Synthesizing Scene-Aware Virtual Reality Teleport Graphs

CHANGYANG LI, George Mason University, USA
HAIKUN HUANG, George Mason University, USA
JYH-MING LIEN, George Mason University, USA
LAP-FAI YU, George Mason University, USA

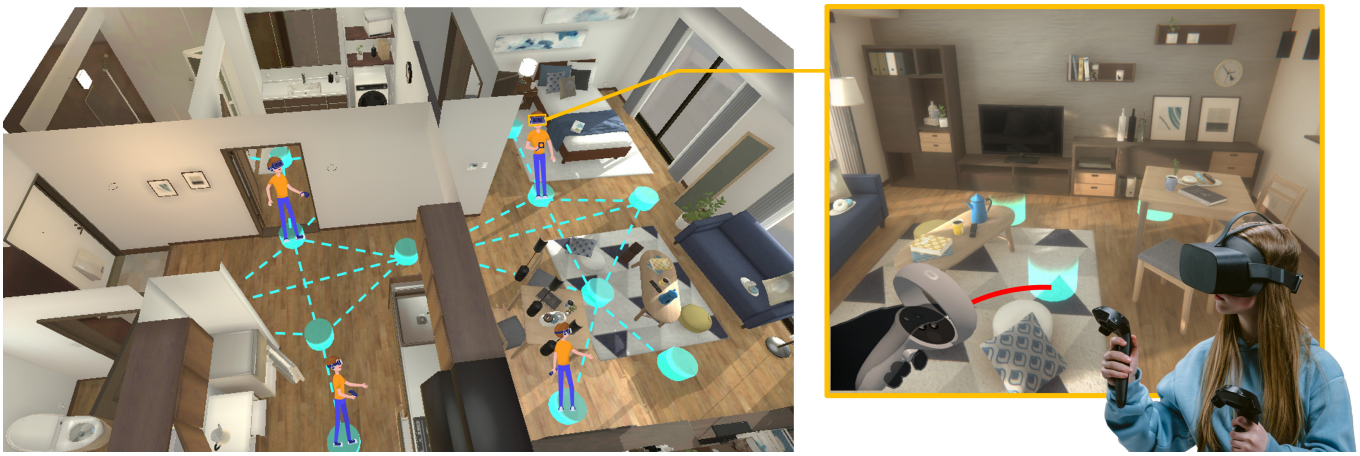


Fig. 1. Users navigate a virtual environment efficaciously via a teleport graph synthesized by our approach. Left: A teleport graph synthesized for an apartment scene. Glowing cyan anchors represent suggested teleport positions with desirable views. Right: The first-person view at a teleport position.

We present a novel approach for synthesizing scene-aware virtual reality teleport graphs, which facilitate navigation in indoor virtual environments by suggesting desirable teleport positions. Our approach analyzes panoramic views at candidate teleport positions by extracting scene perception graphs, which encode scene perception relationships between the observer and the surrounding objects, and predict how desirable the views at these positions are. We train a graph convolutional model to predict the scene perception scores of different teleport positions. Based on such predictions, we apply an optimization approach to sample a set of desirable teleport positions while considering other navigation properties such as coverage and connectivity to synthesize a teleport graph. Using teleport graphs, users can navigate virtual environments efficaciously. We demonstrate our approach for synthesizing teleport graphs for common indoor scenes. By conducting a user study, we validate the efficacy and desirability of navigating virtual environments via the synthesized teleport graphs. We also extend our approach to cope with different constraints, user preferences, and practical scenarios.

CCS Concepts: • **Computing methodologies** → **Graphics systems and interfaces**; **Virtual reality**.

Authors' addresses: Changyang Li, George Mason University, USA, cli25@gmu.edu; Haikun Huang, George Mason University, USA, hhuang25@gmu.edu; Jyh-Ming Lien, George Mason University, USA, jmlien@gmu.edu; Lap-Fai Yu, George Mason University, USA, craigyu@gmu.edu.

0730-0301/2021/12-ART229 $15.00
https://doi.org/10.1145/3478513.3480478

Additional Key Words and Phrases: teleportation, virtual reality, navigation, graph convolutional networks

ACM Reference Format:
Changyang Li, Haikun Huang, Jyh-Ming Lien, and Lap-Fai Yu. 2021. Synthesizing Scene-Aware Virtual Reality Teleport Graphs. *ACM Trans. Graph.* 40, 6, Article 229 (December 2021), 15 pages. https://doi.org/10.1145/3478513.3480478

1 INTRODUCTION

Locomotion is a fundamental task in virtual reality (VR). With the recent popularity of consumer-grade VR devices and the increasing availability of high fidelity virtual environments produced by visual computing technologies, it is crucial to develop effective locomotion techniques to help users navigate immersive virtual environments.

Among various VR locomotion techniques, real walking [Usoh et al. 1999] is the most natural one. However, real walking is constrained by the size of the physical space available. While walking-based approaches allow continuous movements, teleportation is a common VR locomotion technique [Al Zayer et al. 2018; Boletsis 2017] which supports discrete movements without requiring the user to move physically, hence lifting the play area restrictions. To teleport, the user first selects a target position and his or her virtual viewpoint will be instantly transferred there [Bozgeyikli et al. 2016].

While teleportation allows navigation in infinite virtual spaces, it could be challenging for users to choose ideal target positions in a virtual environment. There are two major challenges: (1) Users may find it hard to evaluate target positions before they arrive there. The surroundings at the target positions might deviate from what they

imagine before teleporting. (2) Users select target teleport positions using a ray cast from their perspective view. It could be difficult to select a target position precisely, especially if the position is far away. Due to these reasons, when navigating a new area, users might need to adjust their positions by teleporting several times.

In this paper, we present a scene perception (SP)-based VR teleportation approach to tackle such challenges. We propose teleport graphs whose nodes denote positions in a virtual scene and edges denote visibility among each other. Our approach synthesizes teleport graphs through an optimization by searching over the space of graphs. Our work facilitates users to visually perceive the scenes through teleportation. The goal is to enable users to efficaciously navigate a scene in virtual reality by teleporting through visually-desirable positions. By requiring pairwise visibility for connecting a pair of nodes, our approach ensures graph connectivity thus users could traverse the whole scene continuously via a teleport graph. Figure 1 depicts a synthesized teleport graph. Note that teleport graphs are compatible with conventional teleportation techniques.

We encode several considerations as cost terms of the optimization, one of which is based on a learned data-driven model of position desirability. More specifically, we trained a graph attention network (GAT) [Veličković et al. 2017], which outperformed a convolutional neural network (CNN) in our experiment, to predict position desirability in indoor apartment scenes, based on scene perception relationships modeled in scene perception (SP) graphs. The SP graphs are extracted from RGB-D panoramic images which present observers' first-person views, and can be regarded as high-level abstractions of the spatial relationships between observers and the nearby objects. Meanwhile, other cost terms incorporate properties, including coverage, connectivity, etc. that are exclusive to the VR teleportation task. While our approach can synthesize teleport graphs for a variety of indoor apartment scenes automatically, we also show possible extensions for handling other scene types.

Our approach is useful for compelling use case scenarios. For example, it can serve as a tool for laying out teleport graphs for large procedurally generated virtual environments or user-generated VR maps. Our approach may also help people with motor impairments (e.g., people who have difficulty with fine-grained controller manipulations) to navigate VR scenes.

The major contributions of our work include the following:

- We propose scene perception graphs to encode scene perception relationships from panoramic images; and apply a GAT model to learn and evaluate these relationships.
- We model the set of teleport positions into teleport graphs to handle important VR teleportation considerations; and employ an optimization approach to synthesize teleport graphs.
- We evaluate the efficacy and desirability of the teleport graphs synthesized by our approach for supporting VR navigation through a user study.

2 RELATED WORK

2.1 Virtual Reality Locomotion

Compared to other VR locomotion techniques (e.g. joystick-based locomotion [Boletsis 2017]), real walking is more natural and immersive, resulting in higher subjective presence [Slater et al. 1995; Usoh et al. 1999]. However, a major restriction of real walking is the limited size of a physical VR play area. One promising solution is redirected walking [Dong et al. 2019; Razzaque et al. 2001; Sun et al. 2018], which creates a distorted mapping from the physical space to the virtual environment, enabling users to navigate a large virtual world. Another solution is walking-in-place, where a user stays at a fixed physical position while walking in the virtual world. Such an approach is usually realized with the help of hardware such as a Virtuix Omni treadmill. Besides walking-based techniques, world-in-miniature [Stoakley et al. 1995] or mini-map [Mahalil et al. 2019] is also a famous VR locomotion technique. Refer to recent surveys by Boletsis [2017] and Zayer et al. [2018] for a comprehensive review of VR locomotion techniques.

2.2 Teleportation

Teleportation is a widely used VR locomotion technique [Boletsis 2017]. Previous works have compared teleportation with natural walking [Sayyad et al. 2020] and joystick-based locomotion [Buttussi and Chittaro 2019; Langbehn et al. 2018]. The most straightforward version of teleportation is "point & teleport" [Bozgeyikli et al. 2016], which translates the virtual viewpoint to the destination instantly. To address the problem of disorientation in teleportation, some variations were introduced, like gradually moving the viewpoint [Bhandari et al. 2018] or allowing users to specify target orientations [Bozgeyikli et al. 2016; Funk et al. 2019]. While those teleportation strategies require users to manually specify destinations, such specifications could be inefficacious due to the lack of prior knowledge about the scene and how desirable the target positions are. [Habgood et al. 2018] studied the differences between free teleport and a node-based locomotion technique. Our approach complements VR teleportation by analyzing a virtual scene and suggesting desirable positions, and is compatible with real-world 3D scans, manually created virtual environments, and procedurally generated virtual scenes [Feng et al. 2016; Qi et al. 2018b].

2.3 Scene Perception Graph

Previous works studied the problem of improving VR locomotion experiences by evaluating viewpoint quality and scene visibility [Freitag et al. 2016, 2017, 2018]. We address this problem using a general and flexible representation of viewpoint information. In recent work, human-object interaction (HOI) structures are modeled using HOI graphs [Li et al. 2019; Qi et al. 2018a; Ulutan et al. 2020]. Based on object detection results, human nodes and objects nodes are extracted, and their relations are encoded in edges. Graph representations are general, informative, and flexible, as they comprise a varying number of nodes and edges. Inspired by the notion of HOI graphs and the concept of *scene perception* [Geisler 2008; Rensink 2000] taken as the "visual perception of an environment as viewed by an observer at any given time", we propose scene perception (SP) graphs to model an observer's perception of nearby objects together with their relative spatial relationships.

Based on the SP graph representation, we further come up with its flatted sequence representation for comparing different graphs.

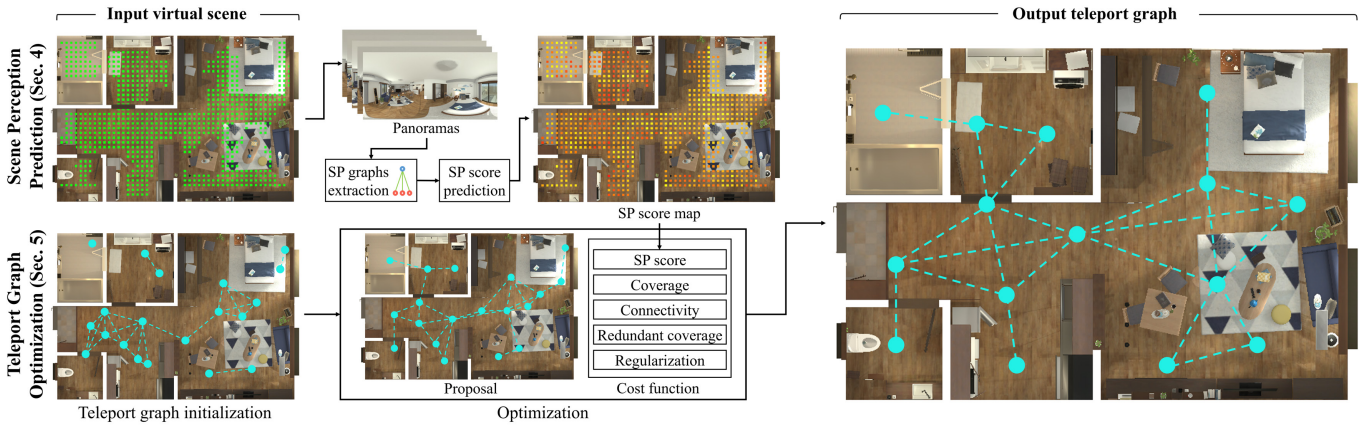


Fig. 2. Given an input virtual scene, our approach goes through two stages to synthesize a teleport graph. First, in the scene perception (SP) prediction stage (Section 4), our approach samples a set of candidate positions within the free space. SP graphs are extracted from the panoramic views and passed to the prediction model to compute an SP score map. Second, in the teleport graph optimization stage (Section 5), our approach synthesizes a desirable teleport graph through an optimization by considering SP scores, coverage, connectivity, etc.

Such a representation refers to a statistical feature akin to the bag-of-words model widely applied in computer vision as a vector of occurrence counts of a visual feature vocabulary [Fei-Fei 2007; Zhang et al. 2010]. In forming the flatted sequence of an SP graph, however, we do not build a vocabulary and do not fix the sequence length, since we would like a flexible representation to encode what objects and how many of them are detected. Thus, the sequence length corresponds to the specific SP graph it represents.

3 OVERVIEW

We devise an approach that automatically analyzes an input virtual indoor scene and suggests teleport positions through which users can navigate the scene efficaciously. Figure 2 shows an overview of our approach comprising two stages.

Scene Perception Score Prediction. Our approach first samples candidate positions in the free space. It then synthesizes a panoramic image at each position for SP score prediction. A panoramic image shows the observer's first-person view, thus conveys the scene perception relationships among the observer and objects in the surrounding environment. Our model extracts SP graphs from the panoramic RGB-D images and passes them to a predictor to generate their corresponding SP scores. These scores indicate the desirability or human preferences of the positions. Hence an SP score map as depicted in Figure 2 can be computed for the virtual scene. Section 4 discusses the technical details of this step.

As it could be challenging to define rules to explicitly determine what positions in virtual scenes are desirable by humans, we use a GAT [Veličković et al. 2017] model to learn and predict human preferences of viewing positions. We use human-marked positions in virtual scenes from the Gibson Environment [Xia et al. 2018] for training, following a rational choice assumption [Hedström and Stern 2008] that such human-marked positions are optimal.

Teleport Graph Optimization. Based on the predicted SP score map, we not only select desirable positions with high SP scores, but also consider the interrelation among the selected positions to enable efficacious navigation of the virtual scene by a user. For example, the connectivity among nodes is considered to ensure a user could traverse a whole scene continuously. To this end, our approach synthesizes a teleport graph whose nodes encode the selected positions and whose edges encode visibility between nodes. We formulate our approach as an optimization problem to search for a teleport graph considering properties such as SP score, coverage, connectivity, etc. which are encoded as cost functions. As sampling the teleport graph throughout the optimization process might involve dimensionality-changing actions, such as addition of nodes, we employ the reversible-jump Markov chain Monte Carlo (RJMCMC) [Green 1995] sampling strategy to maintain the detailed balance condition. The optimization synthesizes a desirable teleport graph for the scene. Section 5 discusses the optimization process.

4 SCENE PERCEPTION PREDICTION

Given a virtual indoor scene as input, the first step of our approach is to evaluate candidate positions in the free space in terms of their desirability represented as SP scores. In this section, we describe the details of SP score prediction. We extract SP graphs based on object detection results on panoramic RGB-D images rendered at the candidate positions. We use panoramic images because they correspond to a user's comprehensive first-person views at the candidate positions in virtual reality.

In our work, we define the desirability of a candidate position as the maximum similarity between the SP graph of this candidate and that of an example in an optima set, following the rational choice assumption [Hedström and Stern 2008]. We define a graph kernel to measure this similarity and further use it to annotate a collected SP dataset for training a GAT-based SP predictor.

Note that we do not use visual saliency [Itti et al. 1998] to evaluate scene desirability because a view with high saliency does not necessarily appear more desirable. For example, most people would avoid standing very close to a picture, even though this would likely put salient content in their fields of view. Instead, we learn scene perception encoding human preferences in a data-driven manner.

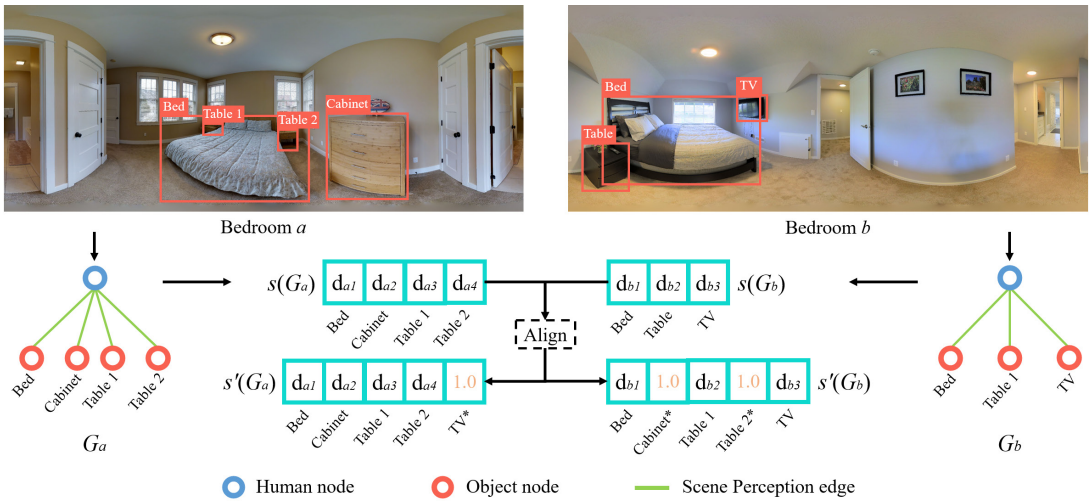


Fig. 3. An example process of aligning the flattened sequences of two SP graphs G_a and G_b. The SP graphs are extracted from the panoramic images and transformed into the flattened sequences $S(G_a)$ and $S(G_b)$. To align the sequences, fake object nodes (denoted by $*$) are added to each sequence so that their object categories and numbers match. The d_{ai} and d_{bi} are the depth features of the objects in G_a and G_b. The fake nodes are assigned with a depth feature $d = 1.0$, which means the objects are infinitely far away.

4.1 Scene Perception Graph Extraction

Akin to the human-object interaction (HOI) graph generation process, a powerful and reliable object detector is essential for SP graph extraction. With object detection, our method could be applied in general virtual environments without semantic annotations. As we focus on indoor apartment scenes, we train a detection model on Indoor-360 dataset [Chou et al. 2020]. Specifically, we use the Faster R-CNN [Ren et al. 2015] + FPN [Lin et al. 2017] model with a ResNet101 [He et al. 2016] backbone. Following the settings in [Chou et al. 2020], we replace the conventional Convolutional Neural Networks (CNNs) by SphereNets [Coors et al. 2018] for learning the invariance of distortions in equirectangular images.

We model SP graphs from a first-person perspective: In an equirectangular panoramic image, the observer, who stands at the center of the position, perceives the surroundings with a 360° view. Therefore, there are two types of nodes in an SP graph: (1) **A human node**, which is independent of the object detector and is implicitly initialized; (2) **Object nodes**, which rely on the object detection results. Every detected object results in an object node in the graph, and an undirected edge between this object node and the human node. Among all the 37 object detection label categories in the Indoor-360 dataset, the label "person" is the 0th label and is assigned only to the human node. Any object detected as "person" in the panoramic images is removed. Figure 3 shows examples SP graphs extracted from bedroom panoramic images.

To evaluate the similarity between different views in a robust manner, our SP graphs use high-level semantic features rather than low-level visual features. As a result, two scenes with consistent furniture layouts but different object textures will be determined as highly similar. The SP graphs use two types of node features: (1) **Category feature**, which corresponds to semantic labels of detected objects by the object detector; (2) **Depth feature**, which represents the spatial relationship between the observer and objects nearby. In our implementation, we extract a depth feature by averaging the depth values within the region of a detected object's bounding box. We normalize the depth feature to $[0, 1]$, where 0 denotes zero distance and 1 denotes being infinitely far away.

Formally, an SP graph is defined as $G = (V, E)$, where V is the set of nodes and E is the set of edges. $v_i \in V$ denotes a node, where $V = \{v_0, v_1, \ldots, v_n\}$ and n is the number of detected objects. v_0 is the human node and all the other nodes are object nodes. $e_{i,j} \in E$ denotes an edge pointing from v_i to v_j, where $E = \{e_{0,1}, e_{0,2}, \ldots, e_{0,n}\}$. The set of node features $H = \{\overrightarrow{h_0}, \overrightarrow{h_1}, \ldots, \overrightarrow{h_n}\}$, where $\overrightarrow{h_i} = (c_i, d_i)$ is a tuple of category feature c_i and depth feature d_i. Specifically, $\overrightarrow{h_0} = (0, 0)$, which corresponds to the human node.

4.2 Scene Perception Score

Determining the desirability of a candidate position based on its panoramic view is challenging as this is a subjective problem and is hard to model by deterministic rules and criteria. We tackle this problem in a data-driven manner. We collect a set of human-marked positions as optimal examples used as references (described in Section 4.3). We propose a graph kernel to measure the similarity (in the range of $[0, 1]$) between two SP graphs. The SP score of a candidate position is defined as the maximum similarity between its SP graph and the SP graph of an optimal example. Figure 4 shows an example of applying our graph kernel to measure the similarity scores of three example panoramic images with respect to a reference.

Graph kernels [Vishwanathan et al. 2010], like the random walk graph kernel [Gärtner et al. 2003] and the Weisfeiler-Leman graph kernel [Shervashidze et al. 2011], are commonly used for measuring the similarity between a pair of graphs. In our problem, when comparing SP graphs, we consider the fact that the SP graphs have a fixed structure in general. Therefore, we define a graph kernel $K(\cdot)$ exclusively for comparing SP graphs. Before defining $K(\cdot)$, we introduce a function $s(\cdot)$ that transforms an SP graph into a flattened

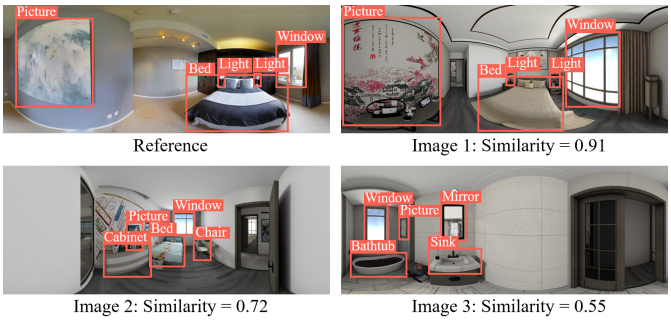


Fig. 4. Three example images and their similarities to the reference computed by our graph kernel. In each image, the top five object detection results are shown with bounding boxes. Image 1 is the most similar to the reference in terms of the object detection results and depth features, hence the highest similarity score. Image 3 has the lowest similarity score.

sequence as $s(G) = (v_{s1}, v_{s2}, \ldots, v_{sn})$. The human node v_0 is ignored in a sequence $s(G)$ as human nodes have a uniform representation in all SP graphs according to our definition. Nodes are sorted first by the category feature and then by the depth feature. Note that G and $s(G)$ are equivalent; flattening G does not lose information. Refer to the supplementary material for further discussion.

When comparing two sequences $s(G_a)$ and $s(G_b)$ of two SP graphs G_a and G_b, we further transform the sequences into aligned sequences $s'(G_a)$ and $s'(G_b)$, which have the same number of nodes for each category to enable computing the cosine similarity. The transformation is performed as follows: for each category of node present in $s(G_a)$ or $s(G_b)$, suppose, without loss of generality, $s(G_a)$ has fewer nodes of that category. Then we append $s(G_a)$ with "fake" nodes of that category with a depth value of 1.0 (meaning infinitely far away) so that $s'(G_a)$ and $s'(G_b)$ have the same number of nodes of that category. Figure 3 shows an example. $s(G_a)$ has no "TV" node and $s(G_b)$ has one "TV" node. So a "TV" node with a depth value of 1.0 is added to $s(G_a)$ in forming $s'(G_a)$. Based on the aligned sequences $s'(G_a)$ and $s'(G_b)$, we define the graph kernel as:

$$K(G_a, G_b) = \frac{s'(G_a) \cdot s'(G_b)}{|s'(G_a)||s'(G_b)|}. \tag{1}$$

4.3 Dataset

We extracted a set of 5,000 panoramic images at human-marked positions as optimal examples from the Gibson Environment [Xia et al. 2018], which comprises a large-scale database of spaces of 572 buildings consisting of 1,447 floors covering a total area of 211,000m^2. Each space comes with a set of RGB panoramic images (e.g., one captured image per 5-10m^2), with global camera poses and reconstructed 3D meshes. Similar to [Zhu et al. 2016] which follows the rational choice assumption [Hedström and Stern 2008], we assume that positions in the scenes where the authors of the Gibson Database chose to capture equirectangular panoramic images are desirable. We conducted a pilot study to validate this assumption whose details are included in the supplementary material.

In principle, running the graph kernel directly to produce SP scores for novel panoramic views is feasible but computationally inefficient. Therefore, we propose to learn and predict scene perception in a data-driven manner. We constructed an SP dataset, which consists of 20,000 synthesized panoramic images (together with their SP graphs) at randomly sampled positions in the Gibson Environment, for training an SP predictor. The ground truth SP score label of a sample is set as its similarity to the closest optimal example according to $K(\cdot)$. We used 70% of the data for training, 15% for validation, and 15% for testing.

While the SP dataset is for training and testing the SP predictor, which is a key component in our pipeline, we also built another 16-scene dataset for testing the whole teleport graph synthesis approach. This dataset consists of 16 virtual indoor scenes: 12 of the scenes are synthestic, while the other 4 are reconstructed 3D scans extracted from the Matterport3D dataset [Chang et al. 2017] which help validate the generalization ability of our approach. Each scene has complete 3D meshes and textures. The ground truth SP score of a sampled position in free spaces is also set as its similarity to the closest optimal example.

4.4 Prediction via Graph Convolution

To handle the input represented as SP graphs, we train a graph convolutional network (GCN) for SP score prediction. Compared to previous convolutional neural networks, GCNs do not require the input data to be organized in an Euclidean domain [Zhang et al. 2019]. To incorporate attention mechanisms [Bahdanau et al. 2014; Hu 2019], we apply a graph attention network (GATs) [Veličković et al. 2017] in practice, allowing the model to focus on the most relevant parts of the input rather than putting attention evenly on every component, which fits with our problem setting where the user would pay attention to objects in the surrounding.

Graph Attentional Layer. We briefly illustrate how a single GAT layer works. In the GAT, the self-attention operation is performed on the nodes to compute attention coefficients:

$$\beta_{ij} = a(\mathbf{W}\overrightarrow{h_i}, \mathbf{W}\overrightarrow{h_j}) = LeakyReLU(\overrightarrow{\mathbf{a}}^T(\mathbf{W}\overrightarrow{h_i} \parallel \mathbf{W}\overrightarrow{h_j})). \tag{2}$$

In this step, $\mathbf{W} \in \mathbb{R}^{F' \times F}$ is a learnable weight matrix that performs a shared linear transformation on the input node feature, where F is the number of node features and F' is the size of node feature embedding. The coefficient β_{ij} denotes the importance of z_j to z_i, where there exists an edge e_{ij} connecting node v_i to its neighbor node v_j. GAT employs an attentional mechanism $a(\cdot)$, which is parameterized by a weight vector $\overrightarrow{\mathbf{a}} \in \mathbb{R}^{2F'}$ and activated by a $LeakyReLU$ function. Here $||$ denotes concatenation. Next, these coefficients are normalized using a softmax function:

$$\alpha_{ij} = \frac{e^{\beta_{ij}}}{\sum e^{\beta_{ik}}}. \tag{3}$$

Analogous to using multiple channels in CNNs, GAT allows multi-head attention. By employing K attention heads, node features are updated with an activation function σ after a round of graph convolution by message passing as follows:

$$\overrightarrow{h_i'} = \sigma(\frac{1}{K}\sum_k \sum_j \alpha_{ij}^k \mathbf{W}^k \overrightarrow{h_j}). \tag{4}$$

Network Structure. Our SP predictor includes five GAT layers. Represent each GAT layer as (F, F', K). The five GAT layers are parameterized as: $(2, 256, 4)$, $(256, 128, 2)$, $(128, 64, 2)$, $(64, 32, 1)$, and

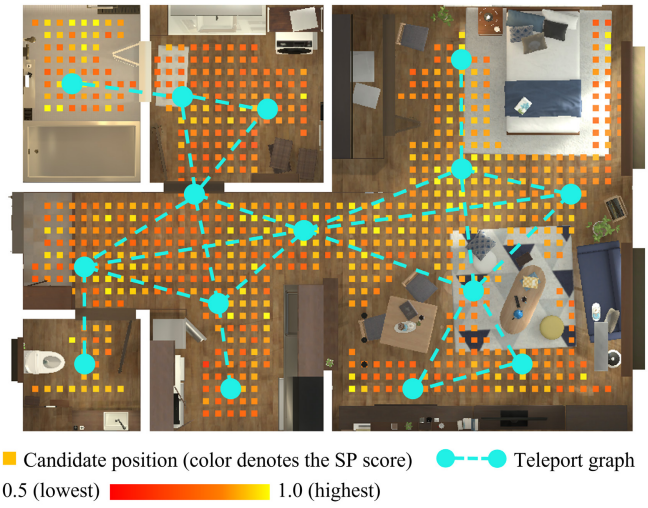


Fig. 5. A teleport graph example. Small squares refer to the candidate positions $\mathcal{P} = \{p_i\}$, whose colors correspond to their SP scores $\{\hat{y}_i\}$. A teleport graph $\mathcal{G}$ is shown, which comprises a set of nodes $\mathcal{V}$ depicted in cyan and a set of edges $\mathcal{E}$ represented as dashed lines.

$(32, 16, 1)$, respectively. We use *ReLU* as the activation function σ. On an SP graph, messages are passed in both directions from the human to the objects, and from the objects to the human. This message passing manner resembles the scenario of scene perception, where the human observes the environment and receives visual information from the surrounding objects.

After the five-layer graph convolution, we only extract the final feature of the human node $\overrightarrow{h_0'}$ and pass it into a readout function $R(\cdot)$ to predict a score y:

$$y = R(\overrightarrow{h_0'}). \tag{5}$$

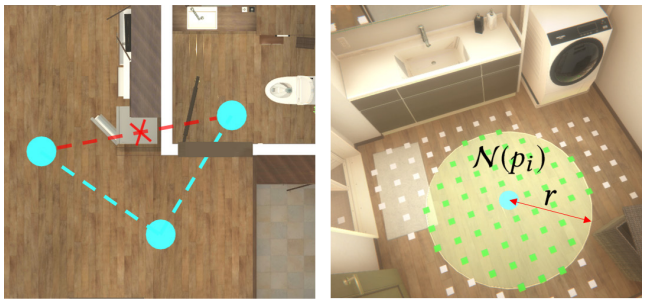
In our implementation, we represent $R(\cdot)$ as a multi-layer perceptron (MLP). The predicted SP scores will be used to prompt the teleport graph to include positions with high scores. Such a consideration is incorporated as a cost term in the teleport graph optimization discussed in Section 5.2.

5 TELEPORT GRAPH OPTIMIZATION

Our approach aims to produce desirable positions by considering SP scores and other properties for supporting VR teleportation. The position set is modeled using a teleport graph $\mathcal{G}$. We incorporate a few constraints on $\mathcal{G}$ into a cost function comprising several cost terms. Based on the optimization formulation, we synthesize a desirable teleport graph through a sampling process.

5.1 Teleport Graph Representation

We sample a set of candidate positions in the free space of the input virtual scene using a grid with a 0.2m cell length. Let $\mathcal{P} = \{p_i\}$ be the set of all candidate positions. Applying the trained SP predictor (Section 4) to these candidate positions, we obtain a set of predicted SP scores $\{\hat{y}_i\}$ correspondingly as a score map. Let $M(\cdot)$ be a mapping function that maps a candidate position to its SP score, i.e. $M(p_i) = \hat{y}_i$. Figure 5 depicts $\mathcal{P}$ and $\{\hat{y}_i\}$.



(a) Visibility (b) Neighborhood

Fig. 6. Node relationships. (a) Visibility among three nodes. An edge is formed between each pair of nodes that are visible to each other. No edge is added between the pair of nodes occluded by a wall. (b) The neighborhood of the cyan node p_i is highlighted as a circle. The candidate positions within the neighborhood are shown in green.

We define a teleport graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes corresponding to the selected positions and $\mathcal{E}$ is the set of edges. Figure 5 shows an example. As each node corresponds to a unique position, for notation convenience and brevity, we represent each node as its corresponding position. Hence $\mathcal{V} = \{p_i^\dagger\}$ is a subset of $\mathcal{P}$, where $p_i^\dagger$ is a selected position.

Considering the connectivity between two selected positions $p_a^\dagger$ and $p_b^\dagger$, whether to add an undirected edge $(p_a^\dagger, p_b^\dagger)$ into $\mathcal{E}$ depends on the pairwise adjacent visibility, as follows. Under a VR teleportation setting, if the user is currently at position $p_a^\dagger$, he or she can teleport to position $p_b^\dagger$ only if position $p_b^\dagger$ is visible. In practice, we cast a ray from position $p_a^\dagger$ to position $p_b^\dagger$ to determine visibility. If the positions are visible to each other and the distance between them is shorter than a threshold (8m in our implementation), we add an edge to connect them. Figure 6(a) shows an example. Note that the visibility check is performed in 3D space even though Figure 6(a) shows a top-down 2D view. Moreover, if there exists another position between two positions, no edge is added between them even though they are visible to each other.

As the edge set $\mathcal{E}$ is deterministically defined by the node set $\mathcal{V}$, the node set $\mathcal{V}$ itself corresponds to a teleport graph configuration in our search space. In other words, once the node set $\mathcal{V}$ is determined, the corresponding edge set $\mathcal{E}$ can be computed, thus $\mathcal{E}$ changes as $\mathcal{V}$ is updated in the optimization process.

We further define a position's neighborhood region, within which a user can move by walking. The neighborhood region is a circle with a radius r centered at position p_i. Based on our observations that VR users tended to walk only within a small region around a teleport position and to move farther away by triggering a teleportation, we set $r = 1.0$m empirically. Figure 6(b) shows an illustration. The set of candidate positions within the neighborhood of position p_i is defined as:

$$\mathcal{N}(p_i) = \{p_j | p_j \in \mathcal{P} \wedge |p_i - p_j| < r\}. \tag{6}$$

5.2 Cost Function

We define a cost function to evaluate teleport graphs, which consists of five cost terms for evaluating different properties. Our goal is

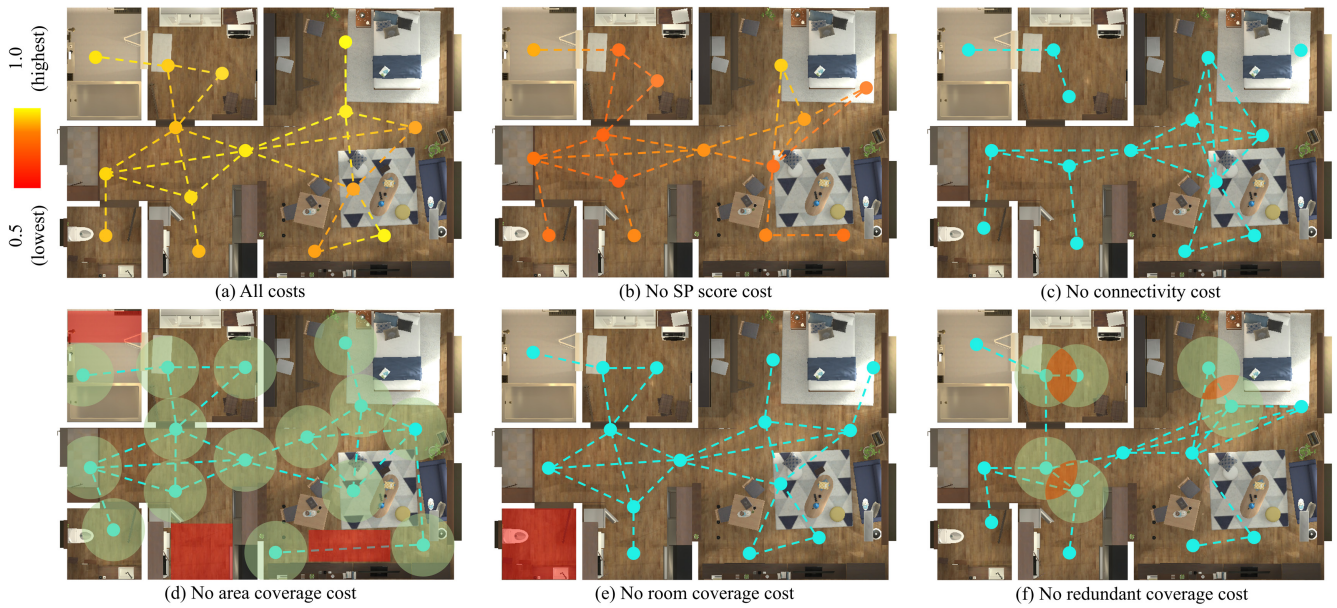


Fig. 7. Effects of omitting costs. (a) A result synthesized by considering all costs. (b) Without the SP score consideration, positions with low SP scores are sampled. (c) Without the connectivity consideration, disconnected graphs are synthesized; users cannot traverse the whole scene by teleporting continuously as a result. (d) Without the area coverage consideration, the areas highlighted in red are uncovered and unexplored. (e) Without the room coverage cost, the bathroom at the lower left is uncovered. (f) Without the redundant coverage consideration, the neighborhood areas of some nodes overlap severely.

to synthesize a teleport graph that makes the VR navigation: (1) *desirable*, achieved by high SP scores of the included positions; (2) *complete*, achieved by high coverage; and (3) *efficacy*, achieved by avoiding redundant coverage and enforcing connectivity. Our approach synthesizes a teleport graph by optimizing an overall cost function composed of a weighted sum of cost terms:

$$\begin{aligned} C(\mathcal{G}) =& w_{\text{SP}}C_{\text{SP}}(\mathcal{G}) + w_{\text{COV}}C_{\text{COV}}(\mathcal{G}) + w_{\text{RCOV}}C_{\text{RCOV}}(\mathcal{G}) \\ &+ w_{\text{CON}}C_{\text{CON}}(\mathcal{G}) + w_{\text{REG}}C_{\text{REG}}(\mathcal{G}). \end{aligned} \tag{7}$$

SP Score. Positions with high SP scores suggest that they might offer desirable visual views and are more likely favored by users. To encourage the teleport graph $\mathcal{G}$ to include positions with high SP scores, we compute the mean SP score of $\mathcal{G}$ and penalize a low mean score. We include an SP score cost defined as follows:

$$C_{\text{SP}}(\mathcal{G}) = (1 - \frac{\sum_{p_i^\dagger \in \mathcal{V}} M(p_i^\dagger)}{|\mathcal{V}|})^2. \tag{8}$$

Coverage. A teleport graph is expected to help users navigate the virtual scene efficaciously. Therefore we want the teleport graph $\mathcal{G}$ to cover as much free space as possible. Meanwhile, we do not want the graph to be too dense. The positions should keep reasonable distances from each other. To strike this balance, we define that a position is covered if it is within the neighborhood area of any position in the set of selected candidate positions $\mathcal{V}$. Formally, we define the set of covered positions as:

$$\mathcal{V}_{\text{COV}} = \{p_i | \exists p_j^\dagger \in \mathcal{V} \implies p_i \in \mathcal{N}(p_j^\dagger)\}. \tag{9}$$

With this definition, we define two sub-terms to evaluate area and room coverage as follows.

Area Coverage:. As we sample positions to represent the whole free space, the portion of covered area is approximated by $\frac{|\mathcal{V}_{\text{COV}}|}{|\mathcal{P}|}$. We define the following term to penalize low area coverage:

$$C_{\text{AC}}(\mathcal{G}) = (1 - \frac{|\mathcal{V}_{\text{COV}}|}{|\mathcal{P}|})^2. \tag{10}$$

Room Coverage:. With the area coverage term only, some rooms are probably uncovered. Such a problem likely occurs in large indoor scenes with some small rooms (e.g., bathrooms) as the small rooms only constitute a relatively small portion of the whole space and their coverage has limited influence on the area coverage cost. However, a teleport graph is supposed to be complete in the sense that it should cover all rooms whether they are large or small. To this end, we regard a room as being covered if there is at least one selected position $p_i^\dagger \in \mathcal{P}$ inside this room. Let φ be the percentage of covered rooms over all rooms. We define the room coverage cost as:

$$C_{\text{RC}}(\mathcal{G}) = (1 - \varphi)^2. \tag{11}$$

Combining the sub-terms, the coverage cost is defined as:

$$C_{\text{COV}}(\mathcal{G}) = w_{\text{AC}}C_{\text{AC}}(\mathcal{G}) + w_{\text{RC}}C_{\text{RC}}(\mathcal{G}), \tag{12}$$

where w_{AC} and w_{RC} are the relative weights of these sub-terms.

Redundant Coverage. While we encourage the teleport graph to cover as much space as possible, we also want the selected positions to spread out so that they do not cover the same space redundantly. We define that a position is redundantly covered if it is covered more than once. We define the over-covered position set as:

$$\mathcal{V}_{\text{RCOV}} = \{p_i | \exists p_j^\dagger, p_k^\dagger \in \mathcal{V} \implies p_i \in \mathcal{N}(p_j^\dagger) \wedge p_i \in \mathcal{N}(p_k^\dagger)\}. \tag{13}$$

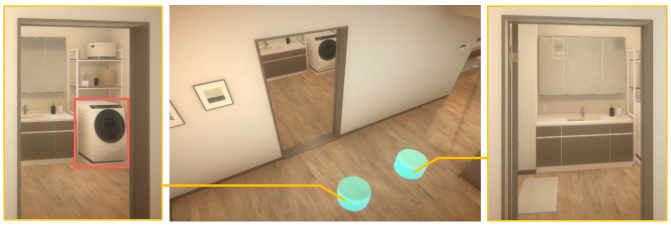


Fig. 8. Positions in a small neighborhood may have considerably different SP scores due to changes in object visibility. In this example, a washer is visible at the left position but not at the right position.

We define a cost term to penalize redundant coverage of areas:

$$C_{\mathrm{RCOV}}(\mathcal{G}) = (\frac{|\mathcal{V}_{\mathrm{RCOV}}|}{|\mathcal{P}|})^2, \tag{14}$$

where $\frac{|\mathcal{V}_{\mathrm{RCOV}}|}{|\mathcal{P}|}$ approximates the portion of redundantly covered area. Essentially, this cost term encourages the teleport graph to include positions that are apart from each other so that their neighborhood regions do not overlap.

Connectivity. To allow smooth and convenient VR navigation, it is important to ensure the connectivity of the teleport graph $\mathcal{G}$. As the edges of a teleport graph are constructed according to the pairwise adjacent visibility of the positions, only a connected graph allows a user to start navigating the virtual scene at an arbitrary teleport position and teleport through all other positions, traversing the whole scene. In other words, if there are more than one connected components in a teleport graph, a user would not be able to teleport through the scene in a continuous manner. Therefore we include a connectivity cost to penalize a teleport graph if its number of connected components κ is larger than one:

$$C_{\mathrm{CON}}(\mathcal{G}) = 1 - e^{1-\kappa}. \tag{15}$$

Regularization. We regularize the number of selected positions to allow efficacious exploration of the solution space by our sampling process, which we discuss in Section 5.3. We analyze the scene data of the Gibson Database to obtain an estimate of the number of positions present in a teleport graph $\mathcal{G}$ for a virtual scene. More specifically, based on the scene data with human-marked positions from the Gibson Environment, we fit a polynomial function $\lambda(|\mathcal{P}|)$ to give an expected number of selected positions based on the size of the virtual scene (estimated by the total number of candidate positions $|\mathcal{P}|$). We include the following regularization cost to keep the number of positions $|\mathcal{V}|$ close to $\lambda(|\mathcal{P}|)$:

$$C_{\mathrm{REG}}(\mathcal{G}) = \begin{cases} 0 & \text{if } |\mathcal{V}| \leqslant \lambda(|\mathcal{P}|) \\ \gamma_1^{|\mathcal{V}|-\lambda(|\mathcal{P}|)+\gamma_2} & \text{otherwise,} \end{cases} \tag{16}$$

where $\gamma_1 = 2$ and $\gamma_2 = -5$ are hyperparameters. This cost softly adds a penalty as the number of nodes is over $\lambda(|\mathcal{P}|)$.

This cost term strikes a balance of acceptance probabilities between an *Add* action A_{a} and a *Delete* action A_{d}, which we discuss in the supplementary material. Note that this cost could reach $\gamma_1^{|\mathcal{P}|-\lambda(|\mathcal{P}|)+\gamma_2}$ at most while all the other costs are in range $[0, 1]$.

Unless otherwise specified, we set the weights as $w_{\mathrm{SP}} = 3.0$, $w_{\mathrm{COV}} = 1.5$ (with $w_{\mathrm{AC}} = 0.7$ and $w_{\mathrm{RC}} = 0.3$), $w_{\mathrm{RCOV}} = 1.0$, $w_{\mathrm{CON}} = 2.0$, and $w_{\mathrm{REG}} = 1.0$. Figure 7 shows the effects of omitting costs (except the regularization cost) on the synthesized teleport graph. Note that we used Figure 7(a) as the initialization to synthesize (b) to (f) for comparison and illustration. Even though (a) and (b) have similar graph structures, their SP scores differ considerably. As Figure 8 shows, a small change in position in a neighborhood may lead to significant SP score differences due to changes in object visibility, caused by occlusion, for example.

5.3 Graph Optimization

We use simulated annealing [Kirkpatrick et al. 1983] to efficiently explore the solution space containing various teleport graph solutions. We define a Boltzmann-like objective function:

$$f(\mathcal{G}) = e^{-\frac{C(\mathcal{G})}{t}}, \tag{17}$$

where t is the temperature parameter in simulated annealing. The optimization proceeds iteratively. For initialization, we randomly sample $\lambda(|\mathcal{P}|)$ positions from $\mathcal{P}$ and add them to $\mathcal{V}$. Based on our cost terms, we expect that the number of selected positions is around $\lambda(|\mathcal{P}|)$. According to our definition of edges in a teleport graph, $\mathcal{E}$ is determined by $\mathcal{V}$ in each iteration, and they form a complete teleport graph representation $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Simulated annealing is controlled by the temperature parameter t. As the optimization begins, t is set to be high to prompt the optimizer to aggressively explore possible solutions. The temperature drops over iterations by a decay factor until it reaches a low value near zero. Such a setting essentially makes the optimizer more greedy in refining the solution towards the end of the optimization. The optimization process is terminated as the change in the total cost $C(\mathcal{G})$ converges (less than 3% over the past 100 iterations).

A teleport graph is sampled at each iteration. As a position may be added or removed from the teleport graph, the number of selected positions $|\mathcal{V}|$ may vary. To handle dimensionality-changing cases, we apply the Reversible-jump Markov chain Monte Carlo technique [Green 1995] to maintain the detailed balance condition.

5.4 Proposed Actions

In each optimization iteration, one of the following actions is randomly applied to modify the teleport graph:

- *Add* a position: Randomly pick an unselected position $p_i \in \mathcal{P} - \mathcal{V}$ and add it to $\mathcal{V}$.
- *Delete* a position: Randomly pick a selected position $p_i^\dagger \in \mathcal{V}$ and delete it from the teleport graph.
- *Move* a position: Randomly pick a selected position $p_i^\dagger \in \mathcal{V}$ and an unselected position p_j in its neighborhood such that $p_j \in \mathcal{N}(p_i^\dagger)$ and $p_j \notin \mathcal{V}$. Replace $p_i^\dagger$ with p_j (i.e. remove $p_i^\dagger$ from $\mathcal{V}$ and add p_j to $\mathcal{V}$).

The probabilities of choosing the three actions are $\mathbb{P}_{\mathrm{a}} = 0.2$, $\mathbb{P}_{\mathrm{d}} = 0.2$, and $\mathbb{P}_{\mathrm{m}} = 0.6$, respectively. The acceptance probabilities of the three actions are defined based on the RJMCMC formulation.

- The *Add* action is dimensionality-changing with acceptance probability:

$$A_{\mathrm{a}}(\mathcal{G}^*|\mathcal{G}) = \min(1, \frac{f(\mathcal{G}^*)}{f(\mathcal{G})} \frac{\mathbb{P}_{\mathrm{d}}}{\mathbb{P}_{\mathrm{a}}} \frac{|\mathcal{P} - \mathcal{V}|}{|\mathcal{V}^*|}). \tag{18}$$

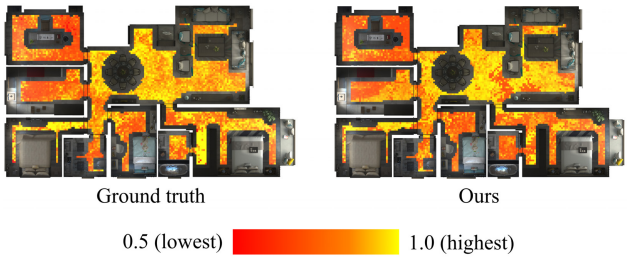


Fig. 9. *Scene 1*'s SP score map generated by our SP predictor.

- The *Delete* action is dimensionality-changing with acceptance probability:

$$A_{\mathrm{d}}(\mathcal{G}^*|\mathcal{G}) = \min(1, \frac{f(\mathcal{G}^*)}{f(\mathcal{G})} \frac{\mathbb{P}_{\mathrm{a}}}{\mathbb{P}_{\mathrm{d}}} \frac{|\mathcal{V}|}{|\mathcal{P} - \mathcal{V}^*|}). \tag{19}$$

- The *Move* action is not dimensionality-changing with acceptance probability:

$$A_{\mathrm{m}}(\mathcal{G}^*|\mathcal{G}) = \min(1, \frac{f(\mathcal{G}^*)}{f(\mathcal{G})}). \tag{20}$$

6 RESULTS

We show prediction results of the SP predictor and teleport graph synthesis results of the overall optimization approach. We discuss the results for the four scenes shown in Figure 10, which are used in the user study, in the main paper. The supplementary material contains the results for the other 12 scenes. Moreover, we show that our approach could be extended to handle different special scenarios with exploratory experiments.

6.1 Implementation

We conducted our experiments on a machine equipped with an Intel Core i7-9700 CPU and an NVIDIA GeForce RTX 2070 GPU. The SP predictor was implemented in Python using Pytorch. Training the model took about one day. In the inference stage, it generally takes the SP predictor around 20−40 seconds to generate the SP score map for typical indoor apartment scenes with 2k−4k positions in the free space. The teleport graph optimization module was implemented in the Unity3D game engine. A single teleport graph synthesis takes about 5,000 iterations and 2-3 minutes based on our implementation.

6.2 Scene Perception Score Prediction

We first evaluated our GAT-based SP predictor's ability to predict human preferences of positions in indoor scenes. For comparison, we built a baseline CNN model that directly took panoramic RGB-D images rather than SP graphs as input. We directly used the backbone of our trained object detector as the RGB feature extraction model for the baseline, thus we expect the baseline to be object-aware as well. In addition, we used a Resnet-18 network to process the depth channel. The extracted RGB and depth features were concatenated and fed into an MLP to produce an SP score.

Both the SP predictor and the baseline were trained on the SP dataset described in Section 4.3. The SP predictor attained a root mean squared error (RMSE) of 0.0591 on the test set, and the baseline model achieved an RMSE of 0.0725.

Table 1. Quantitative results of SP score prediction for our SP predictor (GAT) and the baseline (CNN) in the four experiment scenes. For AE p-value of *Ours/Baseline*, bold indicate significant differences with 95% confidence.

Scene	Pos.#	RMSE		AE Variance		AE
		Ours	Baseline	Ours	Baseline	p-value
1	4,766	**0.057**	0.069	**0.0014**	0.0018	6.83e-1
2	2,781	**0.073**	0.085	**0.0019**	0.0027	**6.35e-3**
3	4,639	**0.049**	0.071	**0.0010**	0.0023	**9.92e-66**
4	5,454	**0.072**	0.112	**0.0038**	0.0056	**1.10e-55**

Table 2. Statistics of our synthesized results in the four experiment scenes. For each teleport graph, we calculate the percentage of covered area, and the average SP score of selected positions versus that of candidate positions.

Scene	Selected Pos.#	Covered	Selected Pos. SP score	Candidate Pos. SP score
1	19	67.13%	0.92	0.84
2	14	79.77%	0.94	0.80
3	19	68.15%	0.88	0.78
4	23	71.74%	0.91	0.82

While the SP dataset consists of panoramic RGB-D images synthesized using real-world images, we further evaluated the predictor's performance with general virtual environments in the 16-scene dataset described in Section 4.3. We show quantitative results, including RMSE, absolute error (AE) variance, and t-test results of *Ours/Baseline* considering AE, for *Scene 1* to *Scene 4* in Table 1. Figure 9 visualizes an example SP score map generated for *Scene 1* by our predictor. Refer to the supplementary material for comprehensive results for all scenes. The results suggest that our SP predictor produced more accurate results than the baseline method did.

6.3 Teleport Graph Synthesis for Indoor Scenes

We synthesize teleport graphs using our approach for the four indoor scenes. Table 2 shows some result statistics. Figure 10 shows the results and the panoramic views at some sampled positions. We also show qualitative comparisons between synthesized graphs with and without the SP score consideration in Section 3.3 in the supplementary material.

In general, the synthesized teleport graphs are shaped well by the cost function. First, for desirable navigation, each node is associated with a position of a relatively high SP score in its neighborhood area, which is vital as our approach is supposed to suggest desirable positions to users. Second, for complete navigation, the synthesized teleport graphs are connected, which means that a path exists between any arbitrary pair of nodes. Third, free space and all rooms in the virtual scenes are well covered by the teleport graphs so that users would not miss any area. Fourth, for efficacious navigation, the number of positions is regularized such that users could perceive rich visual information by traversing sparse teleport graphs.

6.4 Exploratory Experiments

Our approach can be extended to incorporate other considerations in synthesizing teleport graphs. To demonstrate the generalizability of our approach, we conducted exploratory experiments where we modified our original approach to handle some special cases.

Position Density. In our approach, an approximate number of positions (nodes) for a teleport graph synthesis is predicted by $\lambda(|\mathcal{P}|)$. In

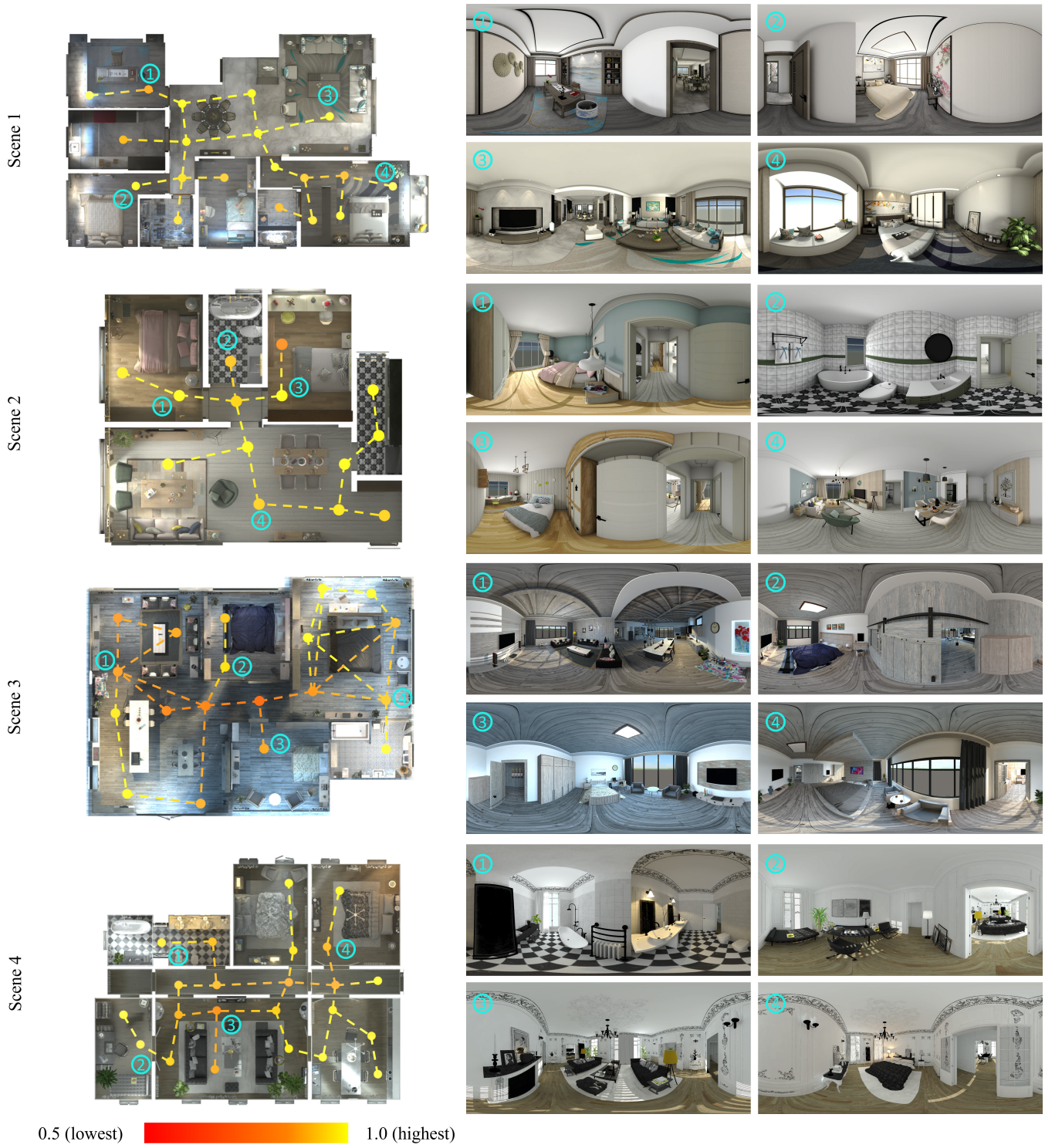


Fig. 10. Teleport graphs synthesized for different scenes. Each node is colored according to the SP score at its position. For each scene, the views at four positions are visualized as panoramic images.

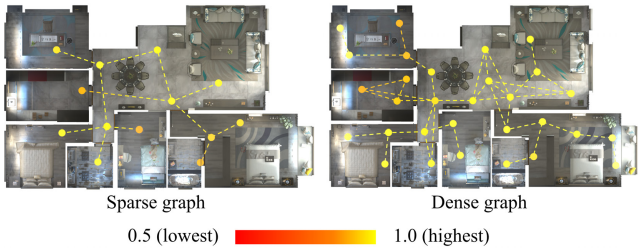


Fig. 11. Using different target numbers of nodes for teleport graph synthesis.

practice, users might prefer to specify a target number of positions. For example, some users may prefer to traverse a scene quickly by visiting some key positions only, while other users may prefer to navigate the scene in smaller steps by visiting more positions.

We allow such flexibility by replacing $\lambda(|\mathcal{P}|)$ with a user-specified target number of positions. First, we calculate a standard number of positions by dividing the total area of free space of the scene by a node's neighborhood area. This number refers to the number of positions needed to cover the whole free space in the ideal case where all neighborhood regions do not overlap. Then we multiply it with a density value to obtain a target number of positions.

Figure 11 shows two examples for *Scene 1*. The standard number of positions of *Scene 1* is 22. We assign a low density value of 0.6 to obtain a target number of positions of 13 for synthesizing a sparse graph. Similarly, we assign a high density value of 1.4 to obtain a target number of positions of 30 for synthesizing a dense graph. Note that both graphs cover the whole scene. We believe this flexibility of controlling the target number of positions could provide a handy navigation setting option for VR applications such as games.

Layout Change. There are scenarios where the furniture layout of a virtual scene may change. For example, in an interactive VR application such as a game, the user may move a furniture object. On the other hand, when a designer models a virtual scene in a game engine or an interior design application, the designer may move the furniture objects. Our approach supports re-computation of the teleport graph to cope with furniture object movements.

Suppose an initial teleport graph $\mathcal{G}$ has been synthesized for a scene. If a furniture object is moved to a nearby position, our approach updates the SP score map and the teleport graph as follows.

(1) Update the free space by removing candidate positions that are now occupied by the moved furniture object from $\mathcal{P}$; and by adding candidate positions in the region previously occupied by the furniture object to $\mathcal{P}$.
(2) Removing the selected positions which are now occupied by the moved furniture object from the node set $\mathcal{V}$ of graph $\mathcal{G}$.
(3) Update the SP scores of positions from which the furniture object can be seen before or after the furniture movement.
(4) Re-trigger the teleport graph optimization process starting from $\mathcal{G}$; use a low temperature to fine-tune the graph to produce another graph $\mathcal{G}^*$ that fits with the changed layout.

Figure 12 shows two examples of moving a furniture object in *Scene 1* and *Scene 2*. In each case, the teleport graph is updated to cope with the furniture object movement (an update took around 30 seconds). The graph shows bigger adjustments in the region where

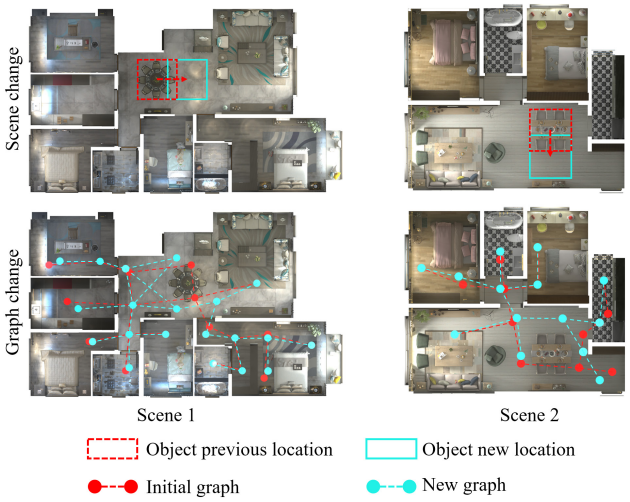


Fig. 12. Teleport graph fine-tuning induced by a change in the layout.

the object movement occurs, whereas the selected positions in the other regions are slightly adjusted. Note that, if desired, the designer could fix the selected positions far away from the moved furniture object using hard constraints, or using soft constraints by adding a regularization cost term to prompt the positions to stay constant.

Other Environments. While we focus on synthesizing teleport graphs for indoor scenes, our approach could be extended to handle other environments as long as the desirability of the teleport positions in these environments could be evaluated. The evaluation could be performed similarly by a data-driven approach with suitable datasets or by heuristic principles set by a designer.

As an example, we tested this idea for a natural park environment as shown in Figure 13. For such a large outdoor environment, teleportation is vital to efficacious navigation in virtual reality. To facilitate wayfinding, we want users to be able to see the landmarks (i.e. flags) when they navigate the park so that they are aware of their locations. We use our approach to synthesize a teleport graph for this park by making two changes, as follows.

- *Score Map Generation:* We redefine the SP score of a teleport position based on the visibility of a flag within a certain distance (i.e. 100m) from that position. If a user standing at a position can see a flag within the distance, the position has an SP score of 1. Otherwise, the position has an SP score of 0.
- *Cost Terms:* Unlike indoor scenes which are divided into rooms, a natural park is not divided into distinct regions. Therefore we turn off the room coverage cost term by setting $w_{\text{RC}} = 0$. Akin to the number of positions preference experiment, for the regularization cost term C_{REG}, we set the target number of positions by using a density value of 20.0 instead of using the polynomial function $\lambda(\cdot)$.

Figure 13 shows synthesized results. A majority (81.48%) of the teleport positions have a visible flag in their proximity. While the graph covers the whole scene well, it also maintains connectivity. Note that some positions where no flag is in sight are still selected to satisfy the coverage and connectivity considerations. For example, some positions are added between the pink flag and the brown flag

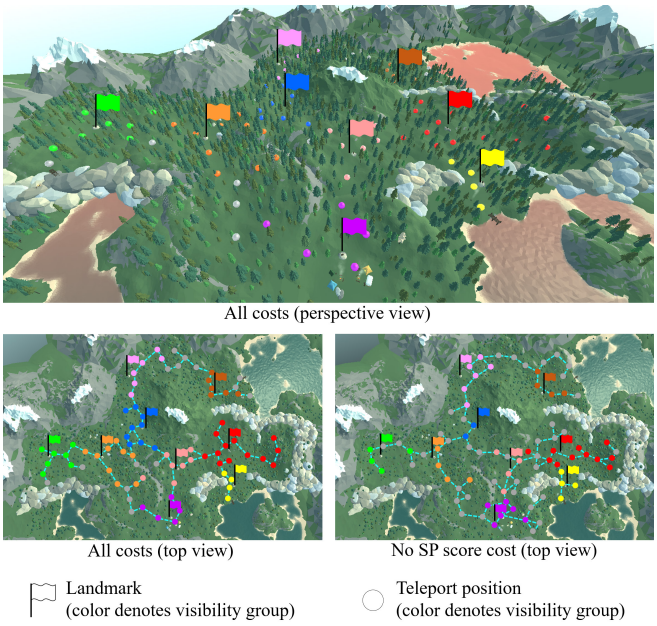


Fig. 13. Teleport graphs synthesized for a natural park. Gray nodes denote positions where no flag is visible. For the graph synthesized with all costs, 81.48% of the positions have a visible flag in their proximity. For the graph synthesized without the SP score cost, only 54.22% of the positions have a visible flag. No position is sampled from the inaccessible mountains.

to enhance connectivity even though no flag is visible at these positions. For comparison, we synthesize another teleport graph with all costs except the SP score cost. Figure 13 shows the synthesized graph, where only 54.22% of the teleport positions have a visible flag nearby. The supplementary material contains the results for two other environments: a canyon scene with the same SP score definition as the natural park example and a multi-floor outlet scene with a different SP score definition.

Fixing User-Specified Positions. In certain situations, a designer or a user might want to visit particular positions such as some places of interest in a virtual scene. To achieve this goal, we allow a designer to define fixed nodes for a teleport graph and then run our approach. By not applying any action to these nodes in the teleport graph sampling and optimization process, the fixed nodes are kept unchanged. Figure 14 shows two examples on *Scene 1* and *Scene 2*. The green nodes, which correspond to some places of interest, are assigned and fixed by the designer. For each scene, two teleport graphs are synthesized. It can be observed that the fixed nodes are guaranteed to be included in the generation process, while other nodes and edges might vary. Therefore, it is possible to synthesize different teleport graphs that share these fixed nodes. This feature is useful for extending our approach to consider interactive VR scenarios: Designers may specify positions where interactive objects are accessible and synthesize teleport graphs based on such constraints.

7 USER STUDY

We conducted a user study to validate if our synthesized teleport graph could improve teleportation efficiency in comparison to a conventional approach that requires users to manually determine

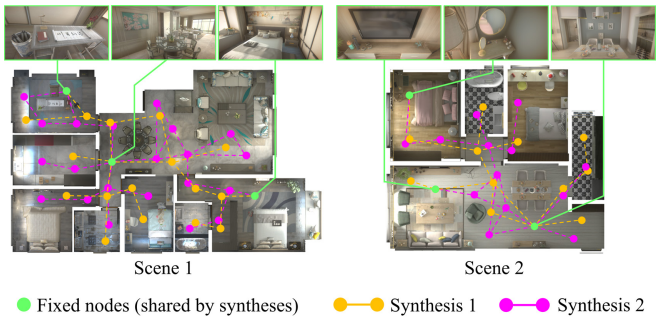


Fig. 14. Fixing user-defined positions in teleport graphs. Green nodes denote the fixed positions, which are shared by all syntheses in the same scene.

teleport positions. By analyzing virtual environments, predicting human preferences, and then synthesizing teleport graphs, we expect that our approach could make up for the deficiencies that users have no prior knowledge about unknown environments, thus allowing users to navigate an environment more efficaciously in VR. We also examined if our SP predictor is able to produce reasonable predictions of desirable positions.

7.1 Settings

In this user study, we compare navigating virtual environments using a teleport graph synthesized by our approach (*ours*) with the conventional "point & teleport" (*P&T*) approach [Bozgeyikli et al. 2016]. For *ours*, the suggested positions of synthesized teleport graphs are visualized as glowing anchors in the environment, which the participant can point at and teleport to. Note that while the edges of teleport graphs encode constraints such as connectivity, they are not visualized during VR navigation. For *P&T*, there is no synthesized teleport graph and no teleport anchor. The participant chooses their own teleport positions. In our experiments, we applied the Dash [Bhandari et al. 2018] technique to gradually change the viewpoint when teleporting to reduce spatial disorientation and enhance user experience.

We recruited 32 participants to navigate in the four virtual apartment scenes shown in Figure 9 using an Oculus Quest 2, an all-in-one VR system consisting of a VR headset and two controllers. All participants were undergraduate or graduate students, including 11 females and 21 males aged 19 through 28. Since we compare two approaches in four scenes, there are eight possible conditions in total. A participant used either of the two approaches to explore each scene. Hence there are 128 navigation records in total. Through a pseudorandom procedure, we ensured that (1) one participant used *ours* in two scenes and used *P&T* in the other two scenes; (2) the occurrences of all conditions were counterbalanced such that there are 16 navigation records for each condition.

7.2 Procedure

Our user study procedure was approved by the Institutional Review Board. One round of study comprises three stages as follows.

Warm Up. Before navigation experiments, we prepared a warm up scene where participants got familiar with device operations, and how to use both *ours* and *P&T* to navigate the scene. Participants could stay in the warm up session as long as they wanted.

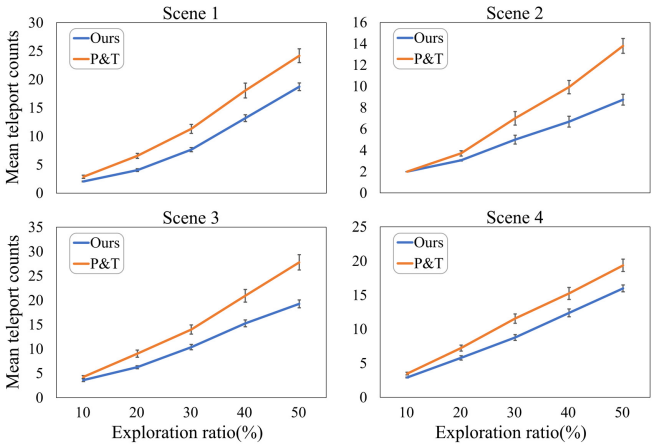


Fig. 15. Mean teleport counts versus exploration ratio. Error bars show standard errors. With the help of the anchor positions of the teleport graphs (*ours*), the participants generally took fewer teleports to explore a scene.

Navigation. This is the main stage of the user study. The participants navigated in the four virtual scenes for given conditions. The goal was to investigate whether *ours* could lead to more efficacious VR navigation. While we asked them to freely explore the scenes and to observe the context (structure, objects, furniture, etc.) of the environments, there was no mandatory requirement or goal that they needed to achieve. For each condition, a participant stayed in the assigned scene for two minutes and was transferred to the next scene afterwards. Note that participants could still manually select and teleport to any desired position even when they were in conditions with *ours* where anchor positions were presented.

Desirability Evaluation. The purpose of this stage is to validate whether the suggested positions given by our SP predictor matched human preferences. This stage was conducted in a two-alternative forced choice (2AFC) manner. The procedure is similar to the pilot study in Section 4.3: participants visit a pair of virtual positions in an immersive manner through VR devices, and they were asked to choose the position with a more desirable view based on the surrounding objects (e.g., furniture, decoration).

We picked the two scenes which a participant visited using *P&T* in the navigation stage, and the participant was transferred to three pairs of positions in each scene. In a comparison pair, one position was randomly selected from the participant's manual choices (*manual*), and the other was randomly selected from the anchor positions (*anchor*) of the teleport graph for that scene under *ours*.

7.3 Results

Navigation. We recorded participants' trajectories during the experiment procedure. We define two metrics: (1) *exploration ratio*, which is the percentage of scene area covered by all visited positions' neighborhood areas; and (2) *anchor choice ratio*, which is the percentage of counts an anchor position is chosen in all teleports. The anchor choice ratio is used for the conditions with *ours*. In our analysis, we found that although participants' exploration progress varied given the two-minute time limit, most of them explored more than 50% of a scene under either *ours* or *P&T*, with a mean of 71.44% and a standard deviation of 16.32%. We obtained a mean anchor choice ratio of 66.95% for conditions with *ours*, with a standard deviation of 21.40%. We show the distribution of participants' anchor choice ratios in the supplementary material. In our analysis, we discard 5 statistical outliers out of 128 navigation records considering the exploration ratio or the anchor choice ratio. There are 14 to 16 pieces of data remaining for each condition.

Table 3. Participants' choices for *anchor* results and *manual* results. Chi-square analysis results are shown in p-values (bold indicates *anchor* results are preferred with a 95% confidence level). Bayesian analysis results are shown in odds O_{10} on the alternative hypothesis H_1 over the null hypothesis H_0 (bold indicates *anchor* results are preferred with evidence).

Scene	*Anchor#* vs. *Manual#*	p-value	O_{10}
1	34 : 14	**0.0039**	**11.91**
2	31 : 17	**0.0433**	1.35
3	33 : 15	**0.0094**	**5.25**
4	29 : 19	0.1489	0.50

We did not evaluate participants' performances using the time they spent to reach specific exploration ratios due to varying navigation habits (i.e., some participants perceived the scenes more carefully and explored slower, and some others navigated faster). Instead, we counted the number of teleports performed when participants' exploration ratio reached 10%, 20%, 30%, 40%, and 50%, respectively. Data was truncated at the 50% exploration ratio in our analysis because 50% is a rough bar that all remaining data reached (i.e. some records could not reach higher ratios). If people perform fewer teleports using one approach than the other to reach the same exploration ratio in a scene, it would suggest that this approach is more efficacious. We average our collected data in each of the eight conditions and present the results in Figure 15. In all of the four scenes, participants performed fewer teleports using *ours* than using *P&T*, thus attained improvement in VR teleportation efficiency. We show boxplots for all conditions in our supplementary material. We performed a t-test to examine the differences in teleport counts at the exploration ratio of 50% using the two approaches for each scene. Significant differences (p = 4.57e−4, 3.50e−6, 3.34e−5, 2.33e−3, respectively) were found. We also measured the effect size by calculating the Cohen's d, and found very large effect sizes (d = 1.67, 2.20, 1.87, 1.28, respectively) according to [Sawilowsky 2009].

An interesting observation is that in conditions where *P&T* was used, to move to a different region, a participant usually triggered a sequence of intermediate teleports along the trajectory and finally stayed at a destination. In contrast, under *ours*, participants usually directly jumped to anchor positions without intermediate teleports. This might explain why people performed more teleports using *P&T*. By predicting and suggesting desirable positions, our approach could reduce the number of participants' teleports.

Desirability Evaluation. We collected 48 pairs of participants' forced choices for each scene. Our null hypothesis H_0 was that users perceive no significant difference in the desirability of *anchor* and *manual*, and the alternative hypothesis H_1 was that users did perceive significant differences. We first adopted the Chi-square nonparametric analysis technique. The obtained optimal/random frequency in each scene were compared to an expected 24/24 result to ascertain whether this difference is significant. Second, we

adopted a Bayesian analysis to determine whether the number of participants who selected *anchor* was what would be expected by chance, or if there was a preference pattern. We assumed that the participant had a probability P of picking *anchor*, and used a binomial distribution to model the results. We computed the odds O_{10} on H_1 over H_0. According to [Rouder et al. 2009], $O_{10} > 3$ shows "some evidence", whereas $O_{10} > 10$ shows "strong evidence", favoring H_1.

Numerical results are shown in Table 3. Generally, more people prefer to choose *anchor* than *manual*. For the Chi-square analysis results, the p values are 3.89e−3, 4.33e−2, 9.37e−3 and 1.49e−1 for the 4 scenes respectively, and significant differences were found in *Scene 1*, *Scene 2* and *Scene 3*. For the Bayesian analysis results, the odds O_{10} are 11.91, 1.35, 5.25 and 0.50 in the 4 scenes respectively, and evidence favoring H_1 was found in *Scene 1* and *Scene 3*.

The results might indicate three points: (1) Using *P&T*, participants occasionally chose teleport positions that they did not favor. Consistent with our observation, the reason could be the lack of prior knowledge about the views at the positions before teleporting; (2) Using *ours*, participants were generally able to move to desirable positions in the teleport procedure, suggesting that our SP predictor is capable of predicting the desirability of positions; (3) Preferences towards the positions suggested by *ours* are more prominent in scenes with a larger open space. In contrast, in scenes with a small open space, the *anchor* positions and *manual* positions tend to be close to each other. For example, *Scene 4* consists of a narrow corridor that connects small rooms. As a result, the teleport positions, whether suggested by *anchor* or selected by *manual*, have similar panoramic views. This may explain why the preference difference is not significant as indicated by a small odd $O_{10} = 0.50$.

8 CONCLUSION

We presented scene-aware VR teleport graphs aiming to improve virtual navigation. Our approach predicts desirable positions in virtual environments via a graph convolutional model and synthesizes teleport graphs via an optimization. Experiments show that our SP predictor is reliable, and the synthesized teleport graphs satisfy a set of VR teleportation constraints. Through exploratory experiments, we also demonstrated the generalizability of our approach which allows users to control the synthesis for different practical scenarios. Our user study suggests that our teleport graphs help users navigate virtual environments more efficaciously, and our SP predictor suggests desirable positions that match human preferences.

8.1 Limitations and Future Work

As our SP predictor is trained to work on indoor scenarios, it cannot predict the desirability of outdoor positions. For this reason, in our implementation, we also discarded sampled positions in balconies with outdoor views. In the future, if large-scale panorama dataset of outdoor scenes (e.g., campus, parks, downtowns) becomes available, it would be interesting to train an SP prediction model for predicting the desirability of teleport positions for outdoor environments. To circumvent this current limitation to deal with general environments, we propose a substitute of the SP prediction model by replacing the SP score with other user-defined scores based on heuristics as discussed in Section 6.4.

In our work, we focus on navigation for visually perceiving the environments. In some different navigation scenarios which prioritize other considerations (e.g., interaction or sound) rather than visual perception, the usage of predicted SP score could be limited. For example, if some positions associated with interactive events are predicted with low SP scores, the predicted perception would hinder incorporating those positions into teleport graphs. Variations of our current constraints or new considerations could be applied to handle specific cases.

We validated that users could (1) explore virtual scenes with fewer teleport actions, and (2) obtain more visually-desirable views using our method compared to conventional teleportation in Section 7. We also provided qualitative analysis on comparisons of synthesized teleport graphs with and without the SP score cost in the supplementary material. In future work, it would be interesting to perform a perceptual study on how the SP consideration may influence users' navigation behaviors. For example, one may investigate users' navigation performances using teleport graphs synthesized with and without SP considerations.

We are also interested in more in-depth user studies to investigate how our approach could help VR navigation. For example, we can define a set of clearly defined tasks that need to be completed via teleportation, and analyze users' behaviors with some task performance metrics and subjective usability metrics. Moreover, it is worth studying how participants' familiarity with VR could influence their teleportation behaviors.

We used the SP graphs to model spatial relationships between a user and the surrounding objects. Beside facilitating teleportation, the SP graphs might find other interesting applications too, such as for positioning virtual agents and driving them to navigate naturally in virtual environments based on their visual observations of the surrounding objects.

We formulate an optimization-based framework for synthesizing teleport graphs, which is extensible and amenable to user control by adjusting the optimization parameters. In case we can obtain an enormous amount of teleportation data from many users in virtual reality in the future (e.g., from VR social platforms), it would be worth investigating how to train an end-to-end data-driven model such as a deep generative model of graphs [Li et al. 2018; Wang et al. 2019] to synthesize teleportation graphs.

Another interesting direction for future work is to extend our approach to accommodate users with different mobility restrictions. For example, for wheelchair users, it would be reasonable to use a smaller radius r for neighborhood regions, since it could be challenging for those users to freely move around a teleport position.

Navigation habits vary among users. For example, some users may like to contemplate decorations, whereas others may like to traverse an environment quickly. Future works might consider learning user navigation habits to synthesize personalized teleport graphs.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their constructive comments. We thank Amilcar Gomez Samayoa and Javier Talavera for their help with the user study. This project was supported by an NSF CAREER Award (award number: 1942531).

REFERENCES

Majed Al Zayer, Paul MacNeilage, and Eelke Folmer. 2018. Virtual locomotion: a survey. *IEEE transactions on visualization and computer graphics* (2018).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

Jiwan Bhandari, Paul R MacNeilage, and Eelke Folmer. 2018. Teleportation without Spatial Disorientation Using Optical Flow Cues.. In *Graphics interface*. 162–167.

Costas Boletsis. 2017. The new era of virtual reality locomotion: A systematic literature review of techniques and a proposed typology. *Multimodal Technologies and Interaction* 1, 4 (2017), 24.

Evren Bozgeyikli, Andrew Raij, Srinivas Katkoori, and Rajiv Dubey. 2016. Point & teleport locomotion technique for virtual reality. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*. ACM, 205–216.

Fabio Buttussi and Luca Chittaro. 2019. Locomotion in Place in Virtual Reality: A Comparative Evaluation of Joystick, Teleport, and Leaning. *IEEE transactions on visualization and computer graphics* (2019).

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3D: Learning from RGB-D Data in Indoor Environments. *International Conference on 3D Vision (3DV)* (2017).

Shih-Han Chou, Cheng Sun, Wen-Yen Chang, Wan-Ting Hsu, Min Sun, and Jianlong Fu. 2020. 360-Indoor: Towards Learning Real-World Objects in 360deg Indoor Equirectangular Images. In *The IEEE Winter Conference on Applications of Computer Vision*. 845–853.

Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. 2018. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *Proceedings of the European Conference on Computer Vision*. 518–533.

Zhi-Chao Dong, Xiao-Ming Fu, Zeshi Yang, and Ligang Liu. 2019. Redirected smooth mappings for multiuser real walking in virtual reality. *ACM Transactions on Graphics (TOG)* 38, 5 (2019), 1–17.

Li Fei-Fei. 2007. Recognizing and learning object categories. *CVPR Short Course, 2007* (2007).

Tian Feng, Lap-Fai Yu, Sai-Kit Yeung, KangKang Yin, and Kun Zhou. 2016. Crowd-driven mid-scale layout design. *ACM Trans. Graph.* 35, 4 (2016), 132–1.

Sebastian Freitag, Benjamin Weyers, and Torsten W Kuhlen. 2016. Automatic speed adjustment for travel through immersive virtual environments based on viewpoint quality. In *2016 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 67–70.

Sebastian Freitag, Benjamin Weyers, and Torsten W Kuhlen. 2017. Efficient approximate computation of scene visibility based on navigation meshes and applications for navigation and scene analysis. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 134–143.

Sebastian Freitag, Benjamin Weyers, and Torsten W Kuhlen. 2018. Interactive exploration assistance for immersive virtual environments based on object visibility and viewpoint quality. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 355–362.

Markus Funk, Florian Müller, Marco Fendrich, Megan Shene, Moritz Kolvenbach, Niclas Dobbertin, Sebastian Günther, and Max Mühlhäuser. 2019. Assessing the Accuracy of Point & Teleport Locomotion with Orientation Indication for Virtual Reality using Curved Trajectories. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. 2003. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*. Springer, 129–143.

Wilson S Geisler. 2008. Visual perception and the statistical properties of natural scenes. *Annu. Rev. Psychol.* 59 (2008), 167–192.

Peter J Green. 1995. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* 82, 4 (1995), 711–732.

MP Jacob Habgood, David Moore, David Wilson, and Sergio Alapont. 2018. Rapid, continuous movement between nodes as an accessible virtual reality locomotion technique. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 371–378.

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.

Peter Hedström and Charlotta Stern. 2008. Rational choice and sociology. *The new Palgrave dictionary of economics* 2 (2008).

Dichao Hu. 2019. An introductory survey on attention mechanisms in NLP problems. In *Proceedings of SAI Intelligent Systems Conference*. Springer, 432–448.

Laurent Itti, Christof Koch, and Ernst Niebur. 1998. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence* 20, 11 (1998), 1254–1259.

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

Eike Langbehn, Paul Lubos, and Frank Steinicke. 2018. Evaluation of locomotion techniques for room-scale vr: Joystick, teleportation, and redirected walking. In *Proceedings of the Virtual Reality International Conference-Laval Virtual*. 1–9.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. 2018. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324* (2018).

Yong-Lu Li, Siyuan Zhou, Xijie Huang, Liang Xu, Ze Ma, Hao-Shu Fang, Yanfeng Wang, and Cewu Lu. 2019. Transferable interactiveness knowledge for human-object interaction detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3585–3594.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2117–2125.

Imran Mahalil, Azmi Mohd Yusof, Nazrita Ibrahim, Eze Manzura Mohd Mahidin, and Mohd Ezanee Rusli. 2019. Virtual Reality Mini Map Presentation Techniques: Lessons and experience learned. In *2019 IEEE Conference on Graphics and Media (GAME)*. IEEE, 26–31.

Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. 2018a. Learning human-object interactions by graph parsing neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 401–417.

Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. 2018b. Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5899–5908.

Sharif Razzaque, Zachariah Kohn, and Mary C. Whitton. 2001. Redirected Walking. In *Eurographics 2001 - Short Presentations*. Eurographics Association. https://doi.org/10.2312/egs.20011036

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.

Ronald A Rensink. 2000. Scene perception. *Encyclopedia of psychology* 7 (2000), 151–155.

Jeffrey N Rouder, Paul L Speckman, Dongchu Sun, Richard D Morey, and Geoffrey Iverson. 2009. Bayesian t tests for accepting and rejecting the null hypothesis. *Psychonomic bulletin & review* 16, 2 (2009), 225–237.

Shlomo S Sawilowsky. 2009. New effect size rules of thumb. *Journal of modern applied statistical methods* 8, 2 (2009), 26.

Ehsan Sayyad, Misha Sra, and Tobias Höllerer. 2020. Walking and Teleportation in Wide-area Virtual Reality Experiences. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 608–617.

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12, 9 (2011).

Mel Slater, Martin Usoh, and Anthony Steed. 1995. Taking steps: the influence of a walking technique on presence in virtual reality. *ACM Transactions on Computer-Human Interaction (TOCHI)* 2, 3 (1995), 201–219.

Richard Stoakley, Matthew J Conway, and Randy Pausch. 1995. Virtual reality on a WIM: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 265–272.

Qi Sun, Anjul Patney, Li-Yi Wei, Omer Shapira, Jingwan Lu, Paul Asente, Suwen Zhu, Morgan Mcguire, David Luebke, and Arie Kaufman. 2018. Towards virtual reality infinite walking: dynamic saccadic redirection. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 67.

Oytun Ulutan, ASM Iftekhar, and Bangalore S Manjunath. 2020. VSGNet: Spatial Attention Network for Detecting Human Object Interactions Using Graph Convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13617–13626.

Martin Usoh, Kevin Arthur, Mary C Whitton, Rui Bastos, Anthony Steed, Mel Slater, and Frederick P Brooks Jr. 1999. Walking> walking-in-place> flying, in virtual environments. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 359–364.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *The Journal of Machine Learning Research* 11 (2010), 1201–1242.

Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. 2019. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–15.

Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. 2018. Gibson env: real-world perception for embodied agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. 2019. Graph convolutional networks: a comprehensive review. *Computational Social Networks* 6, 1 (2019), 11.

Yin Zhang, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics* 1, 1-4 (2010), 43–52.

Yixin Zhu, Chenfanfu Jiang, Yibiao Zhao, Demetri Terzopoulos, and Song-Chun Zhu. 2016. Inferring forces and learning human utilities from videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3823–3833.