palace scene on OSM. Therefore, given this observation, in this paper, we present a novel approach to automatically synthesize immersive, highly detailed, and visually compelling virtual world palace scenes on OSM that capture the essence of prominent and culturally significant architectural structures found worldwide.



(a) Virtual Urban Scene on OSM (d) Virtual World Palace Scene on OSM

Fig. 2. Overview of our approach.

II. Overview

Figure 2 shows the overview of our approach. Given a real-world location on the OSM, the Unity3D game engine can automatically download the OSM data through Google Map API. OSM data contains the urban environment features such as buildings, roads, land, and water. Then, through an open-source Unity 3D asset, GO Map [14], OSM data can be constructed as a 3D model of the urban layout. After applying realistic textures to this generated scene, a virtual urban scene is procedurally generated on OSM as shown in Figure 2 (a). Then, for each building in the urban scene, we apply a series of procedural modeling steps to replace the original modern buildings into world palaces with different styles. As shown in Figure 2 (b), procedural modeling approaches are designed to construct world palaces with five architectural styles including Baroque, Byzantine, Gothic, Islamic, and Neoclassic. For each style, the procedural modeling process includes adding steps and doors, repeatedly adding window and eave for each floor, and adding roof for the highest floor. In the end, after updating textures for the world palaces as shown in Figure 2 (c), the output of realistic virtual world palaces scenes is procedurally synthesized on OSM as shown in Figure 2 (d).

III. Technical Approach

According to the OSM Google Map API, any modern building geometry is an extruded geometry from a polygon. Mathematically, each building b_i in the building cluster $B = \{b_1, b_2, ...\}$, there is $b_i = \mathbf{P}_i \uparrow h_i$ where polygon $\mathbf{P}_i = \{\mathbf{p}_1, \mathbf{p}_2, ...\}$, building height is h_i, and $\uparrow$ denotes the extrusion operation along the y-axis. Our proposed approach automatically generates world palaces with different architectural styles according to this polygon $\mathbf{P}_i$ so as to replace the original modern building b_i into a palace building b_i'. Figure 3 shows the building geometry procedurally generated with our approach for each architectural style. In this example, according to the modern building shown in Figure 3 (a) which is extruded from a rectangle, the input of these five synthesized results of world palaces is a rectangle. More specifically, the architectural styles of synthesized world palaces include (b) Baroque, (c) Byzantine, (d) Gothic, (e) Islamic, and (f) Neoclassic. In this section, we will introduce the mathematical representations and our technical approach for synthesizing procedural world palaces with these five architectural styles.

Baroque Palace. Given arbitrary polygon $\mathbf{P}_i$, Baroque palace geometry b_i' can be procedurally generated using the Boolean OR ($\vee$) operation as shown in the equation below:

$$b_i' = f_{\mathrm{s}}^{0}(\mathbf{P}_i) \vee \xi_{\mathrm{d}}^{0}(\mathbf{P}_i) \vee f_{\mathrm{e}}^{1}(\mathbf{P}_i) \vee \chi_{\mathrm{f}}^{N_i}(\mathbf{P}_i) \vee f_{\mathrm{r}}^{N_i+1}(\mathbf{P}_i) \quad (1)$$

where $f_{\mathrm{s}}^{*}(\mathbf{P}_i)$ is the steps geometry, $\xi_{\mathrm{d}}^{*}(\mathbf{P}_i)$ is the door geometry, $f_{\mathrm{e}}^{*}(\mathbf{P}_i)$ is the eave geometry, $f_{\mathrm{r}}^{*}(\mathbf{P}_i)$ is the roof geometry, and $\chi_{\mathrm{f}}^{N_i}(\mathbf{P}_i)$ is the floors geometry calculated as:

$$\chi_{\mathrm{f}}^{N_i}(\mathbf{P}_i) = \bigvee_{j=1}^{N_i} \left(\xi_{\mathrm{w}}^{j}(\mathbf{P}_i) \vee f_{\mathrm{e}}^{j+1}(\mathbf{P}_i) \right)$$

where $\xi_{\mathrm{w}}^{*}(\mathbf{P}_i)$ is the window geometry, the total number of floors is $N_i = \lfloor (h_i - h_{\mathrm{d}} - h_{\mathrm{r}})/h_{\mathrm{w}} \rfloor$ and h_{d}, h_{r}, h_{w} are the heights of door, roof, and window respectively. For Baroque palace, $h_{\mathrm{d}} = 10$, $h_{\mathrm{r}} = 15$, $h_{\mathrm{w}} = 8$. Door / window geometry $\xi_{\mathrm{d/w}}^{j}(\mathbf{P}_i)$ is calculated as:

$$\xi_{\mathrm{d/w}}^{j}(\mathbf{P}_i) = \left(\bigcup_{k=1}^{|\mathbf{P}_i|} \left(\mathbf{p}_k + w_{\mathrm{e}}(\mathbf{c}_i - \mathbf{p}_k) + \mathbf{y}_{d/w}^{j} \right) \right) \uparrow h_{\mathrm{d/w}}$$

where y-axis offset $\mathbf{y}_{d/w}^{j} = (0, h_{\mathrm{s}}/h_{\mathrm{s}} + h_{\mathrm{d}} + (j-1)h_{\mathrm{w}}, 0)$.

Parametric Extrusion. In Equation 1 and Equation III, steps geometry $f_{\mathrm{s}}^{*}(\mathbf{P}_i)$, eave geometry $f_{\mathrm{e}}^{*}(\mathbf{P}_i)$, and roof geometry $f_{\mathrm{r}}^{*}(\mathbf{P}_i)$ are represented using parametric extrusion operation. For simplicity, we use $f_{*}^{j}(\mathbf{P}_i)$ to represent the parametric extrusion with the height function $h_*(t)$ at the j^{th} floor where $t \in [0, 1]$. Then, the $M \times |\mathbf{P}_i|$ vertices of the triangle mesh for the parametric extrusion $f_{*}^{j}(\mathbf{P}_i)$ are represented through the point set calculated with the following equation:

$$f_{*}^{j}(\mathbf{P}_i) = \bigcup_{m=0}^{M-1} \bigcup_{n=1}^{|\mathbf{P}_i|} \mathbf{p}_n + t(\mathbf{c}_i - \mathbf{p}_n) + \mathbf{h}_{*}^{j}(t)$$

where $t = m/(M-1)$, $\mathbf{c}_i$ is the mass center of the polygon $\mathbf{P}_i$, and $\mathbf{h}_{*}^{j}(t) = (0, h_*(t), 0) + \mathbf{y}_{*}^{j}$. In our approach, the mesh precision M is set to 100. Height function for steps is: $h_{\mathrm{s}}(t) = \lfloor nt/w_{\mathrm{s}} \rfloor h_{\mathrm{s}}/n$, if $t < w_{\mathrm{s}}$; Otherwise, $h_{\mathrm{s}}(t) = h_{\mathrm{s}}$. We set steps count $n = 3$, steps width $w_{\mathrm{s}} = 1.5$ and steps height $h_{\mathrm{s}} = 3$. Height function for roof is: $h_{\mathrm{r}}(t) = h_{\mathrm{r}} t/w_{\mathrm{r}}$, if $t < w_{\mathrm{r}}$; Otherwise, $h_{\mathrm{r}}(t) = h_{\mathrm{r}}$. We set roof width $w_{\mathrm{r}} = 2.5$ and roof height $h_{\mathrm{r}} = 15$. Height function for eave is: $h_{\mathrm{e}}(t) = (1 - \lfloor nt/w_{\mathrm{e}} \rfloor /n)h_{\mathrm{e}}$, if $t < w_{\mathrm{e}}$; Otherwise, $h_{\mathrm{e}}(t) = h_{\mathrm{e}}$. We set $n = 3$, eave width $w_{\mathrm{e}} = 1$, and eave height $h_{\mathrm{e}} = 3$.

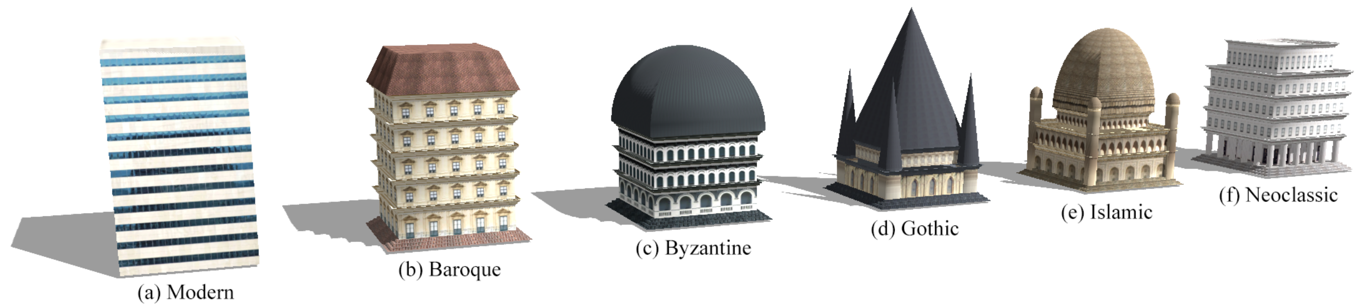


Fig. 3. Building Geometry. This figure shows the building geometry procedurally generated with our approach for different styles of palaces. According to the modern building shown in (a), the input polygon of these synthesized palaces is a rectangle. These architectural styles of synthesized palaces include (b) Baroque, (c) Byzantine, (d) Gothic, (e) Islamic, and (f) Neoclassic.

Byzantine Palace. Similar to Baroque palace geometry, Byzantine palace geometry is also procedurally generated using Equation 1 and Equation III. However, unlike Baroque palace whose roof is flat, rather, Byzantine palace has spherical roof, therefore, Byzantine palace's height function for roof is:

$$h_\text{r}(t) = \sin\left(cos^{-1}(t-1)\right) h_\text{r} \tag{2}$$

Parameter settings for Byzantine palace are: $w_\text{s} = 1.5$, $h_\text{s} = 3$, $h_\text{d} = 10$, $w_\text{e} = 1$, $h_\text{e} = 3$, $w_\text{r} = 2.5$, $h_\text{r} = 30$, $h_\text{w} = 8$.

Gothic Palace. Unlike Baroque and Byzantine palace's roofs, Gothic palace's roof is neither flat nor spherical, rather, it is sharp. Therefore, Gothic palace's height function for roof is $h_\text{r}(t) = th_\text{r}$. Then, Gothic palace geometry is procedurally generated using the following equation:

$$b'_i = f_\text{s}^0(\mathbf{P}_i) \vee \xi_\text{d}^0(\mathbf{P}_i) \vee f_\text{e}^1(\mathbf{P}_i) \vee \chi_\text{f}^{N_i-1}(\mathbf{P}_i) \vee \psi_\text{f}^{N_i}(\mathbf{P}_i) \tag{3}$$

where the last floor geometry $\psi_\text{f}^{N_i}(\mathbf{P}_i)$ is calculated as:

$$\psi_\text{f}^{N_i}(\mathbf{P}_i) = k_\text{f}\left(\xi_\text{w}^{N_i}(\mathbf{P}_i) \vee f_\text{e}^{N_i+1}(\mathbf{P}_i) \vee f_\text{r}^{N_i+1}(\mathbf{P}_i)\right) \vee \tau_\text{t}^0(\mathbf{P}_i)$$

where last floor scale $k_\text{f} = 0.7$ and tower building geometry $\tau_\text{t}^0(\mathbf{P}_i)$ at corners is calculated with the following equation:

$$\tau_\text{t}^0(\mathbf{P}_i) = k_\text{t} \bigvee_{n=1}^{|\mathbf{P}_i|} \left(\xi_\text{d'}^0(\mathbf{Q}^n(\mathbf{P}_i)) \vee f_\text{e'}^1(\mathbf{Q}^n(\mathbf{P}_i)) \vee f_\text{r'}^1(\mathbf{Q}^n(\mathbf{P}_i))\right)$$

where tower scale $k_\text{t} = 0.3$, tower door height $h_\text{d'} = \max(0, h_i - h_\text{d} - h_\text{r}) + h_\text{d}$, tower eave / roof elevation $\mathbf{y}^1_{e'/f'} = (0, h_\text{w} + h_\text{d'}, 0)$, and $\mathbf{Q}^n(\mathbf{P}_i)$ is corner quad geometry for the n^th vertex in polygon $\mathbf{P}_i$ which is calculated as:

$$\mathbf{Q}^n(\mathbf{P}_i) = \{\mathbf{q}_n, \mathbf{p}_n, \mathbf{p}_{n+1}, \mathbf{p}_{n+1} + \mathbf{q}_n - \mathbf{p}_n\}$$

where $\mathbf{q}_n = \mathbf{p}_n + \lambda(\mathbf{p}_{n-1} - \mathbf{p}_n)$ and scale factor $\lambda = 2.5w_\text{e}$. Parameter settings for Gothic palace are: $w_\text{s} = 1.5$, $h_\text{s} = 3$, $h_\text{d} = 10$, $w_\text{e} = 1$, $h_\text{e} = 3$, $w_\text{r} = 2.5$, $h_\text{r} = 30$, $h_\text{w} = 8$. Figure 4 shows an example of a Gothic palace procedurally generated using our approach with the input polygon of a pentagon. This synthesized Gothic palace consists of a main palace building surrounded by five tower buildings generated for five vertices of the pentagon. Our approach for synthesizing tower buildings results in a visually captivating Gothic palace.

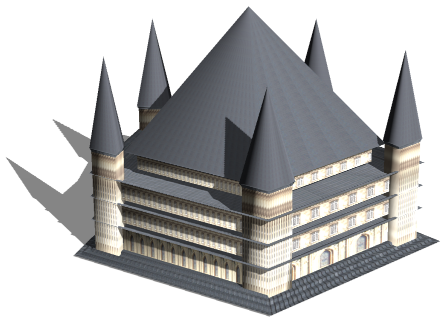


Fig. 4. Gothic Palace Geometry. This figure shows an example of a Gothic palace procedurally generated using our approach with the input polygon of a pentagon. There are five tower building geometry generated for five vertices in the pentagon.

Islamic Palace. Similar to Gothic palace geometry that has tower buildings, Islamic palace geometry is also procedurally generated using Equation 3. Unlike a Gothic palace whose roof is sharp, Islamic palace has a spherical roof. Therefore, Islamic palace's height function for roof is the same as the Byzantine palace's height function for roof as shown in Equation 2. Parameter settings for Islamic palace are: $w_\text{s} = 1.5$, $h_\text{s} = 3$, $h_\text{d} = 10$, $w_\text{e} = 1$, $h_\text{e} = 3$, $w_\text{r} = 2.5$, $h_\text{r} = 40$, $h_\text{w} = 8$.

Neoclassic Palace. We extend our procedural generation approach to create a Neoclassic palace geometry with pillars, building upon the previous discussion of synthesizing other types of palaces. Unlike other palaces' roofs, Neoclassic palace's roof is plane and its geometry can be procedurally generated using the following equation:

$$b'_i = f_\text{s}^0(\mathbf{P}_i) \vee \xi_\text{d}^0(\mathbf{P}_i) \vee f_\text{e}^1(\mathbf{P}_i) \vee \chi_\text{f}^{N_i-1}(\mathbf{P}_i) \vee \eta_\text{f}^{N_i}(\mathbf{P}_i) \tag{4}$$

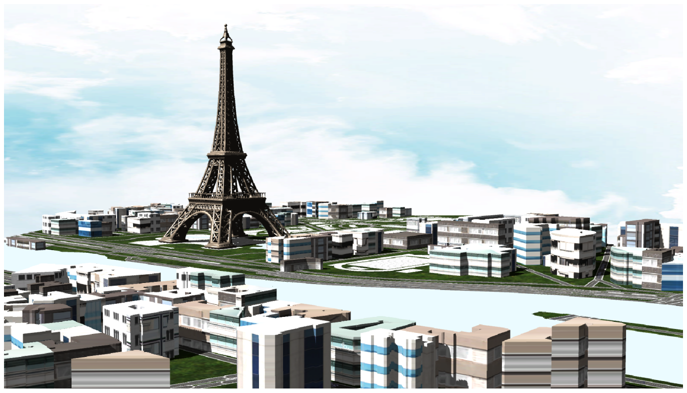
where the last floor geometry $\eta_\text{f}^{N_i}(\mathbf{P}_i)$ is calculated as:

$$\eta_\text{f}^{N_i}(\mathbf{P}_i) = k_\text{f}\left(\xi_\text{w}^{N_i}(\mathbf{P}_i) \vee f_\text{e}^{N_i+1}(\mathbf{P}_i)\right) \vee \zeta_\text{p}^0(\mathbf{P}_i)$$

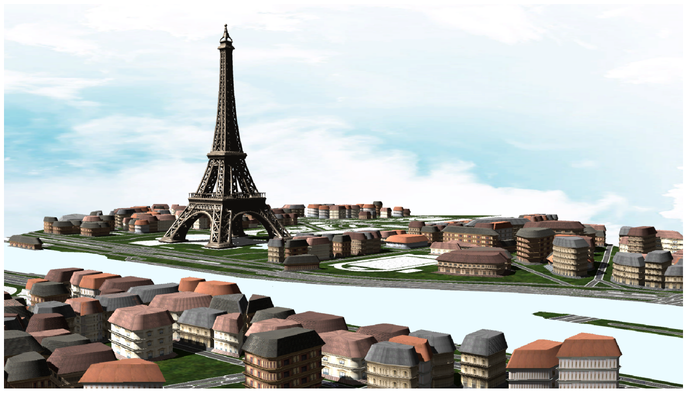
where final scale $k_\text{f} = 0.7$ and pillar geometry $\zeta_\text{p}^0(\mathbf{P}_i)$ is:

$$\zeta_\text{p}^0(\mathbf{P}_i) = \bigvee_{k=1}^{|\mathbf{P}_i|} \bigvee_{n=1}^{N_k} \Theta_{R_\text{p} h_\text{d}}\left(\mathbf{q}_k^n + w_\text{e}\sigma_\text{p}(\mathbf{c}_i - \mathbf{q}_k^n) + \mathbf{y}_\text{d}^0\right) \uparrow h_\text{d}$$

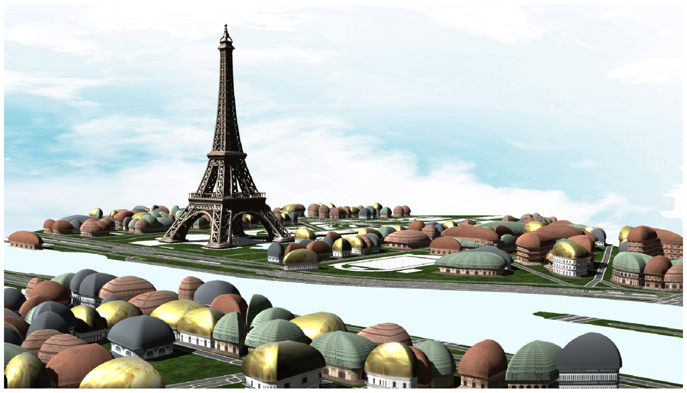
where sample count $N_k = \lfloor \frac{|\mathbf{p}_{k+1} - \mathbf{p}_k|}{2R_\text{p}} \rfloor$, the n^th sample point $\mathbf{q}_k^n = \mathbf{p}_k + n\left(\mathbf{p}_{k+1} - \mathbf{p}_k\right)/N_k$, $\mathbf{p}_k = \mathbf{P}_{i,k}$, circle generator



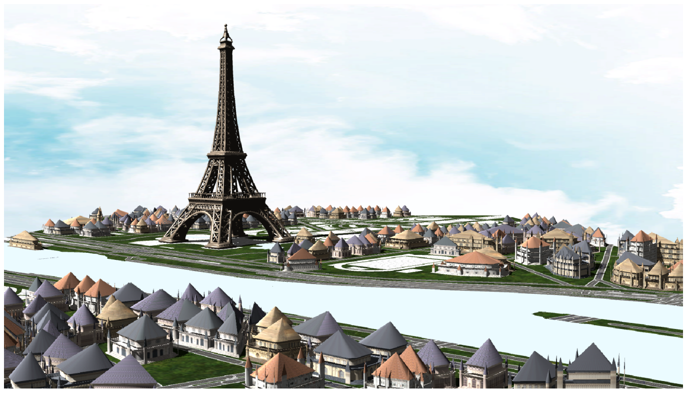
(a) Synthesized Virtual Modern Urban Scene



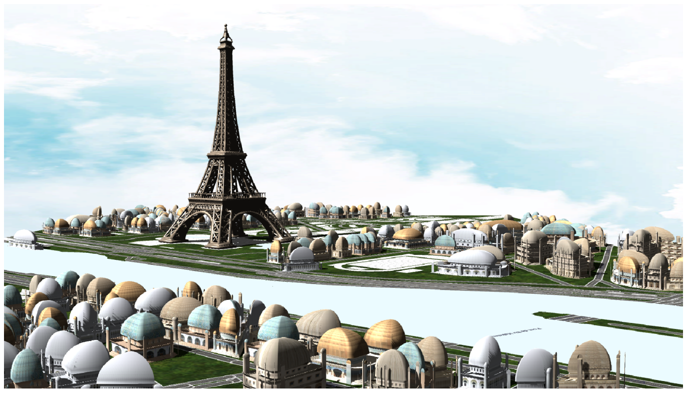
(b) Synthesized Virtual Baroque Palace Scene



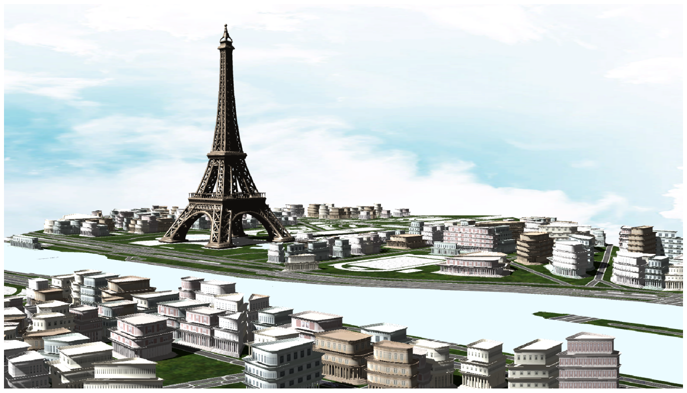
(c) Synthesized Virtual Byzantine Palace Scene



(d) Synthesized Virtual Gothic Palace Scene



(e) Synthesized Virtual Islamic Palace Scene



(f) Synthesized Virtual Neoclassic Palace Scene

Fig. 5. Experiment Result (Part 1). This figure shows the experiment results of synthesizing world palace scenes at the same location with different styles. The location setting is Eiffel Tower, Paris, France. Subfigures are synthesized scenes for modern urban (a), Baroque palaces (b), Byzantine palaces (c), Gothic palaces (d), Islamic palaces (e), and Neoclassic palaces (f).

$\Theta_R(\mathbf{p})$ represents a circle that has a center at $\mathbf{p}$ with radius of R, pillar offset $\sigma_{\mathrm{p}} = -0.8$, relative pillar radius $R_{\mathrm{p}} = 0.2$. Parameter settings for Neoclassical palace are: $w_{\mathrm{s}} = 1.5$, $h_{\mathrm{s}} = 3$, $h_{\mathrm{d}} = 15$, $w_{\mathrm{e}} = 1$, $h_{\mathrm{e}} = 3$, $w_{\mathrm{r}} = 2.5$, $h_{\mathrm{w}} = 8$. For Neoclassical palaces, pillars are dominant architectural elements inspired by classical Greek and Roman architecture. In our proposed approach, by combining the geometry of pillars with other parts of the palace geometry, we can generate visually appealing and immersive palaces that showcase the elegance and grandeur of Neoclassical design.

IV. Experiment Results

We implemented our proposed approach using Unity 3D with the 2019 version and generate these experiment results with the hardware configurations containing Intel Core i5 CPU, 32GB DDR4 RAM, and NVIDIA GeForce GTX 1650 4GB GDDR6 Graphics Card. Figure 5 figure shows the details of synthesized virtual world palace scenes at the same location, specifically the Eiffel Tower in Paris, France. The figure consists of six subfigures, each representing a synthesized scene with a different architectural style.

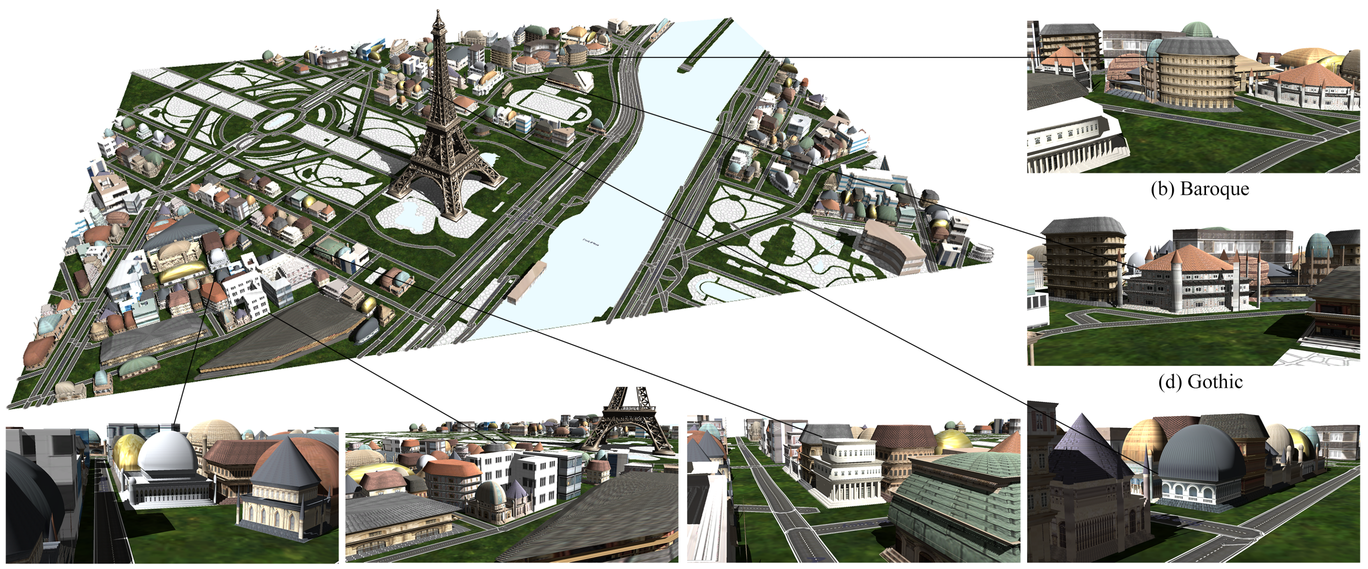


Fig. 6. Experiment Results (Part 2). This figure shows the details of synthesized virtual world palace scenes. Six different views are rendered from six cameras placed in the virtual world palace scene that is synthesized with our approach.

Figure 5 (a) showcases a synthesized modern urban scene where all buildings are extruded from a polygon. (b) presents a synthesized scene where there are Baroque features such as relatively flat roofs. (c) presents a synthesized scene with a fusion of the Byzantine styles such as the spherical roofs. (d) illustrates a synthesized scene featuring iconic Gothic architectural styles such as towering spires and sharp roofs. (e) showcases a synthesized scene with Islamic architectural elements such as spherical roofs, domes, and minarets. (f) depicts a synthesized scene with Neoclassical elements of pillars that are inspired by Greek and Roman architecture.

Figure 6 displays the details of synthesized virtual world palace scenes that incorporate a mixture of different architectural styles in the same location. Six subfigures show six different views to focus on palaces with different architectural styles. Palaces are synthesized with a random generator that has different probabilities to generate different styles among which Modern, Baroque, Byzantine, Neoclassic are 15% respectively, Gothic and Islamic are 20% respectively.

V. Conclusion

In this paper, we propose a novel technical approach to synthesize the virtual world palace scenes on OpenStreetMap. The experiment results present a unique visual representation of the palace scenes synthesized using different architectural styles across the world at the same location and demonstrates a visual effect of incorporating the diverse range of architectural styles into the same synthesized palace scene. Those experiment results validate the correctness and efficacy of our approach. Future work can synthesize palace scenes with more unique features to incorporate cultural elements from minority so as to enhance the inclusivity and diversity of virtual palace scenes by integrating underrepresented cultural elements.

References

[1] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive computing*, vol. 7, no. 4, pp. 12–18, 2008.

[2] D. Luxen and C. Vetter, "Real-time routing with openstreetmap data," in *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2011, pp. 513–516.

[3] J. J. Arsanjani, C. Barron, M. Bakillah, and M. Helbich, "Assessing the quality of openstreetmap contributors together with their contributions," in *Proceedings of the AGILE*, 2013, pp. 14–17.

[4] P. Neis and D. Zielstra, "Recent developments and future trends in volunteered geographic information research: The case of openstreetmap," *Future internet*, vol. 6, no. 1, pp. 76–106, 2014.

[5] P. Mooney, M. Minghini *et al.*, "A review of openstreetmap data," *Mapping and the citizen sensor*, pp. 37–59, 2017.

[6] A. Pluta and O. Lünsdorf, "esy-osmfilter–a python library to efficiently extract openstreetmap data," *Journal of Open Research Software*, vol. 8, no. 1, 2020.

[7] W. Li, "Make uber faster: Automatic optimization of uber schedule using openstreetmap data," in *Proceedings of the 2021 EURASIAGRAPHICS*, 2021, pp. 19–26.

[8] ——, "Simulating virtual construction scenes on openstreetmap," in *Proceedings of the 6th International Conference on Virtual and Augmented Reality Simulations*, 2022, pp. 14–20.

[9] ——, "Pm4vr: A scriptable parametric modeling interface for conceptual architecture design in vr," in *The 18th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*, 2022, pp. 1–8.

[10] ——, "Terrain synthesis for treadmill exergaming in virtual reality," in *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, 2023, pp. 263–269.

[11] W. Li, C. Li, M. Kim, H. Huang, and L.-F. Yu, "Location-aware adaptation of augmented reality narratives," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–12.

[12] W. Li, "Synthesizing virtual night scene on openstreetmap," in *2023 International Conference on Communications, Computing and Artificial Intelligence (CCCAI))*, 2023, pp. 1–6.

[13] ——, "Synthesizing virtual chinese palace scene on openstreetmap," in *2023 2nd International Conference on Image Processing and Media Computing (ICIPMC)*, 2023, pp. 1–6.

[14] A. Grant, "Go map - unity 3d asset, 3d map for ar gaming," https://gomap-asset.com/, 2017.