Dragon's Path: Synthesizing User-Centered Flying Creature Animation Paths for Outdoor Augmented Reality Experiences

Minyoung Kim
George Mason University
Fairfax, VA, USA
mkim229@gmu.edu

Rawan Alghofaili
University of Texas at Dallas
Richardson, TX, USA
rawan@utdallas.edu

Changyang Li
George Mason University
Fairfax, VA, USA
cli25@gmu.edu

Lap-Fai Yu
George Mason University
Fairfax, VA, USA
craigyu@gmu.edu

ABSTRACT

Advances in augmented reality promise to deliver highly immersive storytelling experiences by animating virtual characters naturally in the real world. However, creating such realistic animated content for viewing in augmented reality is non-trivial and challenging. In this paper, we present a novel approach to automatically generate user-centered flying creature animation paths for outdoor augmented reality experiences. Given a sequence of storyline actions, our approach finds suitable locations for the character to perform its actions via a location compatibility predictor trained with user preferences, synthesizing a corresponding animation path optimized with respect to the user's perspective. We applied our approach to synthesize user-centered augmented reality experiences based on different storyline actions and environments. We also conducted user study experiments to validate the efficacy of our approach for synthesizing desirable augmented reality experiences.

CCS CONCEPTS

• **Computing methodologies** → **Graphics systems and interfaces**.

KEYWORDS

character animation, mixed reality, scene understanding

ACM Reference Format:
Minyoung Kim, Rawan Alghofaili, Changyang Li, and Lap-Fai Yu. 2024. Dragon's Path: Synthesizing User-Centered Flying Creature Animation Paths for Outdoor Augmented Reality Experiences. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657397

1 INTRODUCTION

From novels, dramas, movies, to video games, people have always been fascinated by the idea of visualizing how imaginary characters such as dragons, unicorns, and phoenixes navigate and interact with the real world. The recent emergence of augmented reality (AR) devices presents us with exciting opportunities to leverage AR technologies for immersive visualization of such digital storytelling, enabling users to engage in AR experiences from animated performances by imaginary virtual characters to virtual training scenarios [Huang et al. 2021; Liu et al. 2022] in situ. However, unlike traditional storytelling, AR-based storytelling needs to adapt to environment's contexts and the user's viewpoint to provide a compatible, immersive, and pleasurable experience.

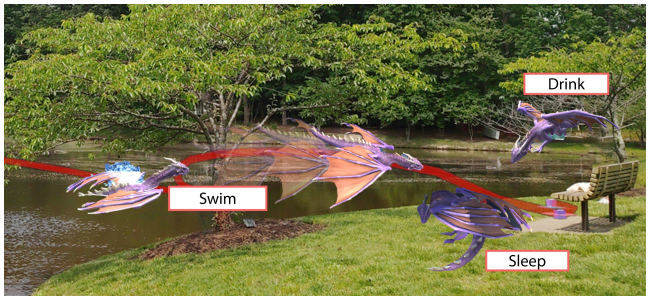


Figure 1: Given storyline actions (e.g. *Swim*, *Sleep*, *Drink*), our approach automatically selects a compatible location for each action and generates a user-centered path (red) by considering factors such as the visibility and distance with respect to the user. The virtual flying creature, a dragon in this example, swims in the water, sleeps in between the trees, and then drinks on the lakeshore. The path generated by our approach is unoccluded and fully visible to the user.

Current augmented reality experiences attempt to add richness to the medium by employing virtual characters that interact with the viewer's real-world environment. Researchers attempted to enhance these experiences by leveraging scene information such as scene semantics [Li et al. 2022; Liang et al. 2021] or scene geometry [Alghofaili et al. 2023]. However, these approaches did not contextualize the user's location and cater the AR experience to the user's unique viewpoint. Furthermore, these approaches [Alghofaili et al. 2023; Li et al. 2022; Liang et al. 2021] operate in indoor scenes. The development of more lightweight wireless AR devices enables the delivery of AR animation experiences to users outdoors. Our approach tackles the challenges of presenting large-scale outdoor virtual character animations driven by the scene contexts and the user's view in augmented reality.

In this paper, we propose a new approach for user-centered content generation focusing on flying creature animation paths for outdoor AR experiences as shown in Figure 1. The generated paths adapt to the user and the environment to deliver immersive AR experiences. Our primary goal in this work is to find an optimal

SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07.
https://doi.org/10.1145/3641519.3657397

character animation path with respect to the user's current status and surroundings based on a input sequence of the character's storyline actions. In other words, when it generates paths for AR animations in the real world, our approach considers the user's position and viewpoint so as to keep the user visually engaged with the AR content. To achieve this goal, our approach tackles several challenges: (1) understanding the input scene's context; (2) determining the waypoints where the character should perform the input storyline actions; and (3) finding the optimal path while considering the storyline action waypoints and the user's perspective.

Our secondary goal is to synthesize adaptive character animations in AR by analyzing scene semantic information. For example, consider animating a dragon in AR. An environment possesses various inherent semantic information and relevant interactions for a dragon, e.g., watery areas for drinking, shady areas for sleeping, and so on. During the preprocessing step, we utilize a learning-based predictor to determine the suitable scene regions for performing different character actions. Based on this correlation between the environment and the character's actions, we explore where certain behavioral animations should occur by optimizing objective functions considering several factors such as visibility and compatibility of the corresponding generated path. The user can then watch the character animation in AR, where the virtual character (e.g., a dragon) follows the generated path and performs storyline actions at chosen waypoints. Overall, our contributions are the following:

- We devise a learning-based method for determining the locations for a virtual character to perform its storyline actions in outdoor environments;
- We propose an automatic approach for synthesizing user-centered flying creature animation paths in 3D scenes for outdoor augmented reality experiences;
- We evaluate the effectiveness of our approach via experiments conducted under various conditions and user studies performed in the real world.

2 RELATED WORK

2.1 Scene Understanding for Augmented Reality

One of the pioneering works of outdoor AR is the Touring Machine [Feiner et al. 1997], a system that integrated virtual 3D graphic information into 3D real-world scenes. Researchers also proposed an approach for providing AR visualization on a city scale [Lee et al. 2012] and devised a virtual component registration mechanism by using 3D Geographic Information System [Huang et al. 2016]. Regarding indoor AR, researchers focused on the analysis of indoor environments by considering 3D object layout generation for AR [Gal et al. 2014], exploiting spatial relationships and interaction between a user and a virtual character [Lang et al. 2019], and expanding scene analysis to 3D scene graphs that encode the relations of objects [Tahara et al. 2020].

Several works explored leveraging scene geometry for indoor augmented reality. Some of these works [Ipsita et al. 2021; Sankar and Seitz 2017; Wang et al. 2021] facilitated capturing scene geometry for augmented reality content-authoring. In terms of design process, SceneCtrl [Yue et al. 2017] enables the re-design of these captured indoor scenes and Warpy [Alghofaili et al. 2023] integrates scene geometry into the creation of in-situ 3D curves in AR. In contrast to indoor AR research, existing outdoor AR works mainly account for displaying information based on geographical data. They generally do not consider context-based scene understanding as much as indoor AR works do. Our work analyzes outdoor scenes to model the correlations between environmental features and affordances pertaining to a virtual character's actions in AR.

2.2 Animation Path Generation

Pathfinding is a classic problem in computer graphics and robotics research. Popular pathfinding algorithms include A* algorithms [Hart et al. 1968] based on grid search, RRT [Karaman and Frazzoli 2011] based on sampling, etc. Abd Algfoor et al. [2015] and Sánchez-Ibáñez et al. [2021] conducted a comprehensive survey about these algorithms. Though there are numerous pathfinding methods, most existing methods are not user-oriented, meaning that they ignore visibility from the user and semantic information in the user's surroundings. As for finding a path for a flying character, prior research works focus on efficient approaches to generate an aerial path, particularly, for drones that perform large-scale urban reconstruction [Yang et al. 2018; Zhang et al. 2021], for quadrotor camera path of aerial video [Gebhardt and Hilliges 2021; Joubert et al. 2015; Xie et al. 2018], and for virtual winged characters [Won et al. 2017, 2018]. Those works explored paths to pursue specific objectives via optimization or reinforcement learning methods.

In AR environment, previous works suggested different approaches to control the motions and paths of virtual characters using hardware devices [Anderegg et al. 2018; Garcia et al. 2019; Ye et al. 2020]. However, these works utilized a mobile device trajectory created by user manipulation, they do not incorporate the information of the scene or surrounding environment. As for animal agents, a scene-aware behavior generator [Liang et al. 2021] was proposed to enable a virtual pet to interact with a virtual indoor scene. In contrast to prior research which operated in a small indoor scene, our work focuses on generating a user-centered flying creature animation path for a large-scale outdoor scene.

2.3 Augmented Reality Storytelling Aid

As AR functionalities advance, many immersive AR storytelling experiences have been explored. As for pioneering works, a tangible AR interface for storytelling [Zhou et al. 2004] and methods for extending real books to AR were proposed [Billinghurst et al. 2001; Grasset et al. 2008]. As for story authoring, Rumiński and Walczak [2013] proposed an authoring tool for creating interactive AR content on mobile devices. Such approaches have been extended to deal with interactive situations, for example, Li et al. [2022] proposed an interactive AR storytelling approach to enable a user to participate in an indoor AR story. However, the above works did not consider the varying viewpoints of users. Like Dreamwalker [Yang et al. 2019], which embeds a pre-authored VR environment into the real world, our work enables users to experience AR storytelling in an outdoor scene by watching virtual character animations synthesized for the scene considering the user's perspective. As for outdoor AR experience, Li et al. [2023] focused on conveying AR storytelling with a pleasurable walking route based on a 2D input map and employing compatibility values from pre-defined zone

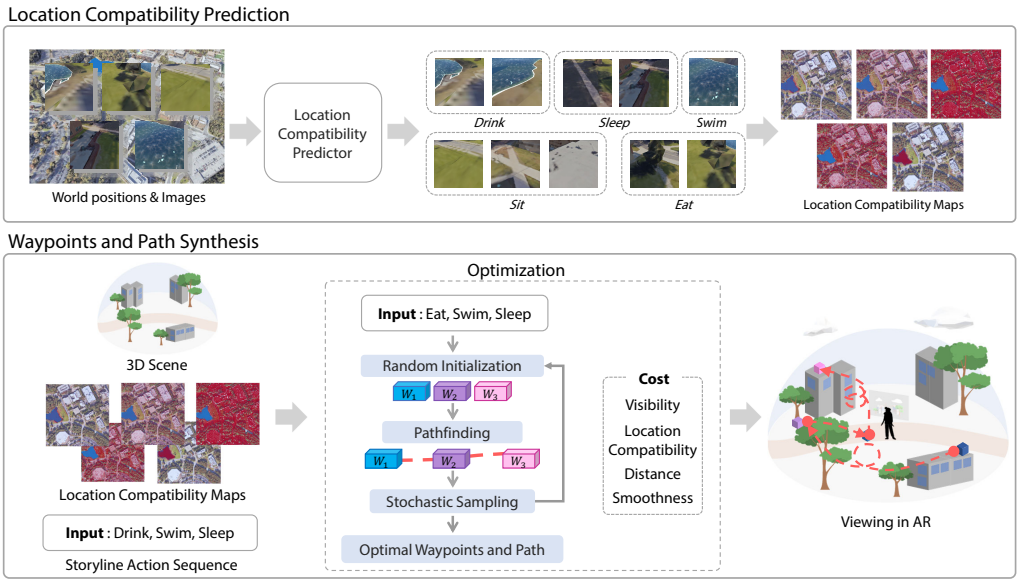


Figure 2: An overview of our approach. The location compatibility prediction model computes the location compatibility values for performing different character storyline actions on the input 3D scene. During target and path synthesis, our approach generates optimal waypoints and a corresponding path on the input 3D scene by using the prediction result from the previous step and a storyline action sequence specified by the designer.

maps. Compared to their work, our work generates user-centered AR animation paths in 3D space represented by voxels, which considers user-centered factors like the visibility of the animation path with respect to the user amid 3D structures in the real world. Moreover, our prediction model infers compatibilities in new 3D scenes by a few-shot learning approach rather than using pre-defined zone maps. Our approach may complement existing story generation works [Chen et al. 2021; Zhang et al. 2019] by adapting generated stories to different environments.

3 OVERVIEW

Figure 2 shows an overview of our approach. Our goal is to generate an optimal character animation path for a given storyline to be visualized in a real-world outdoor environment in augmented reality. It takes a sequence of the virtual character's storyline actions and the real-world 3D surrounding of the user as input. Based on the 3D scene, our approach infers the compatibility of each storyline action by a learning-based method (Section 4). With the trained prediction model, our approach automatically samples waypoints where the virtual character will perform its storyline actions, and synthesizes a corresponding animation path via an optimization. In sampling the waypoints, the optimizer considers the locations' compatibility for performing the character's actions (Section 5). Based on the generated results, the user wearing an AR headset (e.g., a HoloLens 2) watches the virtual character animated via the synthesized animation path, performing the sequence of storyline actions at semantically compatible places within the user's surroundings in the real world.

3.1 Problem Representation

As shown in Figure 3, our approach represents the virtual 3D scene as a voxel set $\mathcal{V} = \{V_i\}$ with an adaptive octree structure. The voxels within the set refer to different levels of detail and vary in size depending on whether the enclosed 3D space contains scene geometry. Each voxel encodes its location compatibility with respect to storyline actions inferred by a learning-based method (Section 4), as well as visibility and distance information with respect to the user, who is assumed to be standing at a particular location and facing a particular direction in the scene.

Our approach is formulated as an optimization. In each iteration of the optimization, it samples waypoints $\mathcal{W} = \{w_i\}$ among the voxel centers, based on which a modified A* pathfinding method is applied to compute a connecting path. The location compatibility and visibility of the sampled waypoints, as well as the visibility, distance from the user, and smoothness of the generated path, are considered in the optimization. The optimization approach generates a set of optimized waypoints and a connecting path in the voxel space as the output. In a postprocessing step, we apply the De Casteljau algorithm to composite multiple Bezier curves, whose end points match the optimized waypoints, to obtain a smooth animation path. To compute Bezier curevs, a closed-form formula could be also applicable.

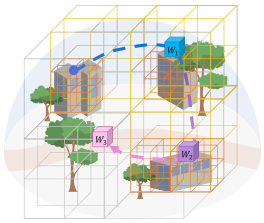


Figure 3: A grid of voxels with varying sizes representing the input 3D scene.

4 LOCATION COMPATIBILITY PREDICTION

We were inspired by the work of Savva et al. [2014] to predict the compatibility of locations within a 3D scene for a virtual character to perform specific actions. Savva et al. [2014] proposed a method to establish correlations between the geometry and the functionality of 3D indoor scenes, which could be adopted for inferring compatible storyline actions. Our location compatibility method infers the character's actions for large-scale outdoor 3D scenes instead. Using a learning-based method, the location compatibility method predicts the probabilities that a specific location is compatible with various classes of storyline actions $A = \{drink, eat, sit, sleep, swim\}$. We focused on five actions that are common across both real and imaginary creatures. More actions can be supported through additional annotated data by using our method, which predicts human-preferred locations for these actions. Figure 6 visualizes the location compatibility values with different actions for an example scene.

4.1 Prediction

Our location compatibility prediction model was based on a few-shot learning method employing a feature extractor and a fine-tuned model. We used a pre-trained feature extractor to extract image features. This method suits our dataset composed of a relatively small number of images with annotation.

Feature Extraction. In this stage, we used a pre-trained ResNet-18 [He et al. 2016] as a feature extractor to extract features from images in our training dataset, which we later used to compute a mean feature vector μ per action. Figure 4 (top) illustrates the feature extraction stage:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} f_\theta(I_i) \tag{1}$$

Given an image I_i, the extractor generates the feature vector $F_i = f_\theta(I_i)$ and computes a mean feature vector μ. We generated $|A|$ mean feature vectors for all storyline action classes by using our training set containing 4,036 locations images in total. Here, $|A|$ is the total number of all available storyline action classes.

Fine-Tuning. Before fine-tuning our storyline action predictor, we initialized its weights. Given the set of storyline action class A, and each class i having $\{\mu_i\}$ computed from Equation 1, we constructs initial weight $M = \{\mu_1, \mu_2, ..., \mu_{|A|}\}$ where $M \in R^{d\times|A|}$ by concatenating all μ_i. Here, d is the length of the feature vector. We trained our storyline action predictor with weight M by fine-tuning the extractor f_θ with the training data.

$$Z_a(I_i) = \sigma(M \cdot F_i) \tag{2}$$

In Equation 2, given an image I_i from the training dataset, we extracted the feature F_i. Then, the predictor measures the similarity between weight M and F_i of input image I_i. The sigmoid layer σ produces the compatibility value vector $Z_a(I_i)$ with respect to the storyline action class $a \in A$ for image I_i.

Location Compatibility Prediction. After the fine-tuning stage, our predictor can predict the compatibility values Z_a with respect to storyline action class $a \in A$ on new image data. For each voxel, we compute its location compatibility values with respect to the storyline actions. Note that the voxels in the grid have varying sizes. A large voxel may contain multiple sampled locations of the scene. Figure 5 shows an illustration, where the red asterisk denotes a voxel's center and the green asterisks denote the sampled locations contained by this voxel. Note that the center of a voxel are not included as sampling. The compatibility value with respect to storyline action a of a voxel v is computed by averaging the compatibility values of the sampled locations contained by the voxel:

$$Z_a(v) = \frac{1}{|\alpha|}\sum_{I\in\alpha} Z_a(I), \tag{3}$$

where α is the set of location images of all sampled locations contained by voxel v. Figure 6 shows the location compatibility maps of different storyline actions. Each map consists of a grid of voxels overlaid in the scene. Each voxel is colored according to its location compatibility value for the storyline action class calculated by (3). As observed, the *drink* action is compatible with locations around the lake shore; the *swim* action is mainly compatible with the lake; while the *eat* action shows moderate compatibility rather than a strong preference for particular locations on the map.

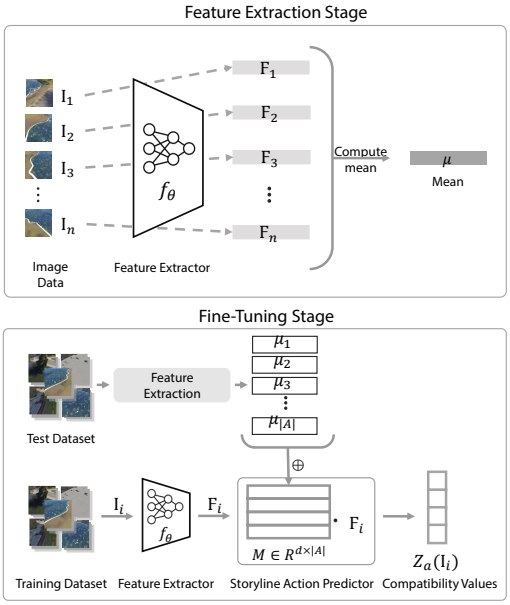


Figure 4: Two stages of training the location compatibility prediction model. In the environmental feature extraction stage, we compute the mean environmental feature by using the pre-trained feature extractor. The computed mean feature vectors are used to initialize the weight of our predictor. In the fine-tuning stage, both feature extractor and fully connected predictor are updated.

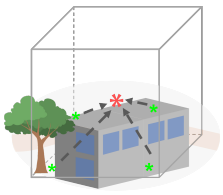
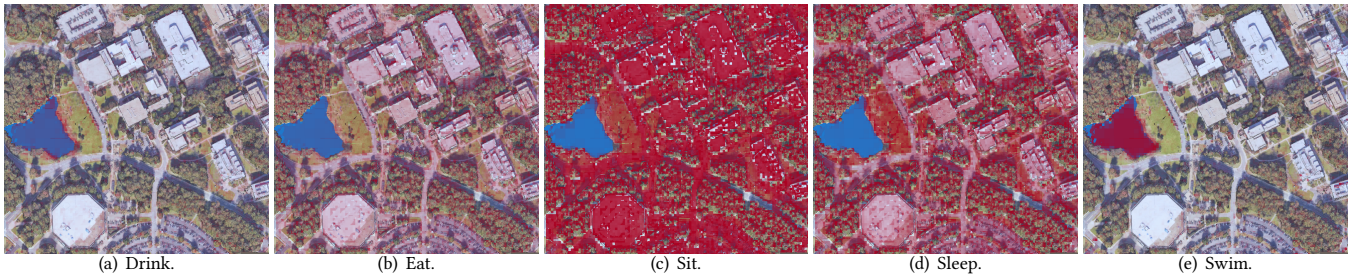


Figure 5: Sampling illustration. Since the voxel size varies, multiple sampling occurs in a voxel.



(a) Drink. (b) Eat. (c) Sit. (d) Sleep. (e) Swim.

Figure 6: Location compatibility maps. Each map depicts the location compatibility values with respect to a storyline action class. The red in the map denotes the location compatibility and its intensity increases with the location compatibility value.

Architecture. Our approach is inspired by few-shot learning methods used in previous works [Chen et al. 2019; Dhillon et al. 2019]. Unlike these previous works which trained their networks from scratch, we employed a pre-trained ResNet-18 [He et al. 2016] without its last fully-connected layer. In our predictor, we added a new fully-connected layer with a sigmoid layer to predict storyline actions. Please refer to related papers [Chen et al. 2019] for details.

5 WAYPOINT AND PATH SYNTHESIS

To achieve our goal of finding a user-centered flying creature animation path, our approach samples waypoints among voxel centers. The virtual character performs the storyline actions at the waypoints. Specifically, our objective is to sample a sequence of waypoints $\mathcal{W} = \{w_1, ..., w_n\}$ for performing the sequence of storyline actions $\mathcal{A} = \{a_1, ..., a_n\}$. During the optimization, the waypoints are updated, and our modified A^* algorithm computes the corresponding optimal path $\mathcal{P} = \{v_1, ..., v_n\}$. We use a Markov chain Monte Carlo sampler to search for the optimal solution. Note that path $\mathcal{P}$ is deterministic on the chosen waypoints because it is obtained by using our modified A* algorithm. In other words, waypoints $\mathcal{W}$ are the only decision variables for optimization.

5.1 Cost Function

To evaluate the proposed solution $T = \{(\mathcal{W}, \mathcal{P})\}$, we define a total cost function:

$$C_{\text{total}}(T) = \lambda_{\text{way}} C_{\text{way}}(\mathcal{W}) + \lambda_{\text{path}} C_{\text{path}}(\mathcal{P}), \tag{4}$$

where $\mathcal{W} = \{w_1 ... w_n\}$ is a sequence of proposed waypoints. The total cost function consists of two parts: waypoint cost $C_{\text{way}}(\mathcal{W})$ and path cost $C_{\text{path}}(\mathcal{P})$. While computing the costs, $C_{\text{path}}(\mathcal{P})$ changes with $C_{\text{way}}(\mathcal{W})$ because $\mathcal{P}$ depends on $\mathcal{W}$. In other words, $C_{\text{path}}(\mathcal{P})$ represents the cost of the current path determined by the current waypoints $\mathcal{W}$.

The waypoint cost $C_{\text{way}}(\mathcal{W})$ comprises two cost terms, location compatibility cost C_{l} and visibility cost C_{v} with weights λ_{l} and λ_{vw}:

$$C_{\text{way}}(\mathcal{W}) = \sum_{i=1}^{n} \big(\lambda_{\text{l}} C_{\text{l}}(v_i, a_i) + \lambda_{\text{vw}} C_{\text{v}}(v_i)\big). \tag{5}$$

Here, v_i is the voxel at which waypoint w_i is located in the current solution. The path cost $C_{\text{path}}(\mathcal{P})$ is calculated using the generated path $\mathcal{P} = \{v_k\}$ covering a series of voxels:

$$C_{\text{path}}(\mathcal{P}) = \sum_{v_k \in \mathcal{P}} \big(\lambda_{\text{vp}} C_{\text{v}}(v_k) + \lambda_{\text{d}} C_{\text{d}}(v_k) + \lambda_{\text{s}} C_{\text{s}}(v_k)\big). \tag{6}$$

$C_{\text{path}}(\mathcal{P})$ consists of three terms: visibility C_{v}, distance C_{d}, and smoothness C_{s} with corresponding weights λ_{vp}, λ_{d}, λ_{s}. Note that the visibility cost is employed twice with different weights, λ_{vw} and λ_{vp}, for evaluating the visibility of the waypoints in (5) and the visibility of the path in (6). We elaborate the constituent cost terms in the following sections.

5.1.1 Location Compatibility Cost. First, we define the location compatibility term to compute whether the waypoint w_i's voxel v_i is compatible with the given storyline action a_i:

$$C_{\text{l}}(v_i, a_i) = 1 - Z_{a_i}(v_i), \tag{7}$$

where $Z_{a_i}(v_i)$ is a compatibility value of storyline action a_i at voxel v_i that we obtain from our predictor (Section 4). This cost prompts storyline action a_i to occur at voxel v_i with a high compatibility.

5.1.2 Visibility Cost. One significant part of our system is checking the visibility of the surrounding environment with respect to the user. As we represent the surrounding environment by a voxel grid, we check whether these voxels are visible with respect to the current position and head orientation of the user. We apply two culling processes, frustum culling and occlusion culling, to determine the visibility cost of each voxel. First, frustum culling excludes the voxels that are outside the camera's view frustum. Specifically, the camera pose is set to the user's head pose. After frustum culling, the retained voxels go through an occlusion culling test. In this test, our approach checks whether the voxels are occluded by obstacles in the surrounding environment from the view of the user.

Given a voxel v_i, our approach calculates its visibility value σ_{v_i} by performing an occlusion test for all its eight corners and the center. If a corner is visible, 1 is added to the visibility value σ_{v_i}. If the center is visible, 8 is added to the visibility value σ_{v_i} to account for the larger significance of the voxel center in representing the voxel's visibility. Therefore, each voxel v_i is associated with a visibility value $\sigma_{v_i} \in [0, V_{\text{max}}]$, where $V_{\text{max}} = 16$. A voxel that is completely invisible has a visibility value of zero. We then define the visibility cost of each voxel v_i as:

$$C_{\text{v}}(v_i) = \frac{V_{\text{max}} - \sigma_{v_i}}{V_{\text{max}}}. \tag{8}$$

5.1.3 Distance Cost. To generate a user-centered result, we add a distance term to keep the generated path close to the user. The distance term acts as a regularization determining a closer path when multiple path candidates have the same visibility. In addition, the designer can choose whether to apply this distance term and set different target distances as needed. Before starting the optimization, our approach computes the distances between the centers of all voxels and the user's position. We assume that the area where the user can walk around is bounded in the real environment. Therefore, we set the maximum distance as $D_{\max}$ to ensure a proper distance between the virtual character and the user. By default, we set $D_{\max} = 250$ in our setting. The distance cost of a voxel v_k is defined as:

$$C_{\mathrm{d}}(v_k) = \frac{d_{v_k}}{D_{\max}}, \tag{9}$$

where d_{v_k} is the distance between the user's position and the center of voxel v_k, and it is normalized by $D_{\max}$.

5.1.4 Smoothness Cost. We define a smoothness cost to steer the A* algorithm to select the smoothest path possible:

$$\begin{aligned} \theta_{v_k} &= \mathbf{v}_{v_{k-1}v_k} \cdot \mathbf{v}_{v_k v_{k+1}}, \\ C_{\mathrm{s}}(v_k) &= \frac{1}{2}\big(1 - \cos(\theta_{v_k})\big). \end{aligned} \tag{10}$$

Figure 7 depicts the smoothness cost. Unlike other cost functions, the smoothness cost is obtained during the runtime of the algorithm by calculating the angle change θ_{v_k} based on the previously selected voxel v_{k-1}, the current voxel v_k, and the next candidate voxel v_{k+1} along the path that the character traverses. Without this term, the algorithm may generate paths with sharp corners.

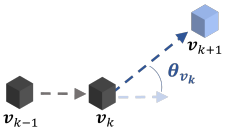


Figure 7: Smoothness cost. The term penalize the angle θ_{v_k} between the current v_k and the next candidate v_{k+1}.

5.2 Modified A* Algorithm

We apply a modified A* algorithm to find the optimal path for the virtual character in AR considering the visibility, distance from the user, and path smoothness. The A* algorithm [Hart et al. 1968] is broadly applied for pathfinding. Due to its simplicity and robustness, we utilize the A* algorithm for generating animation paths considering three terms: visibility cost C_{v}, distance cost C_{d}, and smoothness cost C_{s}. These terms estimate the path cost from the start voxel to the destination voxel by accumulating the cost associated with each voxel of the generated path. λ_{vp}, λ_{d} and λ_{s} are the weights of each cost and they are set as 0.7, 0.2, and 0.1, by default.

Our modified A* algorithm searches for a path, starting from the waypoint where the virtual character is located and ending at the final waypoint where the last storyline action takes place. After running the A* algorithm, we obtain the sequence of voxels comprising the optimized path. The optimized result can produce either a 3D path or a 2.5D path along the ground or wall depends on which voxels are selected. For example, if the designer wants to generate the animation path for a humanoid character walking on the ground, they can choose that the algorithm ignores mid-air voxels and produce 2.5D path.

Our modified A* algorithm computes the term G, which consists of the current path cost $C_{\mathrm{path}}(\mathcal{P}_{\mathrm{g}})$ and an estimated heuristic path cost $C_{\mathrm{path}}(\mathcal{P}_{\mathrm{h}})$

$$G = \lambda_{\mathrm{g}} C_{\mathrm{path}}(\mathcal{P}_{\mathrm{g}}) + \lambda_{\mathrm{h}} C_{\mathrm{path}}(\mathcal{P}_{\mathrm{h}}), \tag{11}$$

where $\mathcal{P}_{\mathrm{g}} = \{v_s, ..., v_i\}$ from the start voxel v_s to the current voxel v_i and $\mathcal{P}_{\mathrm{h}} = \{v_i, ..., v_d\}$ from the current voxel v_i to the destination voxel v_d. By default, we set the weights $\lambda_{\mathrm{g}} = 0.8$ and $\lambda_{\mathrm{h}} = 0.2$. For the customized terms, we apply the three cost terms aforementioned. First, the visibility cost C_{v} helps pick the visible voxels to approach the destination. By reducing the value of the visibility cost, the algorithm selects a visible path through the user's surrounding environment. Second, we use the distance cost C_{d} to prompt the path to stay close to the user. Third, we apply the smoothness cost C_{s} to favor a smooth path rather than a zig-zag path with sharp turns. Figure 8 shows the effects on the waypoints and paths if a cost term is omitted from the optimization as an ablation test.

5.3 Procedure

In the optimization step, we apply simulated annealing [Kirkpatrick et al. 1983] with a Metropolis-Hastings state-search step [Hastings 1970; Metropolis et al. 1953] to find the optimal solution satisfying our optimization criteria. We define a Boltzmann-like objective function:

$$f(T) = e^{\frac{1}{\tau} C_{\mathrm{total}}(T)}, \tag{12}$$

where τ is the temperature parameter of simulated annealing. The temperature is decreased gradually over the optimization iterations. The lower the temperature is, the greedier the optimization is. The optimization is terminated when the total cost change converges to less than 3%. In practice, we set $\lambda_{\mathrm{way}} = 50$, $\lambda_{\mathrm{path}} = 1$, $\lambda_{\mathrm{l}} = 0.9$, $\lambda_{\mathrm{vw}} = 0.1$, $\lambda_{\mathrm{vp}} = 0.7$, $\lambda_{\mathrm{d}} = 0.2$, and $\lambda_{\mathrm{s}} = 0.1$. Note that the lambda values can be adjusted depends on characteristics of a scene, for example, the spaciousness of the experiment locations.

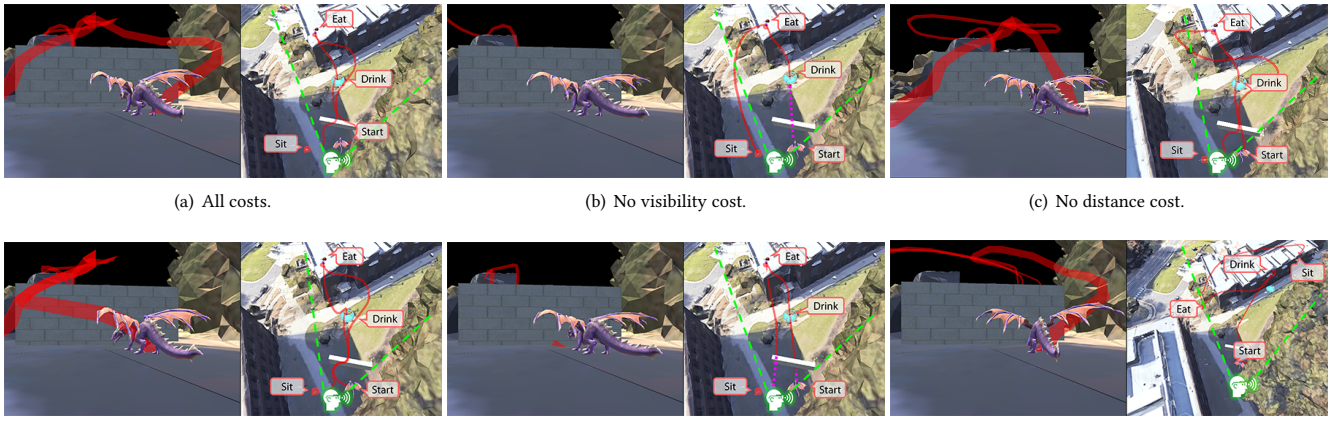
Proposed Moves. To explore the possible waypoint locations, our optimizer proposes a local move that translates a randomly-selected waypoint within a certain range slightly and randomly. This range also reduces with the temperature decrease until reaching 20% of the original range. Our optimizer also applies a global reconfiguration move occasionally that swaps two randomly-selected waypoints. The selection probabilities of the local move and global move are set as 90% and 10%. The proposed solution formed by applying a move is accepted with a probability set by the Metropolis criterion.

6 EXPERIMENTS

We performed a series of experiments to show the general applicability of our approach as well as its capabilities of handling different practical situations. For the implementation details of model training and deployment, please refer to our supplementary material.

6.1 Dragon Results

We use a virtual dragon as the character and a campus as the environment in this experiment. There are two stories, Story A (*Sleep-Sit-Swim*) and Story B (*Sit-Eat-Sleep*), as well as three locations (Locations 1, 2, & 3), used in this experiment. Since there is no area with water near Location 3, we add a virtual environmental object (i.e. a virtual pond) having high location compatibility values



(a) All costs. (b) No visibility cost. (c) No distance cost.

(d) No smoothness cost. (e) No cost (traditional A*). (f) No location compatibility cost.

Figure 8: Effects of omitting costs with a virtual wall. (a) A path generated with all costs. (b) The result without the visibility cost is less visible and more occluded by the wall. (c) The result without the distance cost shows a long path which is far from the user. (d) The result without the smoothness cost has a steep direction change after the start point. (e) The result of traditional A* refers to a shortest path with no user-centered consideration. (f) The result without the location compatibility cost shows actions performed at incompatible places. Green dashed lines denote the user's view range. Magenta dotted lines denote the occluded portions of the generated paths.

for *swim* and *drink*. Based on these, we created four story-location combinations of scenarios to show our approach's efficacy and adaptiveness. Our approach could be integrated into story-agnostic environments and constitute part of AR storytelling system.

Figure 9 shows the results. The top figure shows an overview of the environment and the four story-location combinations. The first two rows show the results of adapting Story A to two different locations (Locations 1 & 2) by our approach, demonstrating the its location adaptiveness. The last two rows show the results of delivering two different stories (Stories A & B) at the same location (Location 3), showing the capability of our approach for generating user-centered AR experiences at a location. In each row of the result, the overview figure shows the waypoints and path generated. The zoom-in views of the three storyline actions are also shown, which depict how the dragon animates and performs the actions at the chosen waypoints from the user's view. The top-right sub-figure shows the compatibility with the corresponding action, with red voxels indicating a high compatibility. In general, the waypoints are reasonably chosen according to their associated storyline actions and they are compatible with the contexts of the environment. According to the location compatibility maps (Figure 6) and the top-right sub-figures in each row of result, all actions (*sleep*, *sit*, *eat*, and *swim*) are placed at locations with high compatibility values of the commensurate actions. For example, the dragon swims in the lake or the virtual fountain. Furthermore, the paths found by our approach are fully visible from the user's locations. The results show that our approach placed the waypoints and the path reasonably. As another example of showing spatial adaptivity across different environments, we conducted experiments in Sydney's urban area and Hawaii's nature park. Please refer to our supplementary material.

6.2 Changing Viewpoint

Depending on the user's position and orientation, our approach can recompute the path regarding the user's updated viewpoint. It updates the visible voxels regarding the user by performing the visibility test again. Figure 10 shows two examples on the lake side and the campus. As the user walks from right to left in both examples, the path are updated so that the dragon animation stays visible to the user. Our approach takes less than a second (50 ms in average) to recompute a path, which can achieve a real-time experience. The frequency of updates depends on the user's distance traveled. In this example, we updated the path every time the user moved one meter from their previous location, while determining the visibility based on the orientation where the user was looking at the moment.

6.3 Waypoint Constraints

Our approach can be extended with waypoint constraints to meet additional design needs (Figure 12). A designer can fix certain waypoints on the 3D scene geometry to control where certain storyline actions take place. The dragon is assumed to fly straight, avoiding obstacles between a pair of fixed waypoints by using traditional A*. Our approach finds the optimized path of the remaining waypoints in adherence to the fixed waypoint constraints. This can generalize our approach to support other animation and locomotion e.g. crawling, walking, and wall climbing. With constraints, our approach can be applied differently, not limited to a flying creature. Figure 12 shows two example of setting a pair of waypoints on the windows and the wall to guide the dragon to fly adequately.

6.4 Special Viewpoint

Our approach considers the 3D layout of the real-world scene when calculating waypoints and the path from the user's perspective. The user is not restricted to standing on the ground when viewing the augmented reality experience because our approach considers 3D space visibility. Figure 11 shows an example where the user is on a fourth floor in a building looking outside the window. Our approach samples appropriate waypoints on the ground and computes a clear path visible to the user to show the dragon in augmented reality. However, it may face restrictions within heavily constrained environments, such as crowded obstacles and tall buildings.

7 DISCUSSION

We conducted a real-world user study to evaluate our approach in terms of human-centered factors (e.g., visibility, engagement, effortlessness). We collected 25 participants' feedback on four paths e.g., Random-Traditional A* (Baseline), Designer-Traditional A*, ML-Traditional A* (Ours), and ML-Modified A* (Ours) at three different locations. Participants' ratings on the results showed that our approach selects the most pleasurable and environmentally compatible waypoints, and generates a user-centered path from the participants' standpoint. Please refer to our supplementary material for the details of our user study design and statistical results.

Participants commented that the right proximity of the animation path is important in AR experiences. Some said that they were entertained when the character was close, while others felt uncomfortable and confused. We believe that the right proximity depends on the user's preference and the type of the animation content to be shown. Findings from psychology and crowd simulation research pertaining to social distances [Hall and Hall 1966; Narain et al. 2009] might also be applied to place virtual characters within a comfortable distance of the user in augmented reality. Additionally, we observed that a location's layout may affect the users' proximity perception. For the details of the location-specific analysis, please refer to our supplementary material. Some participants said that they felt fatigued if they had to look up for a while. As we animated a flying creature, we expected that most of our animation results took place in mid-air. Future research may consider the head and neck tiredness associated with the AR animation path in line with recent research on optimizing AR ergonomics by modeling and predicting a user's neck muscle contraction [Zhang et al. 2023].

8 LIMITATIONS AND FUTURE WORK

We discuss the limitations of our approach, which are interesting avenues for future extension. First, before applying our approach, the virtual and real-world objects (e.g., buildings) need to be aligned. Since our approach runs on the virtual world representation of the real-world environment where the user watches the animations in AR, the virtual world representation needs to be registered with its real-world counterpart for playing the animation via the AR device. A method for fast and accurate registration between the real world and its digital twin will facilitate the deployment of our approach in practice. Second, while our approach considers the 3D scene with buildings and large landscape entities such as lakes and buildings, it does not consider small objects such as benches and flowers, due to the limited resolution and the lack of details of the 3D scene representation that our approach runs on. Moreover, dynamic objects, such as pedestrians and cars, may be present in the scene, which our approach does not consider. Future work may apply computer vision techniques to recognize and reconstruct real-world objects in front of the user in real time. Third, our approach assumes one user, while a large-scale AR experience, such as an AR game, may involve multiple users at different locations and with different viewpoints seeing the same virtual character. For further discussion in practical limitations, please refer to our supplementary material.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their constructive comments. This project was supported by NSF grants (award numbers: 1942531 and 2128867).

REFERENCES

Zeyad Abd Algfoor, Mohd Shahrizal Sunar, and Hoshang Kolivand. 2015. A comprehensive study on pathfinding techniques for robotics and video games. *International Journal of Computer Games Technology* 2015 (2015).

Rawan Alghofaili, Cuong Nguyen, Vojtěch Krs, Nathan Carr, Radomír Měch, and Lap-Fai Yu. 2023. WARPY: Sketching Environment-Aware 3D Curves in Mobile Augmented Reality. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. https://doi.org/10.1109/VR55154.2023.00052

Raphael Anderegg, Loïc Ciccone, and Robert W Sumner. 2018. PuppetPhone: puppeteering virtual characters using a smartphone. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*. 1–6.

Mark Billinghurst, Hirokazu Kato, and Ivan Poupyrev. 2001. The magicbook-moving seamlessly between reality and virtuality. *IEEE Computer Graphics and Applications* 21, 3 (2001), 6–8.

Mengyu Chen, Andrés Monroy-Hernández, and Misha Sra. 2021. SceneAR: Scene-based Micro Narratives for Sharing and Remixing in Augmented Reality. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 294–303.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. 2019. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232* (2019).

Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. 2019. A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729* (2019).

Steven Feiner, Blair MacIntyre, Tobias Höllerer, and Anthony Webster. 1997. A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. *Personal Technologies* 1, 4 (1997), 208–217.

Ran Gal, Lior Shapira, Eyal Ofek, and Pushmeet Kohli. 2014. FLARE: Fast layout for augmented reality applications. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 207–212.

Maxime Garcia, Rémi Ronfard, and Marie-Paule Cani. 2019. Spatial Motion Doodles: Sketching Animation in VR Using Hand Gestures and Laban Motion Analysis. In *Motion, Interaction and Games*. 1–10.

Christoph Gebhardt and Otmar Hilliges. 2021. Optimization-based user support for cinematographic quadrotor camera target framing. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–13.

Raphaël Grasset, Andreas Dünser, and Mark Billinghurst. 2008. Edutainment with a mixed reality book: a visually augmented illustrative childrens' book. In *Proceedings of the international conference on advances in computer entertainment technology*.

Edmund T Hall and Edward T Hall. 1966. *The hidden dimension*. Vol. 609. Anchor.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107.

W Keith Hastings. 1970. Monte Carlo sampling methods using Markov chains and their applications. (1970).

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

Gaoping Huang, Xun Qian, Tianyi Wang, Fagun Patel, Maitreya Sreeram, Yuanzhi Cao, Karthik Ramani, and Alexander J Quinn. 2021. Adaptutar: An adaptive tutoring system for machine tasks in augmented reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

Wei Huang, Min Sun, and Songnian Li. 2016. A 3D GIS-based interactive registration mechanism for outdoor augmented reality system. *Expert Systems with Applications* 55 (2016), 48–58.

Ananya Ipsita, Hao Li, Runlin Duan, Yuanzhi Cao, Subramanian Chidambaram, Min Liu, and Karthik Ramani. 2021. VRFromX: From Scanned Reality to Interactive Virtual Experience with Human-in-the-Loop. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI EA '21)*. Association for Computing Machinery, New York, NY, USA, Article 289, 7 pages. https://doi.org/10.1145/3411763.3451747

Niels Joubert, Mike Roberts, Anh Truong, Floraine Berthouzoz, and Pat Hanrahan. 2015. An interactive tool for designing quadrotor camera shots. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–11.

Sertac Karaman and Emilio Frazzoli. 2011. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research* 30, 7 (2011), 846–894.

Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.

Yining Lang, Wei Liang, and Lap-Fai Yu. 2019. Virtual agent positioning driven by scene semantics in mixed reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 767–775.

Gun A Lee, Andreas Dünser, Seungwon Kim, and Mark Billinghurst. 2012. CityViewAR: A mobile outdoor AR application for city visualization. In *2012 IEEE international symposium on mixed and augmented reality-arts, media, and humanities (ISMAR-AMH)*. IEEE, 57–64.

Changyang Li, Wanwan Li, Haikun Huang, and Lap-Fai Yu. 2022. Interactive augmented reality storytelling guided by scene semantics. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–15.

Wanwan Li, Changyang Li, Minyoung Kim, Haikun Huang, and Lap-Fai Yu. 2023. Location-Aware Adaptation of Augmented Reality Narratives. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*.

Wei Liang, Xinzhe Yu, Rawan Alghofaili, Yining Lang, and Lap-Fai Yu. 2021. Scene-aware behavior synthesis for virtual pets in mixed reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

Huimin Liu, Minsoo Choi, Liuchuan Yu, Alexandros Koilias, Lap-Fai Yu, and Christos Mousas. 2022. Synthesizing Shared Space Virtual Reality Fire Evacuation Training Drills. In *2022 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics* 21, 6 (1953), 1087–1092.

Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C Lin. 2009. Aggregate dynamics for dense crowd simulation. In *ACM SIGGRAPH Asia 2009 papers*. 1–8.

Dariusz Rumiński and Krzysztof Walczak. 2013. Creation of interactive ar content on mobile devices. In *International Conference on Business Information Systems*. Springer, 258–269.

José Ricardo Sánchez-Ibáñez, Carlos J Pérez-del Pulgar, and Alfonso García-Cerezo. 2021. Path Planning for Autonomous Mobile Robots: A Review. *Sensors* 21, 23 (2021), 7898.

Aditya Sankar and Steven M. Seitz. 2017. Interactive Room Capture on 3D-Aware Mobile Devices. *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (2017).

Manolis Savva, Angel X Chang, Pat Hanrahan, Matthew Fisher, and Matthias Nießner. 2014. SceneGrok: Inferring action maps in 3D environments. *ACM transactions on graphics (TOG)* 33, 6 (2014), 1–10.

Tomu Tahara, Takashi Seno, Gaku Narita, and Tomoya Ishikawa. 2020. Retargetable AR: Context-aware augmented reality in indoor scenes based on 3D scene graph. In *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE, 249–255.

Zeyu Wang, Cuong Nguyen, Paul Asente, and Julie Dorsey. 2021. DistanciAR: Authoring Site-Specific Augmented Reality Experiences for Remote Environments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 411, 12 pages. https://doi.org/10.1145/3411764.3445552

Jungdam Won, Jongho Park, Kwanyu Kim, and Jehee Lee. 2017. How to train your dragon: example-guided control of flapping flight. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–13.

Jungdam Won, Jungnam Park, and Jehee Lee. 2018. Aerobatics control of flying creatures via self-regulated learning. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–10.

Ke Xie, Hao Yang, Shengqiu Huang, Dani Lischinski, Marc Christie, Kai Xu, Minglun Gong, Daniel Cohen-Or, and Hui Huang. 2018. Creating and chaining camera moves for quadrotor videography. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.

Hao Yang, Ke Xie, Shengqiu Huang, and Hui Huang. 2018. Uncut Aerial Video via a Single Sketch. *Computer Graphics Forum* 37, 7 (2018), 191–199.

Jackie Yang, Christian Holz, Eyal Ofek, and Andrew D Wilson. 2019. Dreamwalker: Substituting real-world walking experiences with a virtual reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*.

Hui Ye, Kin Chung Kwan, Wanchao Su, and Hongbo Fu. 2020. ARAnimator: in-situ character animation in mobile AR with user-defined motion gestures. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 83–1.

Ya-Ting Yue, Yong-Liang Yang, Gang Ren, and Wenping Wang. 2017. SceneCtrl: Mixed Reality Enhancement via Efficient Scene Editing. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 427–436. https://doi.org/10.1145/3126594.3126601

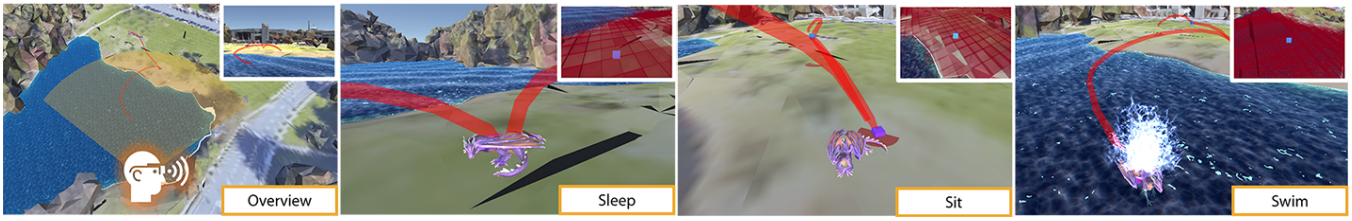
Han Zhang, Yucong Yao, Ke Xie, Chi-Wing Fu, Hao Zhang, and Hui Huang. 2021. Continuous aerial path planning for 3D urban scene reconstruction. *ACM Trans. Graph.* 40, 6 (2021), 225–1.

Yunxiang Zhang, Kenneth Chen, and Qi Sun. 2023. Toward Optimized VR/AR Ergonomics: Modeling And Predicting User Neck Muscle Contraction. In *ACM SIGGRAPH 2023 Conference Proceedings*.

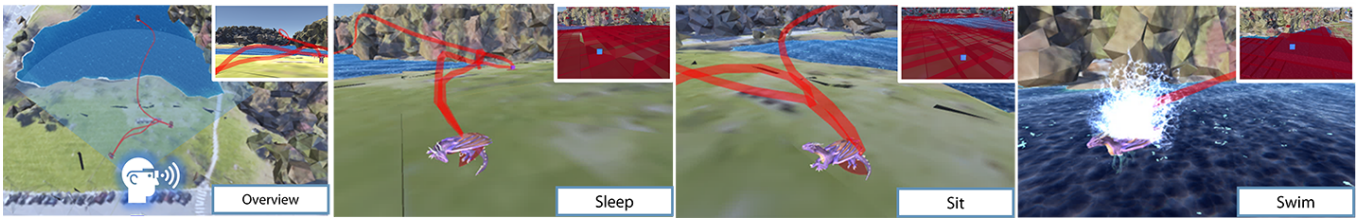
Yeyao Zhang, Eleftheria Tsipidi, Sasha Schriber, Mubbasir Kapadia, Markus Gross, and Ashutosh Modi. 2019. Generating animations from screenplays. *arXiv preprint arXiv:1904.05440* (2019).

Zhiying Zhou, Adrian David Cheok, JiunHorng Pan, and Yu Li. 2004. Magic Story Cube: an interactive tangible interface for storytelling. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology*. 364–365.

(a) Overview.

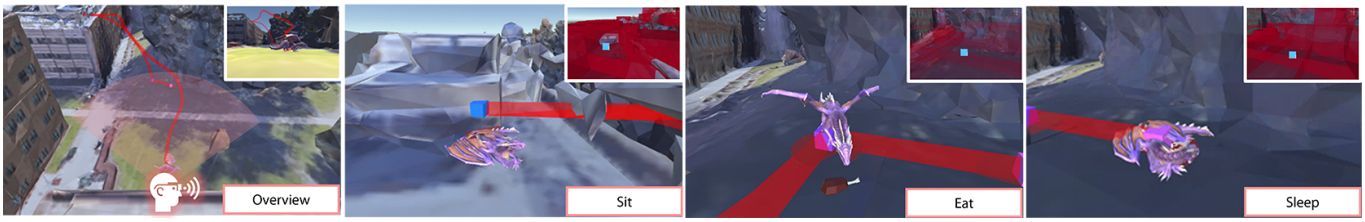


(b) Story A (*Sleep–Sit–Swim*), Location 1.



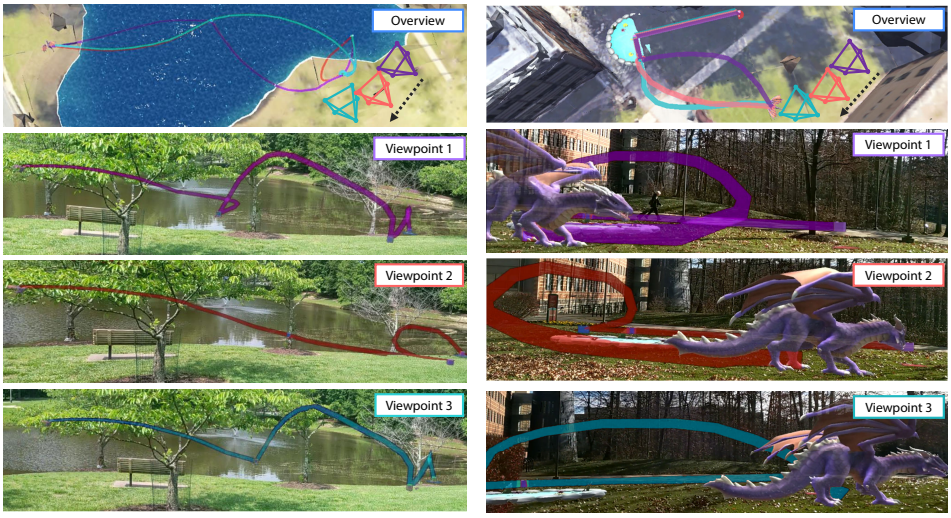
(c) Story A (*Sleep–Sit–Swim*), Location 2.

(d) Story A (*Sleep–Sit–Swim*), Location 3.



(e) Story B (*Sit–Eat–Sleep*), Location 3.

Figure 9: Waypoints and paths synthesized for a dragon on campus with respect to different story-location combinations. The face icon represents the user's position. The circular sector denotes the user's view. For each storyline action, the image at the upper-right corner shows the location compatibility map, where a high redness refers to a high compatibility with the action shown. The blue dot on the map shows the flying creature's waypoint. We generated all paths in this scene with $\lambda_l = 0.7$, $\lambda_{vw} = 0.3$, $\lambda_{vp} = 0.7$, $\lambda_d = 0.2$, and $\lambda_s = 0.1$. Note that the λ values can be adjusted to control the trade-off between different considerations such as location compatibility and visibility.



(a) Lake side. (b) Campus.

Figure 10: Two changing viewpoint results on (a) lake side and (b) campus. As the user walks from right to left, changing the viewpoint from viewpoint 1 (in purple), to viewpoint 2 (in red) and then to viewpoint 3 (in cyan), our approach recomputes the waypoints and the path according to the user's updated viewpoints. Note that in (a), the early portion of the red curve is overlaid by the cyan curve; in (b), the last portions of all the paths overlap with each other.

(a) Optimization result. (b) Real-world visualization.

Figure 11: A result generated for a user looking outside the window from a building. The optimization result and its real-world visualization are shown.

(a) Window-to-window. (b) Along the wall of a building.

Figure 12: Results generated with waypoint constraints. (a) Two waypoints (in green) are set at two windows to guide the dragon to fly through the building. (b) Two waypoints (in green) are set on the wall to guide the dragon to fly along the wall.