Optimized Data Representation for Interactive Multiview Navigation

Rui Ma, Thomas Maugey, Pascal Frossard

Optimized Data Representation for Interactive Multiview Navigation

Rui Ma, *Student Member, IEEE,* Thomas Maugey, *Member, IEEE,* and Pascal Frossard, *Senior Member, IEEE*

***Abstract*—In contrary to traditional media streaming services where a unique media content is delivered to different users, interactive multiview navigation applications enable users to choose their own viewpoints and freely navigate in a 3-D scene. The interactivity brings new challenges in addition to the classical rate-distortion trade-off, which considers only the compression performance and viewing quality. On the one hand, interactivity necessitates sufficient viewpoints for richer navigation; on the other hand, it requires to provide low bandwidth and delay costs for smooth navigation during view transitions. In this paper, we formally describe the novel trade-offs posed by the navigation interactivity and classical rate-distortion criterion. Based on an original formulation, we look for the optimal design of the data representation by introducing novel rate and distortion models and practical solving algorithms. Experiments show that the proposed data representation method outperforms the baseline solution by providing lower resource consumptions and higher visual quality in all navigation configurations, which certainly confirms the potential of the proposed data representation in practical interactive navigation systems.**

***Index Terms*—Multiview navigation, interactivity, navigation segment, multiview image compression**

I. Introduction

With the development of multiview imaging techniques, there has been a lot of interest in interactive multiview navigation [1], [2]. Differently from traditional media streaming systems where a unique media content is streamed to all users, interactive multiview navigation systems provide users with different media data depending on their interactions with the server. In particular, each user watches a specific 2-D image corresponding to his own choice of viewing position and orientation (called a *viewpoint*) and is able to navigate in the scene by freely changing this viewpoint (see Fig. 1). These virtual views are synthesized from the content of different cameras positioned in the 3-D scene.

In order to achieve interactive navigation, it is necessary to consider a complete processing chain consisting of different connected components, including data representation, coding, transmission and view rendering. Indeed, the consideration of every component in isolation can only lead to suboptimal performance. In the literature, the individual components of the navigation system have been extensively studied, e.g., 3-D scene representation [3], [4], multiview video compression [5], [6], multiview data streaming [7], [8] and view synthesis [9], [10]. However, there is clearly a lack of fully integrated frameworks that incorporate these techniques in an end-to-end system and jointly optimize them.

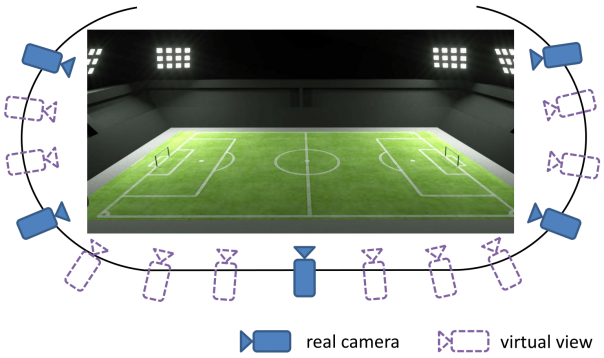


Fig. 1. A typical multiview navigation scenario. Users are able to freely navigate along virtual views that are synthesized from views captured at different camera locations.

The end-to-end system optimization involves complex design trade-offs. As *interactivity* denotes the users' flexibility to choose arbitrary viewpoints during navigation, it first requires a sufficiently large navigation range, i.e., a large set of achievable viewpoints. Second, as users are willing to watch only a subpart of this navigation range, the system must transmit *only what is useful* in order to limit bandwidth usage in practice. This original trade-off between bandwidth limitations, visual quality and navigation flexibility has not been solved in the literature. Indeed, the good compression performance of traditional multiview schemes [5], [6] is obtained at a price of possibly long coding prediction paths, which however prevent independent view decoding (an analogue problem is posed by random access in monoview video [11]). A careful redesign of the whole system is thus needed, starting from the representation of the multiview data itself. In that spirit, the work in [12] has proposed to organize the achievable viewpoints in independently decodable partitions, namely the *navigation segments*. Indeed, the navigation segment can be regarded as the spatial analogue to the temporal GOP (group of pictures) in monoview video transmissions. This approach has however left some important questions opened, such as the optimal design of the navigation segments.

In this paper, we formally describe the novel trade-offs posed by interactive schemes between bandwidth limitations, visual quality and navigation interactivity, in the context of the *navigation segment representation*. Based on this original formulation, we propose to study *the optimal design of the navigation segments*. We further take into account the delay

R. Ma is with the Department of Electrical and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: rmaaa@connect.ust.hk).

T. Maugey is with Inria Rennes-Bretagne Atlantique Research Centre, Rennes Cedex 35042, France (e-mail: thomas.maugey@inria.fr).

P. Frossard is with Signal Processing Laboratory LTS4, École Polytechnique Fédérale de Lausanne CH-1015, Switzerland (e-mail: pascal.frossard@epfl.ch).

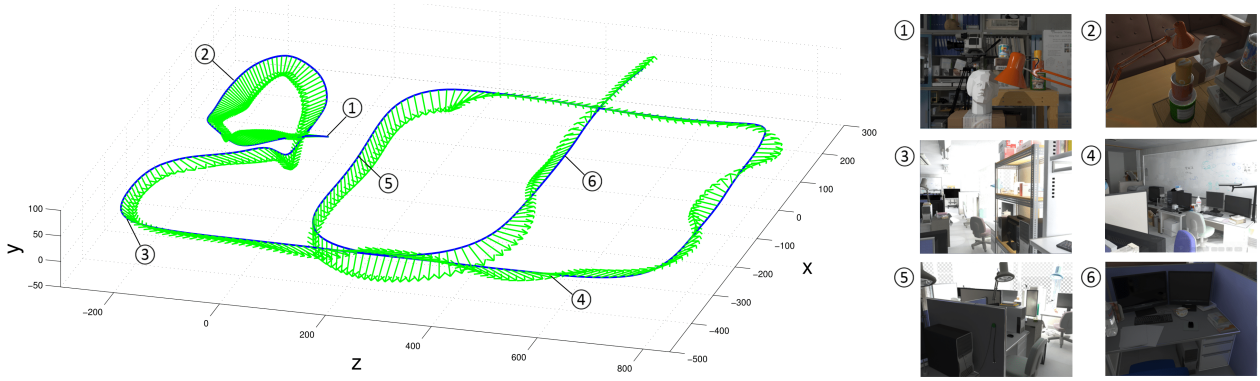


Fig. 2. Illustration of 1-D manifold camera arrangement using the New Tsukuba Stereo Dataset [13], [14]. The solid line denotes the camera trajectory while the arrows point out the orientations. We also show some camera views at different locations.

between user requests and actual data receiving, by introducing the concept of *navigation ball*. We conduct our study in the challenging scenario of wide navigation range like the 1-D manifold camera arrangement depicted in Fig. 2. Experiments on the New Tsukuba Dataset [13], [14] show that the proposed navigation segment representation outperforms the baseline equidistant solution (where navigation segments are equally divided). Our approach offers lower resource consumptions and higher visual quality in all different navigation configurations, due to its high adaptability to various navigation parameters, like the navigation speed and the view popularity.

The rest of the paper is organized as follows. Section II introduces the related work. Section III describes the proposed navigation segment representation under the navigation environment. Section IV proposes the optimization framework for the navigation system, and Section V further elaborates on the problem formulation with novel rate and distortion models. Section VI investigates practical solutions and analyses their complexity. Experimental results are demonstrated in Section VII, and Section VIII draws the conclusions.

II. Related Work

The problem of interactive multiview navigation [1] has recently gained interest in the research community. A first category of works provides navigation interactivity by switching views between a predefined set of real camera viewpoints. The H.264 SP/SI-frames [15], for example, is able to increase the interactivity between view switchings by avoiding transmission of previous frames in the new view. A SP-frame can be inserted at view switching point, which is able to be identically decoded from a cross-view reference instead of a reference in the same view [5]. The distributed source coding (DSC) can also be utilized for interactive streaming [16], [17], since a DSC frame can be identically reconstructed from different predictors. Another way to increase the interactivity is to produce multiple decoded versions of the media subset. In [18], [19], redundant P-frames are used to support multiple decoding process. In [20], the multiple encoding versions are stored in the server for diverse user requests. These methods, however, require large server storage. The interactive navigation also needs to consider the user behaviors. In [21], [22], for example, the prediction structure is adapted to the user position estimated by Kalman filtering. Although the above methods can increase the interactivity, it is limited to actual camera viewpoints, resulting in abrupt and unnatural view switchings. Also, they do not consider the viewing delays incurred in data transmission and processing.

Some approaches propose to extend the navigation interactivity beyond the camera viewpoints by utilizing the virtual view synthesis techniques [23], [24]. In [9], [10], for example the users can access to any virtual views that are rendered using the two nearest coded camera views. The rendering can also be performed on the server side. In [25], the virtual views are encoded using predictive coding and stored in the server, before being streamed directly to the users. However the storage burden is largely increased because every accessible virtual view must be stored.

In order to support the high-quality rendering of virtual views, appropriate data representations are therefore extensively studied. In [2], [4], the light-field representation [26] is adopted for view synthesis due to its efficient and high-quality rendering. However the dense representation of light-field is heavily redundant and poses additional challenges in data compression and transmission. Some other data representation methods are considered to remove the data redundancy in the representation stage. In [12], [27], for example, the scene is represented using only one texture and one depth map, plus some auxiliary information that helps the view synthesis. However the choice of the appropriate auxiliary information is still an open question. In [28], the layered depth image format is used for data representation, where multiple images are constructed in layers corresponding to different levels of occlusion. Although data redundancy can be mostly removed from these representations, additional efforts are required to convert the captured data into the specific representation formats.

The increasing interactivity also brings challenges in data transmission, where the transmission policy needs to react to different requests of multiple users. In [7], [8], for example, the streaming of multiview video content in a navigation environment is studied, where the optimal transmission strategies are designed to provide high-quality content to heterogenous users under limited resources. All the above examples indicate that designing an interactive navigation system relates to many issues, including data representation, compression, transmission and rendering. While most existing works focus on a

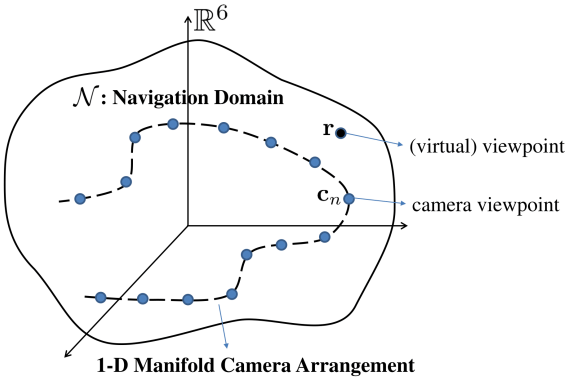


Fig. 3. Navigation domain: A user is able to navigate in the whole navigation domain, based on the data captured by camera viewpoints that lie on a 1-D manifold.

particular part of the system, only a few approaches investigate an end-to-end system design in the literature. In [29], [30], the classical rate-distortion optimization is extended to the interactive streaming scenario by considering the transmission rate and decoding complexity. However it mainly focuses on the coding aspect and does not consider the data representation at first. The design of effective solutions for multiview data representation and coding in the interactive navigation scenario is still an open problem.

Compared with the previous approaches, our work has the following contributions. First, we consider an end-to-end interactive navigation system design from data representation to rendering and propose to jointly optimize the novel trade-offs between navigation interactivity, bandwidth limitation and visual quality. Second, we show that the proper data representation plays an important role in optimizing the navigation system and we investigate practical solutions to find effective data representation strategies adapted to various navigation parameters (e.g., navigation speed, view popularity). Third, since the viewing delay caused by data transmission and processing is discussed in many works but often not properly handled, we propose a novel mechanism called navigation ball to prohibit the viewing delays and enable smooth user navigations. Fourth, we consider a rich 1-D manifold camera arrangement with high degrees of freedom in camera translation and rotation for user navigation, which extends the classical camera array arrangements.

III. Navigation Segment Representation for Low-Delay Navigations

In this section, we describe the proposed interactive multiview navigation system step by step. Based on that, we present our navigation segment representation.

A. 1-D Manifold Camera Arrangement

We are interested in a navigation scenario in a *static* 3-D scene, which is captured by a set of cameras positioned in different locations and orientations. A *camera viewpoint* in 3-D scene can be represented as a 6-D vector $\mathbf{c} = [x, y, z, \theta, \phi, \psi]^T$, where $[x, y, z]$ denotes the position and $[\theta, \phi, \psi]$ denotes the orientation. In our work, we study the challenging camera arrangement depicted in Fig. 2, where all camera viewpoints lie in a 1-D manifold embedded in the 6-D space $\mathbb{R}^6$. This camera arrangement greatly extends the navigation interactivity in terms of navigation range, where multiple degrees of freedom for camera motion can be replicated, including translation and rotation. For simplicity, we index each camera viewpoint along the 1-D manifold and denote it as $\mathbf{c}_n$, where $n \in [1, N_V]$ and N_V is the number of cameras. We assume that all the cameras provide both images and depth maps of the 3-D scene. We use Y_n to represent the image and the depth map captured at camera viewpoint $\mathbf{c}_n$.

B. Navigation Domain and Navigation Path

Similarly to the camera viewpoints, a (virtual) *viewpoint* in the 3-D scene can also be represented as a 6-D vector $\mathbf{r} \in \mathbb{R}^6$. The virtual views are rendered using a depth-image-based rendering (DIBR) technique [31] with data from the closest camera views. In a navigation scenario, the set of all accessible viewpoints within the navigation range forms the *navigation domain*, and it is denoted as $\mathcal{N} \subset \mathbb{R}^6$. Fig. 3 shows the navigation domain, which can be much larger than the camera set.

The user's navigation process is associated with a path traveling through all viewpoints visited by this user in $\mathcal{N}$. We call this path a *navigation path*. In practice, the navigation path is discrete and finite, due to finite frame rate f and bounded navigation period T. Then the total number of visited viewpoints in one navigation path is $N_f = Tf$. We define the navigation path P as the set that sequentially contains all visited viewpoints within the navigation period, i.e., $P = \{\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_{N_f}\}$, with $\mathbf{r}_i$ the i-th viewpoint in P.

C. Navigation Ball

When a user navigates along a path P, he will repeatedly request data in order to render views at each $\mathbf{r}_i$. However, the data response time will lag behind the request time due to the system delay, which includes the transmission delay and other data processing delays (e.g., decoding, rendering). Therefore, more data than the one required by the current viewpoint needs to be transmitted in order to compensate for system delays.

For that purpose, we introduce the concept of *navigation ball* as illustrated in Fig. 4. In more details, we assume that a data request is periodically sent by the user to the server every f_e frames, i.e., data request is sent at viewpoints $\mathbf{r}_1, \mathbf{r}_{1+f_e}, \mathbf{r}_{1+2f_e}$, etc. These viewpoints are called *requested viewpoints*. The set of all requested viewpoints forms a special subset of P, called the *requested path* $P_e = \{\mathbf{r}_1, \mathbf{r}_{1+f_e} \cdots \mathbf{r}_{1+(N_e-1)f_e}\}$, where $N_e = N_f/f_e$ is the number of requested viewpoints within a single path. Different from P that is purely related to user navigation, the requested path P_e is associated to actual data sent to the user, i.e., the data to be transmitted depends on the location of each requested viewpoint. For each $\mathbf{r} \in P_e$, we target to transmit data that enables the user to render any views in a neighborhood around $\mathbf{r}$. This neighborhood is called the navigation ball, and it is defined as

$$\mathcal{N}_B(\mathbf{r}) = \{\mathbf{r}' \in \mathcal{N} | d(\mathbf{r}', \mathbf{r}) \leq t(\mathbf{r})\Delta\}, \quad (1)$$

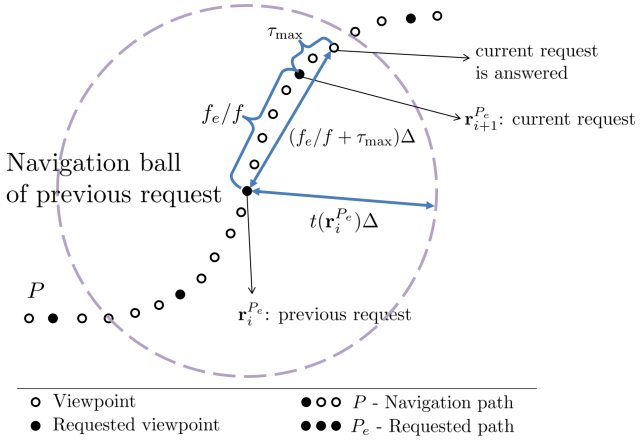


Fig. 4. Navigation ball for data buffering: A navigation ball gathers all viewpoints that will possibly be visited by the user before his next data request is handled by the server.

where $d(\cdot)$ is a distance function and $t(\mathbf{r})\Delta$ measures the size of the ball. The parameter Δ is the navigation speed describing the maximum velocity of the user in the navigation domain, and $t(\mathbf{r})$ is the tolerable delay of the navigation ball. By increasing $t(\mathbf{r})$, longer delay can be tolerated, and therefore more viewpoints can be visited without additional data from the server.

When $t(\mathbf{r})$ increases beyond a certain value, the effect of the system delay can be eliminated and the entire navigation becomes smooth. For the sake of simplicity, we assume a maximum system delay $\tau_{\max}$ for any data request. As illustrated in Fig. 4, the tolerable delay $t(\mathbf{r})$ requires to compensate for the overall delay consisting of the time interval between consecutive requests f_e/f (f is the frame rate) and the system delay $\tau_{\max}$, i.e.,

$$t(\mathbf{r}) \geq f_e/f + \tau_{\max}, \quad \forall \mathbf{r} \in P_e, \ \forall P_e. \tag{2}$$

When this inequality is satisfied at all requested viewpoints, the entire user navigation is smooth and there is no data starvation at the client side.

D. Navigation Segment Representation

An appropriate data representation format is crucial to the efficiency of data transmission and compression in the navigation system. For each data request, the system needs to *transmit only the data that is sufficient to cover the navigation ball of* $\mathcal{N}_B(\mathbf{r})$. Thus, the design of data representation should allow for certain flexibility to choose any potential subset of the whole multiview data. Similarly to [12], we investigate a data representation based on navigation segments. Basically, a *navigation segment* is a set of camera views Y_n, which is coded independently of the rest data. Suppose all camera views are divided into N_K navigation segments. The k-th segment is denoted by

$$V_k = \{n \in [1, N_V] \mid Y_n \text{ is in } k\text{-th segment}\}, \ \forall k \in [1, N_K],$$

and it corresponds to the set of indices of the camera views included in this segment. We further assume that the navigation segments are non-overlapping and connected along the underlying 1-D manifold of the camera views. In this case, the camera views in the left segment is always to the left of camera views in the right segment. An illustration of our navigation segment representation is shown in Fig. 5.

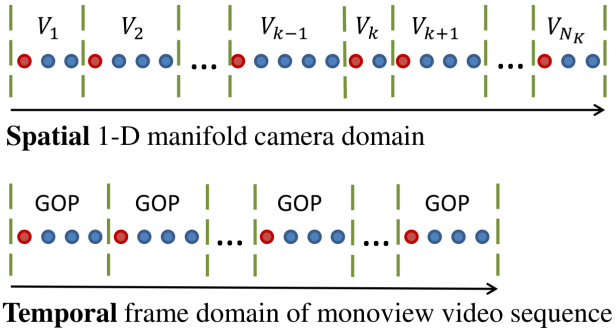


Fig. 5. Navigation segment representation as a spatial analogue to temporal GOP structure in monoview video sequences.

We adopt the following prediction structure in our work for coding views in a navigation segment. In each segment V_k, the first view is chosen as the anchor frame (i.e. I-frame), which is intra coded. The rest views in V_k are predicted frames (i.e. P-frames) using the previous view as the reference for prediction. The images and depth maps are coded using the same prediction structure. We use the state-of-the-art MV-HEVC standard [32], [33] as the compression engine, which is an extension of the HEVC standard [11] for coding multiview sequences. For navigation segment compression, alternative prediction structures are conceivable, but the performance gain is generally marginal [34].

In fact, navigation segments can be regarded as an spatial analogue to the temporal GOP structure in monoview video coding (see Fig. 5). The GOP structure supports temporal random access to frames, and the navigation segment structure supports the spatial random access to viewpoints. Since each segment V_k is independently decodable, the user only requires the necessary segments to enable the current navigation. The bandwidth and delay costs are thus reduced by avoiding the transmission of unnecessary segments. By further adjusting the shape of the navigation segments, the navigation system is able to quantitatively control the interactivity in terms of bandwidth and delay during the data transmission.

IV. Navigation System Optimization

A. Optimization Framework

Based on the navigation definition and the navigation segment representation, we now propose an optimization framework to optimize the full navigation system. In particular, we consider the following optimization problem

$$\begin{aligned} \min_{\mathcal{V},\mathcal{S},\mathcal{T}} \quad & U_R(\mathcal{V},\mathcal{S}) + \mu \cdot U_S(\mathcal{V}) + \nu \cdot U_D(\mathcal{V},\mathcal{S}) \\ \text{s.t} \quad & t(\mathbf{r}) \geq f_e/f + \tau_{\max}, \quad \forall \mathbf{r} \in P_e, \ \forall P_e. \end{aligned} \tag{3}$$

In this problem, we jointly optimize various *navigation costs* of the navigation system, including the compressed data size on the server $U_S(\mathcal{V})$, the transmission rate $U_R(\mathcal{V},\mathcal{S})$ and the view synthesis distortion $U_D(\mathcal{V},\mathcal{S})$. The parameters μ and ν are weights for $U_S(\mathcal{V})$ and $U_D(\mathcal{V},\mathcal{S})$ respectively. The

constraint enforces smooth navigation by using the navigation ball mechanism as indicated in Eq. (2).

The optimization variables denote the optimal design of navigation segments and navigation balls. In particular, $\mathcal{V}$ denotes the partition (or division) of navigation segments, while $\mathcal{S}$ denotes the allocation (or delivery) of navigation segments. The last optimization variable, $\mathcal{T} = \{t(\mathbf{r}) \mid \forall \mathbf{r} \in P_e, \forall P_e\}$, controls the size of the navigation balls. The solution to this problem deals with the optimal partition and allocation of the navigation segments and the optimal choice of every navigation ball, which provides the best trade-offs between the navigation quality and the resource consumption for the system. We next study each term of the cost function.

B. Navigation Costs

Storage cost

The storage cost U_S denotes the size of the compressed multiview data stored in the server. As we compress each navigation segment independently, the overall storage is the sum of the segment size, i.e.,

$$U_S(\mathcal{V}) = \sum_{k=1}^{N_K} h^{(Q)}(V_k), \tag{4}$$

where $\mathcal{V} = \{V_1, \cdots, V_{N_K}\}$ is the partitions of navigation segments. The function $h^{(Q)}(\cdot)$ is the generic compression function. When the navigation segment is predictively coded using the prediction structure as described Section III-D, it is written as

$$h^{(Q)}(V_k) = h_I^{(Q)}(Y_{i_k}) + \sum_{j \in \{V_k \setminus i_k\}} h_P^{(Q)}(Y_j|\hat{Y}_{j-1}), \tag{5}$$

where $h_I^{(Q)}(\cdot)$ and $h_P^{(Q)}(\cdot)$ represent the compression functions of I-frame and P-frame respectively with quantization step size Q. The notation i_k denotes the index of the first camera view in segment V_k, and $\hat{Y}_n$ is the reconstruction of Y_n.

We consider that the quantization step size Q is constant for all segments in order to stabilize the quality of all frames, which is important for a pleasant navigation with steady viewing quality. We further assume that the function $h_P^{(Q)}(Y_j|\hat{Y}_{j-1})$ is independent of the segment partition $\mathcal{V}$, because the quality of the reference view $\hat{Y}_{j-1}$ is steady in different partition choices given a fixed Q value.

Rate cost

The rate cost U_R denotes the transmission rate and it measures the navigation interactivity in terms of system bandwidth. In our work, we define the transmission rate as the size of total transmitted data per data request. We first express the transmission rate at a single requested viewpoint $\mathbf{r} \in P_e$ as

$$u_R(\mathbf{r}; \mathcal{V}, \mathcal{S}) = \sum_{k=1}^{N_K} h^{(Q)}(V_k) \cdot s(\mathbf{r}, V_k; t(\mathbf{r})), \tag{6}$$

where $s(\mathbf{r}, V_k; t(\mathbf{r}))$ is the indicator function for segment allocation. Its value is 1 if segment V_k is required for view rendering considering the navigation ball at $\mathbf{r}$ with size $t(\mathbf{r})$. Otherwise it is 0. The set $\mathcal{S} = \{s(\mathbf{r}, V_k; t(\mathbf{r})) \mid \forall \mathbf{r} \in P_e, P_e, k\}$ contains all indicator functions. Note that we treat the navigation segments as the minimum unseparated unit for data transmission.

From the perspective of a system, the definition of U_R should consider the navigation of different users, as they will have different navigation paths and accordingly different data transmission instances. Therefore we define U_R as the expected average transmission rate over all possible navigation paths:

$$U_R(\mathcal{V}, \mathcal{S}) = E_{P_e}\left[\frac{1}{N_e} \sum_{\mathbf{r} \in P_e} u_R(\mathbf{r}; \mathcal{V}, \mathcal{S})\right], \tag{7}$$

where we first compute the average transmission rate per requested path, and then take the expected value over all possible requested paths of the users.

It should be pointed out that, we assume a memoryless transmission scheme, where we do not consider the client's memory capacity. This means that the user does not reuse the data received at the previous requests. Therefore the transmission rate of a path is simply the sum of individual rates of each request.

View synthesis distortion

The view synthesis distortion U_D is the distortion in the rendered views and it represents the quality of navigation. We first denote the view synthesis distortion at a single viewpoint as $u_D(\mathbf{r}; \mathcal{V}, \mathcal{S})$, because both the partition and allocation of navigation segments influence this distortion. Similar to the rate cost, the distortion term U_D also requires to consider the navigation of different users. Therefore we represent it as the expected sum of distortion over all navigation paths:

$$U_D(\mathcal{V}, \mathcal{S}) = E_P\left[\sum_{\mathbf{r} \in P} u_D(\mathbf{r}; \mathcal{V}, \mathcal{S})\right]. \tag{8}$$

We first compute the sum of view synthesis distortion in a single navigation path, and then calculate the expected distortion over all possible navigation paths of the users.

C. Influencing Navigation Parameters

There exist many navigation parameters influencing the navigation system, like the quantization step size Q, the system delay $\tau_{\max}$, the weights μ and ν, etc. In our work, these parameters are not treated as optimization variables, but are regarded as input parameters of the optimization framework, since we focus our study on the data representation using the navigation segments.

V. Model-Based Problem Formulation

A. Overview

The above navigation problem in Eq. (3) is difficult to handle. First, we need to introduce rate and distortion models in order to properly deal with the distortion function $u_D(\mathbf{r}; \mathcal{V}, \mathcal{S})$ and the expectation operator $E_{P_e}[\cdot]$ and $E_P[\cdot]$ in the rate and distortion terms respectively. Second, it is difficult to solve the segment partition $\mathcal{V}$ and the segment allocation $\mathcal{S}$ simultaneously. In our work, we propose to

study the navigation problem by considering the following two subproblems.

1) We first consider a fixed allocation solution, namely $\mathcal{S}_0$. Given $\mathcal{S}_0$, we solve for the optimal segment partition $\mathcal{V}^\star$ and the optimal size of navigation balls $\mathcal{T}^\star$ in Eq. (3).
2) With the derived optimal $\mathcal{V}^\star$ and $\mathcal{T}^\star$ in 1), we further solve for the optimal segment allocation $\mathcal{S}^\star$ in Eq. (3).

This approach guarantees the optimal solution for users with fixed allocation solution $\mathcal{S}_0$, and provides suboptimal solution to users with other allocation solutions. We next present how we formulate these two subproblems using our rate and distortion models.

B. The Partitioning Problem with Fixed Segment Allocation

Fixed segment allocation $\mathcal{S}_0$

We consider a fixed allocation solution $\mathcal{S}_0$ that targets a low-distortion rendering. In order to define $\mathcal{S}_0$, we need to consider the reference views in DIBR. In many existing approaches, people use two or more reference views for DIBR in order to reduce the view synthesis distortion. However, in our work, we assume a single reference view due to the following reasons. First, the rendering quality is already satisfying with a single reference, because the virtual view is mostly derived from one reference while the rest of the references mainly provide side information for occlusion handling. Second, under the single reference assumption, the subsequent modeling process is much simplified and it becomes easier to solve the navigation problem in Eq. (3).

Under this single reference assumption, we define a fixed allocation solution $\mathcal{S}_0$ as follows. For any virtual viewpoint within the navigation ball, namely $\mathbf{r}' \in \mathcal{N}_B(\mathbf{r})$, we choose the camera view that is closest to $\mathbf{r}'$ as the reference view for rendering, and the index of this camera view is denoted as $l_0(\mathbf{r}')$. We then transmit the corresponding navigation segments that contain the camera view $l_0(\mathbf{r}')$ for all $\mathbf{r}' \in \mathcal{N}_B(\mathbf{r})$. Since any virtual view is assigned with its closest camera view for rendering, the solution $\mathcal{S}_0$ generally provides a low view synthesis distortion very close to the minimum value. However it does not guarantee the minimum transmission rate. The definition of $\mathcal{S}_0$ is consistent with the purpose of having a high quality rendering at the price of a potential suboptimal transmission rate.

Optimal size of navigation balls $\mathcal{T}^\star$

Under the allocation solution $\mathcal{S}_0$, the optimal value of $t(\mathbf{r}) \in \mathcal{T}$ can actually be inferred. As $t(\mathbf{r})$ grows, more virtual viewpoints are included in the navigation ball. As a result, each navigation segment will be requested more often, and consequently the rate term $U_R(\mathcal{V}, \mathcal{S}_0)$ in Eq. (3) will keep increasing. On the other hand, the distortion term $U_D(\mathcal{V}, \mathcal{S}_0)$ is not affected by $t(\mathbf{r})$, because the view synthesis distortion of each viewpoint $\mathbf{r}'$ is fixed due to its unique and determined reference view $l_0(\mathbf{r}')$ in $\mathcal{S}_0$. As a result, the objective function of Eq. (3) will keep increasing as $t(\mathbf{r})$ grows. Then the optimal $t^\star(\mathbf{r})$ is obtained when the equality holds in the smooth navigation constraint:

$$t^\star(\mathbf{r}) = f_e/f + \tau_{\max} \equiv t^\star, \quad \forall t(\mathbf{r}) \in \mathcal{T}. \tag{9}$$

In other words, all navigation balls will have the identical optimal size indicated by $t^\star$.

Modeling process: rate model

Based on $\mathcal{S}_0$ and $t^\star$ defined above, we now propose rate and distortion models in order to convert the original navigation problem in Eq. (3) into a solvable problem. We first rewrite the rate cost $U_R(\mathcal{V}, \mathcal{S}_0)$ using Eq. (6) and (7) as follows.

$$U_R(\mathcal{V}, \mathcal{S}_0) = \frac{1}{N_e} \sum_{k=1}^{N_K} h^{(Q)}(V_k) \cdot \alpha(V_k, \mathcal{S}_0), \tag{10}$$

where $\alpha(V_k, \mathcal{S}_0) = E_{P_e}\left[\sum_{\mathbf{r} \in P_e} s_0(\mathbf{r}, V_k; t(\mathbf{r}))\right]$ denotes the expected number of requests for segment V_k in one path. The notation $\alpha(V_k, \mathcal{S}_0)$ allows us to consider the global influence of the allocation solution $\mathcal{S}_0$ instead of looking into individual indicator functions. We note that $\alpha(V_k, \mathcal{S}_0)$ is influenced by the navigation ball. When $t^\star = 0$, $\alpha(V_k, \mathcal{S}_0)$ has the minimum value $\alpha_0(V_k, \mathcal{S}_0)$. As $t^\star$ increases, each navigation segment will be requested more often, and therefore $\alpha(V_k, \mathcal{S}_0)$ will keep increasing until the maximum value of N_e, which is the number of data requests in the navigation path. We propose to model this relationship using a monotonic decreasing function $g(t^\star)$ as follows,

$$\alpha(V_k, \mathcal{S}_0) = (1 - g(t^\star))N_e + g(t^\star) \cdot \alpha_0(V_k, \mathcal{S}_0). \tag{11}$$

We further derive $g(t^\star)$ using the following linear function, which has been verified empirically as shown in our technical report [34]. The truncation is to ensure that the range of $g(t^\star)$ is in $[0, 1]$.

$$g(t^\star) = \max(1 - 2t^\star\Delta/N_V,\ 0). \tag{12}$$

We next study $\alpha_0(V_k, \mathcal{S}_0)$. In that case, the navigation ball shrinks to a single viewpoint $\mathcal{N}_B(\mathbf{r}) = \mathbf{r}$, and the indicator function is degraded to $s_0(\mathbf{r}, V_k)$. We then use the following approximation

$$\alpha_0(V_k, \mathcal{S}_0) \approx N_e \sum_{n \in V_k} \int_{\mathcal{N}(Y_n)} p_r(\mathbf{r}) \mathrm{d}\mathbf{r}. \tag{13}$$

The subset $\mathcal{N}(Y_n) = \{\mathbf{r} \mid l_0(\mathbf{r}) = n\}$ is the set of viewpoints that require camera view Y_n for rendering, and the function $p_r(\mathbf{r})$ is the density function of $\mathbf{r}$. The integral denotes the probability that camera view Y_n is required for rendering. The summation of all $n \in V_k$ is then the probability of segment V_k that is required for rendering at each data request. The above approximation is only valid under the allocation solution $\mathcal{S}_0$.

We further define the *view popularity* by converting the density function of virtual viewpoints into the popularity function of camera viewpoints, i.e., $p_n = \int_{\mathcal{N}(Y_n)} p_r(\mathbf{r}) \mathrm{d}\mathbf{r}$. The popularity p_n represents the popularity of the camera view Y_n being required by users for view rendering. Finally we derive our rate model using Eq. (10) (11) and (13) as follows.

$$U_R(\mathcal{V}, \mathcal{S}_0) = \sum_{k=1}^{N_K} h^{(Q)}(V_k) \left(1 - g(t^\star) + g(t^\star) \sum_{n \in V_k} p_n\right). \tag{14}$$

In this model, the rate term is influenced by the partitions of navigation segments, the size of navigation balls and the view popularity.

Modeling process: distortion model

We now investigate how to model the distortion term $U_D(\mathcal{V}, \mathcal{S}_0)$ in Eq. (3). We first look at the view synthesis distortion at a single viewpoint, namely $u_D(\mathbf{r}; \mathcal{V}, \mathcal{S})$, which is in general difficult to estimate [35]. However, under the allocation solution $\mathcal{S}_0$, we can derive

$$u_D(\mathbf{r}; \mathcal{V}, \mathcal{S}_0) = u_D(\mathbf{r}, \hat{Y}_{l_0(\mathbf{r})}),$$

where we use the nearest camera viewpoint indexed by $l_0(\mathbf{r})$ as the reference view for rendering. With this, we can rewrite the distortion term in Eq. (8) as

$$\begin{aligned} U_D(\mathcal{V}, \mathcal{S}_0) &= E_P\left[\sum_{\mathbf{r}\in P} u_D(\mathbf{r}, \hat{Y}_{l_0(\mathbf{r})})\right] \\ &\approx \sum_{n=1}^{N_V} \int_{\mathcal{N}(Y_n)} p_r(\mathbf{r}) u_D(\mathbf{r}, \hat{Y}_n) \mathrm{d}\mathbf{r}. \end{aligned} \tag{15}$$

Here we further approximate the distortion term using the density function $p_r(\mathbf{r})$ and the subset $\mathcal{N}(Y_n)$ defined previously. The equation is very similar to Eq. (13) in the rate model, except that we have the distortion function $u_D(\mathbf{r}, \hat{Y}_n)$, which computes the view synthesis distortion given a single reference view.

We next estimate $u_D(\mathbf{r}, \hat{Y}_n)$. In DIBR, a virtual view image is first generated by warping the reference view image according to the corresponding depth map, and then inpainting is applied for hole filling [36]. Therefore we can separate the virtual view image into the hole regions and the non-hole regions, and compute the distortion of each region seperately [37]. In our work, we propose to estimate $u_D(\mathbf{r}, \hat{Y}_n)$ using the following equation:

$$u_D(\mathbf{r}, \hat{Y}_n) \approx D_{inp}\Omega(\mathbf{r}, \mathbf{c}_n) + D_{rec}^{(Q)}(WH - \Omega(\mathbf{r}, \mathbf{c}_n)), \tag{16}$$

where $\Omega(\mathbf{r}, \mathbf{c}_n)$ is the number of pixels in the hole regions at viewpoint $\mathbf{r}$ given reference view at $\mathbf{c}_n$. W and H are the width and height of the image respectively. The first term computes the distortion in the hole regions and we assume a constant inpainting distortion D_{inp} for each pixel location. The second term computes the distortion in the non-hole regions, where pixels are derived from the reconstructed reference view. We assume a reconstruction distortion $D_{rec}^{(Q)}$ which uniquely depends on the quantization step size Q. The detailed derivation process and the estimation of $\Omega(\mathbf{r}, \mathbf{c}_n)$ can be found in our technical report [34]. Finally, with Eq. (15) and (16), we derive the following distortion model

$$\begin{aligned} U_D(\mathcal{V}, \mathcal{S}_0) \approx\ & N_f D_{rec}^{(Q)} WH + \\ & N_f(D_{inp} - D_{rec}^{(Q)}) \cdot \sum_{n=1}^{N_V} \int_{\mathcal{N}(Y_n)} p_r(\mathbf{r})\Omega(\mathbf{r}, \mathbf{c}_n)\mathrm{d}\mathbf{r}. \end{aligned} \tag{17}$$

In this equation, the right hand side only contains the navigation parameters and does not have the optimization variables, because under $\mathcal{S}_0$ the view synthesis distortion does not depend on the segment partition $\mathcal{V}$. In particular, all components in the above expression clearly have determined values, except for the integral. In the integral, the size of the hole regions $\Omega(\mathbf{r}, \mathbf{c}_n)$ is uniquely determined by the viewpoints $\mathbf{r}$ and $\mathbf{c}_n$. The density function $p_r(\mathbf{r})$ and the subset $\mathcal{N}(Y_n)$ are both determined by the navigation domain. Therefore the integral also has a determined value though it is hard to compute. We finally note that with the distortion model in Eq. (17), the distortion term is not influenced by the optimization variables.

The partitioning problem given $\mathcal{S}_0$

We are now able to derive a solvable navigation problem by substituting the rate and distortion models in Eq. (14) and (17) into the original navigation problem in Eq. (3), and we have

$$\begin{aligned} \mathcal{V}^\star = \arg\min_{\mathcal{V}} & \sum_{k=1}^{N_K} \Big(h_I^{(Q)}(Y_{i_k}) + \sum_{j\in\{V_k\setminus i_k\}} h_P^{(Q)}(Y_j|\hat{Y}_{j-1})\Big) \\ & \cdot \Big(\mu + 1 - g(t^\star) + g(t^\star)\sum_{n\in V_k} p_n\Big), \end{aligned} \tag{18}$$

where the compression function is expanded using Eq. (5). Since the allocation solution is already given by $\mathcal{S}_0$ and the optimal size of navigation balls $t^\star$ is provided in Eq. (9), the only optimization variable in this problem is $\mathcal{V}$, which represents the partition of navigation segments. Thus this problem is called *the partitioning problem* in our work. The goal is to find the optimal $\mathcal{V}^\star$ that minimizes the rate and storage costs of the navigation system (note that the distortion cost is discarded using Eq. (17)).

C. Complementary Allocation Problem

Given the optimal size of navigation balls $t^\star$ and the optimal segment partition $\mathcal{V}^\star$ derived in Case 1, we further solve for the optimal segment allocation in the navigation problem of Eq. (3), and we derive the following problem.

$$\mathcal{S}^\star = \arg\min_{\mathcal{S}} \quad U_R(\mathcal{V}^\star, \mathcal{S}) + \nu \cdot U_D(\mathcal{V}^\star, \mathcal{S}) \tag{19}$$

Note that the storage cost is discarded in this problem, because it is fixed with the segment partition $\mathcal{V}^\star$ and it is not influenced by the segment allocation $\mathcal{S}$.

It is realized that we are able to further separate this problem into smaller problems for each data request at $\mathbf{r} \in P_e$ using the definition of rate and distortion costs in Eq. (7) and (8) as follows.

$$\begin{aligned} \{s^\star(\mathbf{r}, V_k; t^\star) \mid \forall k\} = \arg\min \quad & \sum_{k=1}^{N_K} h^{(Q)}(V_k) \cdot s(\mathbf{r}, V_k; t^\star) \\ & + \nu \cdot E_{\mathbf{r}}\left[u_D(\mathbf{r}', \hat{Y}_{l(\mathbf{r}')}) \mid \mathbf{r}' \in \mathcal{N}_B(\mathbf{r})\right]. \end{aligned} \tag{20}$$

In the cost function, the first term is the rate cost, which is further expanded using Eq. (6). The second term is the distortion term, which is the expected view synthesis distortion within the navigation ball $\mathcal{N}_B(\mathbf{r})$. The optimization variable are the streaming indicator functions $s(\mathbf{r}, V_k; t^\star), \forall k \in [1, N_K]$.

Here we still take the assumption of single reference rendering, but the camera index $l(\mathbf{r}')$ denotes the closest camera viewpoint to $\mathbf{r}'$ *given only the transmitted segments* indicated by $s^\star(\mathbf{r}, V_k; t^\star)$. Note that it is different from the rendering in $l_0(\mathbf{r}')$, which always uses the closest camera view as reference. With this, the solution allows to use farther camera views as references for rendering, which leads to possibly larger view

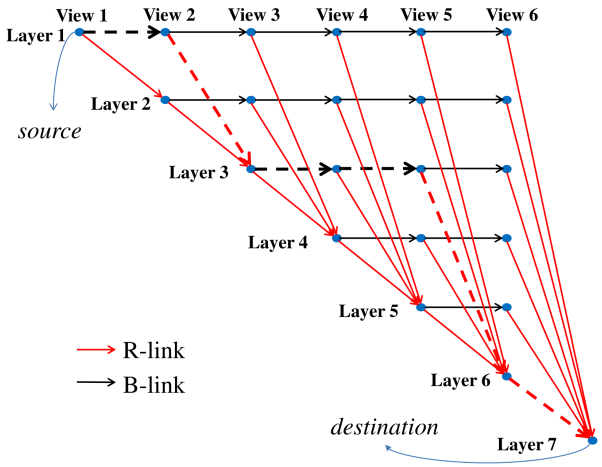


Fig. 6. The graph constructed for the partitioning problem (18): an example of 6 camera viewpoints. The dashed path represents a navigation segment structure of {view 1, 2}, {view 3, 4, 5} and {view 6}.

synthesis distortion. In the extreme case when $\nu \to \infty$, the solution converges to $\mathcal{S}_0$ that provides the minimum distortion. As ν decreases, the solution gradually provides larger distortion than $\mathcal{S}_0$, but with correspondingly lower transmission rate. Therefore, the derived allocation solution complements the fixed allocation solution $\mathcal{S}_0$ for practical users that might not be able to afford the amount of data required by $\mathcal{S}_0$ due to bandwidth limitations. It provides a flexible trade-off between the transmission rate and the viewing quality. We call this problem *the complementary allocation problem* in our work.

VI. Optimization Algorithm

In this section we propose solving algorithms for the partitioning problem in Eq. (18) and the complementary allocation problem in Eq. (20).

A. The Partitioning Problem

The partitioning problem in Eq. (18) can be solved using the Dijkstra shortest path algorithm, similar to a key view selection problem in [38]. The Dijkstra's algorithm works for all partitioning parameters, i.e., $i_k, \mathcal{V}_k, N_K$, simultaneously. Moreover, it is a fast algorithm that provides a globally optimal solution.

We first construct a graph where the vertices represents the views in the navigation domain. The views are organized in different layers that represent the potential navigation segments. An example of such a graph is given in Fig. 6. The nodes aligned in the same vertical line across different layers represent the same camera view. In each layer, the first node is encoded as an I-frame and the other nodes are encoded as P-frames according to our coding structure in each navigation segment. The last layer is the destination node, which is not a real camera view. The source node is the first node in the first layer. Two kinds of directed links, namely R-link and B-link, are assigned in the graph. Their costs are defined as follows. The link cost between neighbouring views in different layers (R-link) is the cost of starting a new navigation segment by adding the first view of the former layer as an I-frame. While the link cost between neighbouring views of the same layer (B-link) is the cost of adding the latter view to the current navigation segment as a P-frame.

We derive the link costs from the cost function of Eq. (18) as follows. First of all, the original cost function can be separated into the unary term and the pairwise term:

$$(\mu + 1 - g(t^\star)) \sum_{k=1}^{N_K} \sum_{n \in V_k} h_n + g(t^\star) \sum_{k=1}^{N_K} \sum_{n \in V_k} \sum_{m \in V_k} h_n p_m.$$

For simplicity, here we use h_n to denote the encoding bits of an arbitrary I-/P- frame for view n. The unary term aggregates only the encoding bits h_n of each view, while the pairwise term computes a pairwise cost of $h_n p_m$ for each ordered pair of (n, m) in segment V_k. Based on this, we are able to write the link costs as

$$\text{R-link}: e(v_i^l, v_{i+1}^m) = h_I^{(Q)}(Y_l) \cdot \big(\mu + 1 - g(t^\star) + g(t^\star) p_l\big), m > i$$

$$\text{B-link}: e(v_i^l, v_{i+1}^l) =$$

$$\begin{cases} h_I^{(Q)}(Y_l) \cdot g(t^\star) \cdot p_{l+1} + \\ h_P^{(Q)}(Y_{l+1}|\hat{Y}_l) \cdot \big(\mu + 1 - g(t^\star) + g(t^\star)(p_l + p_{l+1})\big), & i = l \\ \big(h_I^{(Q)}(Y_l) + \sum_{j=l}^{i-1} h_P^{(Q)}(Y_{j+1}|\hat{Y}_j)\big) \cdot g(t^\star) \cdot p_{i+1} + \\ h_P^{(Q)}(Y_{i+1}|\hat{Y}_i) \cdot \big(\mu + 1 - g(t^\star) + g(t^\star) \sum_{j=l}^{i+1} p_j\big), & i \geq l+1 \end{cases}$$

where we let $e(v_i^l, v_j^m)$ to be the link cost between view i in layer l and view j in layer m. The R-link $e(v_i^l, v_{i+1}^m)$ has the cost of starting a new segment with the first view of layer l, which is view l. Therefore, its cost is the unary term of view l plus the pairwise term of itself (l, l). The B-link $e(v_i^l, v_{i+1}^l)$ has the cost of appending view $i+1$ to the current segment of layer l. Therefore, its cost contains the pairwise term of $(j, i+1), \forall j < i+1$ (the first term of B-link), plus the pairwise term of $(i+1, j), \forall j \leq i+1$ and the unary term of $i+1$ (the second term of B-link). Moreover, the design of the graph in Fig. 6 guarantees that each unary and pairwise term is aggregated only once along any solution from the source to the destination. As a result, the sum of the link costs of any solution exactly corresponds to the value of the cost function in Eq. (18). Therefore, the shortest path solution of this graph is exactly the solution to the partitioning problem in Eq. (18).

Once the graph is constructed, we can apply the Dijkstra's algorithm to find the shortest path from the source to the destination. The resulting shortest path yields the optimal $\mathcal{V}^\star$, where each passed layer represents a navigation segment. For example, the dashed path in Fig. 6 represents three navigation segments: $V_1 = \{1, 2\}$, $V_2 = \{3, 4, 5\}$ and $V_3 = \{6\}$.

B. The Complementary Allocation Problem

In the complementary allocation problem of Eq. (20), the rate term is computed by adding up the sizes of the compressed navigation segments to be transmitted. The distortion term is estimated using Eq. (16). Differently from the partitioning problem, which can be pre-computed offline before the actual user navigation, the allocation problem has a real-time requirement, where the system needs to react immediately and selects the best navigation segments to be transmitted for each data request. For this purpose, instead of finding the optimal solution, we adopt an efficient heuristic algorithm for real-time processing. It contains three steps as follows.

Step 1: For each requested viewpoint $\mathbf{r} \in P_e$, determine the subset of its navigation ball, namely $\mathcal{N}_B^S(\mathbf{r}) \subseteq \mathcal{N}_B(\mathbf{r})$, using Eq. (1) but with a smaller tolerable delay $t_S \leq t^\star$.
Step 2: For each $\mathbf{r}' \in \mathcal{N}_B^S(\mathbf{r})$, find the nearest camera viewpoint indexed by $l_0(\mathbf{r}')$ and determine the corresponding navigation segment V_k it belongs to, i.e., $l_0(\mathbf{r}') \in V_k$.
Step 3: The streaming indicator function $s^\star(\mathbf{r}, V_k; t^\star)$ for all navigation segments is then determined by:

$$s^\star(\mathbf{r}, V_k; t^\star) = \begin{cases} 1, & \exists\ \mathbf{r}' \in \mathcal{N}_B^S(\mathbf{r}),\ l_0(\mathbf{r}') \in V_k \\ 0, & \text{otherwise} \end{cases}, \ \forall k. \tag{21}$$

In this solution, we always guarantee the best rendering quality for the virtual viewpoints that are closest to the requested viewpoint, i.e., the viewpoints within $\mathcal{N}_B^S(\mathbf{r})$, because the user will more likely visit these viewpoints than the ones farther away from the requested viewpoint $\mathbf{r}$. The rate-distortion trade-off, which is originally controlled by the weight ν in Eq. (20), can be alternatively achieved by changing the size of $\mathcal{N}_B^S(\mathbf{r})$, namely t_S. When $t_S = t^\star$, the solution is exactly S_0. As t_S decreases from $t^\star$ to a lower value, the algorithm will gradually request a smaller amount of navigation segments with smaller rate cost. The distortion will however increase because more and more viewpoints outside of $\mathcal{N}_B^S(\mathbf{r})$ will no longer have the closest camera view for rendering. Note that, in practice it is impossible to run the above algorithm for all viewpoints in $\mathcal{N}_B^S(\mathbf{r})$, simply because there is infinite number of them. Instead, we sample $\mathcal{N}_B^S(\mathbf{r})$ with equal distance, and run the algorithm only on the sampled viewpoints.

It should be pointed out that, although this solution is designed for our problem that assumes a memoryless transmission scheme, it can be extended to the case that considers the client's memory by simply avoiding the transmission of the repeated navigation segments received in the previous requests.

C. Complexity Analysis

We briefly analyze the complexity of our algorithms here. For the offline partitioning problem, the Dijkstra's algorithm runs in time $\mathcal{O}(|E|+|V|\log|V|)$ when a min-priority queue is used [39], where $|E|$ and $|V|$ represent the number of edges and nodes respectively. In our problem (Fig. 6), $|E| = N_V^2$ and $|V| = \frac{1}{2}N_V^2 + \frac{1}{2}N_V + 1$, where N_V is the number of camera viewpoints. The computational complexity is thus $\mathcal{O}(N_V^2 \log N_V)$. This complexity is tolerable when N_V is not huge. In our experiment, when $N_V = 450$, it costs around 120 seconds on average in Matlab on a Inter(R) Core(TM) i5-3320M PC, which suits for an offline solution. However it might still be necessary to speed up the running time especially when N_V keeps growing. One simple way to reduce the complexity is to assume a maximum number of camera viewpoints in each navigation segment. It can be verified that in this case the computational complexity reduces to $\mathcal{O}(N_V \log N_V)$.

The computational complexity of the complementary allocation problem is more critical, as this has to be solved during the real-time data transmissions. In the solutions of Eq. (21), the main computation lies in the search of the nearest camera viewpoint. We suppose that M points are sampled from the subset $\mathcal{N}_B^S(\mathbf{r})$. For each sampled viewpoint, the worst case complexity for finding its nearest camera viewpoint is $\mathcal{O}(N_V)$. Then the worst case complexity for solving the allocation problem is $\mathcal{O}(MN_V)$. Therefore we can adjust the complexity by changing the sampling rate in the solutions. In real implementations, when $N_V = 450, M \approx 200, t_S = 4s, \Delta = 20$ (a dense sampling for a considerable size of the navigation ball), the run time is around 1 second in Matlab on the same Inter(R) Core(TM) i5-3320M PC. This complexity is suitable for the real-time computation in the streaming session.

VII. Experimental Results

In this section, we evaluate the performance of the proposed navigation segment representation. In particular, we compute the resource consumptions in terms of storage and rate costs, and the navigation quality in terms of view synthesis distortion under different navigation configurations. We compare the proposed method with a baseline method, where the navigation segments are equally divided.

A. Experimental Setup

Dataset

We perform experiments on the New Tsukuba Stereo Dataset [13] [14], which provides groundtruth stereo image and depth pairs for 1800 camera viewpoints along a 1-D manifold trajectory. We further uniformly sample the 1800 camera viewpoints and obtain 450 of them, because practical navigation systems generally could not afford too many camera viewpoints due to resource limitations. Fig. 2 illustrates this camera arrangement.

Navigation parameters

We set the navigation parameters as shown in Table I. As mentioned in Sec. IV-A, these navigation parameters are not optimized in our problem, but are treated as input parameters of the optimization, i.e., their values are set beforehand. Note that the navigation speed Δ and the view popularity p_n are to be determined according to different navigation configurations in our experiments. The 450 image and depth pairs are encoded by the MV-HEVC engine [32] [33]. The quantization step size Q is controlled by the QP (quantization parameter) in the MV-HEVC engine. The higher the QP value, the larger the quantization step size. As aforementioned, we assume that the compression function $h_P^{(Q)}(Y_n|\hat{Y}_{n-1})$ does not depend on navigation segment partitions under the constant Q value. Therefore we only encode the pair of neighboring camera viewpoints and obtain 450 values of $h_I^{(Q)}(Y_n)$ and 449 values of $h_P^{(Q)}(Y_n|\hat{Y}_{n-1})$ (excluding the first camera viewpoint) in order to estimate the compression function in the optimization.

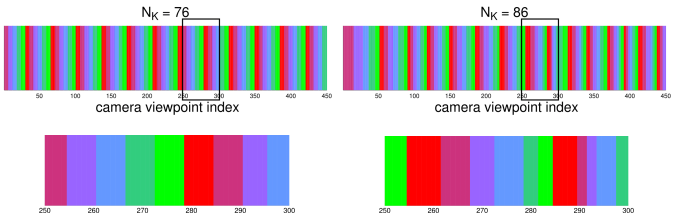
Comparison algorithms

In our experiments, we use "NBPA" (navigation ball and popularity-aware) to denote the proposed partitioning algorithm, and "NBPU" (navigation ball and popularity-unaware) to denote its popularity-unaware version where a uniform view popularity is assigned. We compare our algorithms with a

[1]QP values for P-frames and depth maps are assigned automatically using the default settings of the MV-HEVC engine.

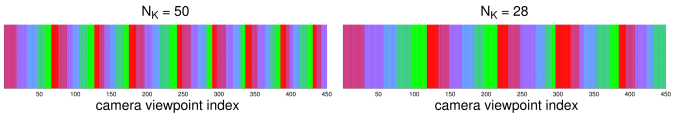
TABLE I
NAVIGATION PARAMETER SETTING

Notation	Value	Description
N_V	450	# camera viewpoints
T	90s	navigation period
f	30fps	frame rate
N_f	2700	# total frames ($N_f = Tf$)
f_e	90	request interval in frames
N_e	30	# requested viewpoints ($N_e = N_f/f_e$)
$\tau_{\max}$	1s	system delay
μ	0.05	weight for storage cost
Δ	TBD	navigation speed
p_n	TBD	view popularity
QP	25	quantization parameter for I-frames[1]
$h_I^{(Q)}(Y_n)$	-	encoding bit rates of I- and P-frames:
$h_P^{(Q)}(Y_n\|\hat{Y}_{n-1})$		determined by the MV-HEVC engine



(a) baseline equidistant (b) NBPU, $\Delta = 0$

Fig. 7. Partitions of navigation segments for different methods. Top: full partition patterns. Bottom: enlarged patterns for camera indices from 200 to 350.



(a) NBPU, $\Delta = 5$ (b) NBPU, $\Delta = 20$

Fig. 8. Partitions of navigation segments for proposed NBPU with different navigation speeds.

baseline method that corresponds to a blind N_K-equidistant partitioning, where the N_V camera views are equally divided into N_K non-overlapping navigation segments (when N_V is not divisible by N_K, a rounding is performed). We consider two types of baseline method. The first one is denoted as "Baseline", which always uses fixed value of N_K for different navigation configurations. The value of N_K is determined by evaluating the proposed cost function in the partitioning problem of Eq. (18) with the baseline equidistant partitions. Because it does not consider the navigation configurations, we set $\Delta = 0$, and p_n to be uniform respectively. The second one is denoted as "Baseline-NB", which further considers the usage of navigation ball, and the value of N_K is flexible for different values of Δ. We determine N_K by changing the value of Δ when we evaluate the cost function in Eq. (18) using the baseline equidistant partitions.

B. Partitioning Evaluation

We first evaluate only the partitioning results, where we compare different partitioning methods in terms of storage and rate costs under different navigation configurations of navigation speeds and view popularities.

Visual results

We compare the visual results of proposed method and the baseline method in Fig. 7. For fair comparison, we neglect the navigation speed ($\Delta = 0$) and the view popularity (uniform distribution) for the propose method, i.e., we use NBPU with $\Delta = 0$. We align the camera views along the 1-D manifold and use color bars to represent different navigation segments. It is clearly seen that, the proposed method provides an irregular partitioning compared to the baseline equidistant method. It is shown in our technical report [34] that this unbalanced irregular partitioning is due to the variation of the encoding bit rates. As shown later, this irregular partition brings us potential benefits in terms of storage and rate costs reduction.

Influence of navigation speed

We study next the influence of the navigation speed Δ on the navigation segment partitioning, and we use NBPU as we only focus on the navigation speed and we neglect the effect of the view popularity. Since the cameras in our dataset are almost equally placed along the 1-D manifold, the distances between neighboring cameras are similar so that they are denoted by the unit distance 1 for simplicity, and we measure Δ accordingly.

Fig. 8 illustrates the partitions of navigation segments of our NBPU under different values of Δ. It is seen that, as Δ increases, the navigation segments grow wider and sparser. This is because, with a larger Δ, more camera views are requested to support the view rendering within the increasing navigation ball, and therefore it becomes more efficient to compress and transmit a larger number of camera views together. As a result, the navigation segments tend to contain more camera views and therefore become wider.

Table II summarizes the comparison results of our NBPU and the baseline method, where we gradually increase Δ and compute the relative reduction of the partitioning function cost ("total"), the rate cost ("rate") and the storage cost ("storage") respectively according to Eq. (18). We highlight large performance improvements in green. It is observed that our NBPU outperforms the baseline methods in all aspects. Compared to the Baseline with fixed value of N_K , our NBPU achieves larger rate and storage cost reductions as Δ increases. This shows the effectiveness of considering the influence of navigation speed. When compared to Baseline-NB, the reduction is less as expected, because Baseline-NB is more flexible as N_K is further optimized for different values of Δ. The cost reduction against Baseline-NB is due to the irregular partitions of NBPU.

We also plot the corresponding rate-storage curves in Fig. 9. The navigation speed Δ increases from left to right. For all methods, the rate costs increase as Δ grows, because a larger Δ indicates a larger size of navigation balls and therefore more data is transmitted for data buffering at each request. The storage cost of Baseline is constant due to its fixed partition pattern. The storage costs of the other methods decrease as Δ grows, because a larger Δ leads to wider navigation segments and therefore more redundancies between camera views can be exploited and eliminated when encoding wider segments. In this figure, the gap between Baseline and Baseline-NB shows the cost reduction obtained by adapting the segment partitions

TABLE II
PERFORMANCE COMPARISON BETWEEN PROPOSED NBPU AND THE BASELINE METHOD UNDER VARIOUS NAVIGATION SPEEDS FOR A UNIFORM VIEW POPULARITY

Navigation speed	# Navigation segments			NBPU vs Baseline			NBPU vs Baseline-NB		
	Baseline	Baseline-NB	NBPU	total	rate	storage	total	rate	storage
0	76	76	86	-1.95%	-6.13%	-0.85%	-1.95%	-6.13%	-0.85%
5	76	46	50	-4.27%	-2.10%	-8.65%	-1.57%	-1.73%	-1.24%
10	76	35	38	-6.95%	-5.77%	-11.40%	-1.44%	-1.50%	-1.18%
15	76	27	29	-8.96%	-8.14%	-13.46%	-1.44%	-1.46%	-1.29%
20	76	23	28	-10.36%	-9.90%	-13.72%	-1.35%	-1.45%	-0.59%
25	76	19	20	-11.55%	-11.13%	-15.38%	-1.27%	-1.26%	-1.43%
30	76	17	19	-12.68%	-12.39%	-15.76%	-1.29%	-1.28%	-1.34%
Avg.	-	-	-	-8.10%	-7.94%	-11.32%	-1.47%	-2.12%	-1.13%

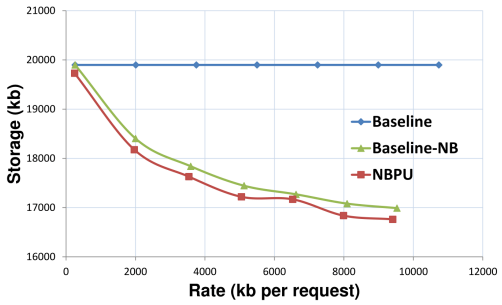
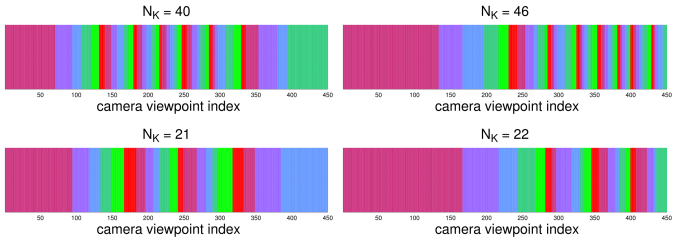


Fig. 9. Rate-storage curves of different partitioning methods under various navigation speeds for a uniform view popularity.



(a) center view preferred (b) right view preferred

Fig. 10. Partitions of navigation segments of proposed NBPA under pre-defined view popularity distributions at different navigation speeds. Top: low speed $\Delta = 5$. Down: high speed $\Delta = 20$.

to the navigation speed, and the gap between Baseline-NB and NBPU shows the additional cost reduction induced by applying the irregular partitions.

Influence of view popularity

We now study the influence of the view popularity that denotes the probability of each camera view being requested for view rendering during user navigation. It is generally non-uniformly distributed in practice due to the user preferences. Fig. 10 illustrates the segment partitions of our NBPA under several pre-defined view popularity distributions. A clear adaptation to the view popularity can be observed, where popular camera views have finer partitions and unpopular views have coarser ones. Note that the finer partitions generally lead to lower transmission rate and higher navigation interactivity due to the less data dependencies between camera views. The popular camera views are requested more often during navigation and consequently occupy a larger percentage of the transmission rate. Therefore it is more efficient to reduce the overall

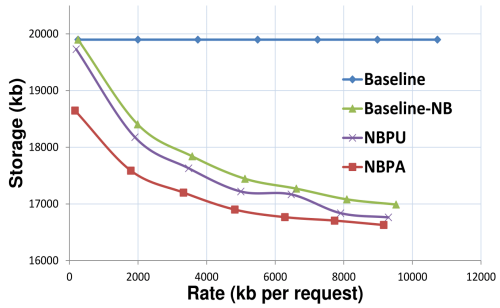


Fig. 11. Rate-storage curves of different partitioning methods under various navigation speeds for a popularity distribution that the right views are preferred.

transmission rate by applying a finer partitioning to these camera views rather than the unpopular ones.

We present the performance of different partitioning methods under various view popularity distributions in Table III, where the green box indicates large performance improvement and the red box indicates performance drop. We first observe that the popularity-aware NBPA provides significant rate and storage cost reduction against the popularity-unaware NBPU when the navigation speed is low (more than 17% rate reduction for $\Delta = 0$). However, as Δ increases, the performance gap becomes small. This is because, for large Δ, the partitions become coarser and the partition patterns under different view popularity distributions become more similar to each other, leading to a closer performance between them. Compare with the baseline methods, our NBPA achieves a further rate and storage reduction for non-uniform view popularities (referring to Table II for the uniform popularity). These results indicate that, by adapting the segment partitions to the view popularity, the proposed method is able to provide different levels of navigation interactivity for camera views and further reduce the rate and storage consumptions of the system.

We further visualize the corresponding rate-storage curves in Fig. 11 for the popularity distribution that the right views are preferred. The navigation speed Δ increases from left to right. The gap between NBPA and NBPU shows the additional rate and storage cost reductions of considering the influence of view popularity for the segment partitions.

Influence of other navigation parameters

The partition of navigation segment is also influenced by other navigation parameters. We briefly analyze them here.

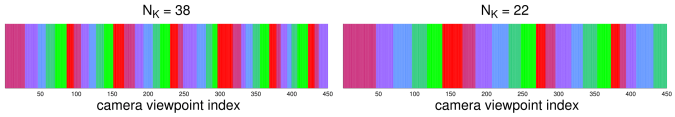
TABLE III
PERFORMANCE COMPARISON OF DIFFERENT PARTITIONING METHODS UNDER VARIOUS NAVIGATION SPEEDS AND PRE-DEFINED VIEW POPULARITY DISTRIBUTIONS.

Navigation speed	# Navigation segments				NBPA vs Baseline			NBPA vs Baseline-NB			NBPA vs NBPU		
	Baseline	Baseline-NB	NBPU	NBPA	total	rate	storage	total	rate	storage	total	rate	storage
0	76	76	86	71	-7.79%	-25.31%	-2.75%	-7.79%	-25.31%	-2.75%	-6.41%	-22.46%	-1.91%
5	76	46	50	40	-7.64%	-6.43%	-10.10%	-5.46%	-6.66%	-2.80%	-4.58%	-5.92%	-1.58%
10	76	35	38	31	-9.25%	-8.47%	-12.20%	-4.33%	-4.88%	-2.07%	-3.09%	-3.62%	-0.90%
15	76	27	29	23	-10.61%	-9.99%	-14.07%	-3.68%	-3.96%	-1.98%	-2.23%	-2.49%	-0.70%
20	76	23	28	21	-11.80%	-11.39%	-14.77%	-3.37%	-3.57%	-1.81%	-1.72%	-1.79%	-1.23%
25	76	19	20	19	-12.76%	-12.49%	-15.24%	-3.03%	-3.22%	-1.26%	-1.88%	-2.09%	0.17%
30	76	17	19	15	-13.64%	-13.41%	-16.16%	-2.70%	-2.77%	-1.81%	-1.25%	-1.32%	-0.48%
Avg.	-	-	-	-	-10.50%	-12.50%	-12.18%	-4.34%	-7.20%	-2.07%	-3.02%	-5.67%	-0.95%

(a) center view preferred (see Fig. 10a)

Navigation speed	# Navigation segments				NBPA vs Baseline			NBPA vs Baseline-NB			NBPA vs NBPU		
	Baseline	Baseline-NB	NBPU	NBPA	total	rate	storage	total	rate	storage	total	rate	storage
0	76	76	86	77	-12.03%	-34.91%	-6.29%	-12.03%	-34.91%	-6.29%	-7.44%	-17.27%	-5.49%
5	76	46	50	46	-10.76%	-10.33%	-11.61%	-8.28%	-10.06%	-4.44%	-5.33%	-6.32%	-3.24%
10	76	35	38	34	-11.64%	-11.13%	-13.56%	-6.40%	-7.11%	-3.58%	-3.96%	-4.34%	-2.43%
15	76	27	29	25	-12.61%	-12.16%	-15.06%	-5.40%	-5.79%	-3.12%	-3.30%	-3.55%	-1.85%
20	76	23	28	22	-13.48%	-13.17%	-15.74%	-4.77%	-5.01%	-2.91%	-2.77%	-2.83%	-2.34%
25	76	19	20	20	-14.16%	-13.95%	-16.04%	-4.21%	-4.42%	-2.20%	-2.11%	-2.25%	-0.78%
30	76	17	19	17	-14.76%	-14.60%	-16.43%	-3.65%	-3.78%	-2.13%	-1.43%	-1.49%	-0.80%
Avg.	-	-	-	-	-12.78%	-15.75%	-13.53%	-6.39%	-10.15%	-3.52%	-3.76%	-5.44%	-2.42%

(b) right view preferred (see Fig. 10b)



(a) $\mu = 0.05$ (b) $\mu = 0.5$

Fig. 12. Partitions of navigation segments for different μ (NBPU with $\Delta = 10$).

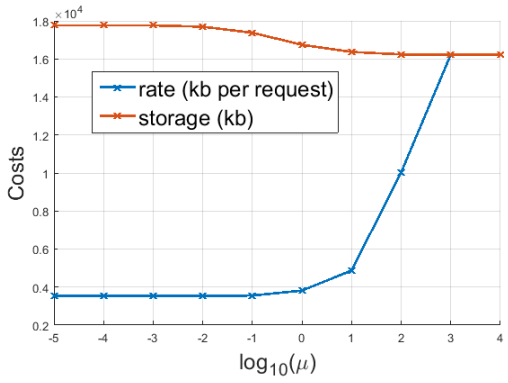
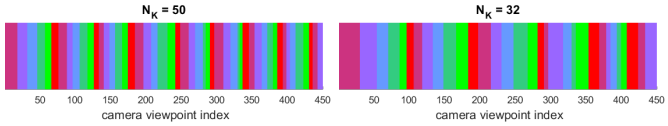


Fig. 13. Evolution of rate and storage costs versus storage weight μ (NBPU with $\Delta = 10$).



(a) $f_e = 30$ (b) $f_e = 120$

Fig. 14. Partitions of navigation segments for different f_e (NBPU with $\Delta = 10$).

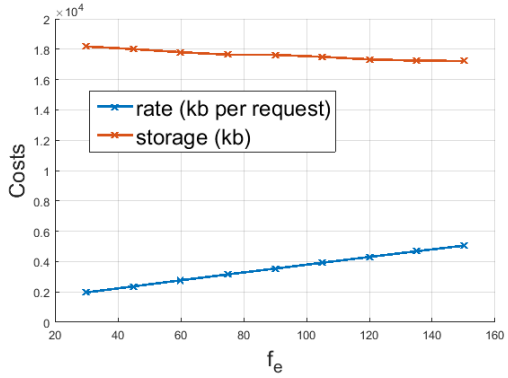


Fig. 15. Evolution of rate and storage costs versus f_e (NBPU with $\Delta = 10$).

The storage weight μ influence the relative weight between rate and storage costs, which further changes the optimal segment partitions. Fig. 12 shows the partition pattern of proposed NBPU for different μ. As μ increases, more weight is given to the storage, and the segments become wider; wider segments increase the compression efficiency and decrease the storage cost. On the other hand, wider segments increase the transmission rate. This evolution of rate and storage costs is plotted in Fig. 13 for the proposed NBPU. Different values of μ provide different combinations of rate and storage costs. The proper choice of μ depends on whether the storage or the bandwidth resource is more limited in practical navigation scenarios.

The request interval f_e changes the size of the navigation ball as indicated in Eq. (9), which further influences the partition of segments. Fig. 14 illustrates the partition pattern of NBPU for different values of f_e. Similarly to the navigation speed Δ, a larger f_e leads to a larger size of navigation balls, which further results in wider navigation segments. Fig. 15 plots the rate and storage costs versus f_e. Similar trend of rate and storage costs is observed due to the growing size of navigation segments as f_e increases. The choice of f_e in practical navigation systems depends on the storage and bandwidth limitations but also on the system delay. As long

TABLE IV
RD PERFORMANCE COMPARISON OF DIFFERENT METHODS AVERAGED OVER 100 SIMULATED NAVIGATION PATHS ALONG CAMERA VIEWPOINTS

Navigation speed	View popularity	NBPA vs Baseline						NBPA vs NBPU					
		BD-PSNR			BD-rate			BD-PSNR			BD-rate		
		Y	U	V	Y	U	V	Y	U	V	Y	U	V
10	left view preferred	0.2547	0.1608	0.1530	-4.36%	-4.93%	-4.80%	0.0094	-0.0056	-0.0071	-0.16%	0.16%	0.22%
	center view preferred	0.1490	0.1054	0.1134	-2.58%	-3.18%	-3.49%	-0.0401	-0.0323	-0.0257	0.71%	1.05%	0.89%
	right view preferred	0.1254	0.0823	0.0971	-2.12%	-2.53%	-3.00%	-0.0200	-0.0181	-0.0157	0.35%	0.55%	0.55%
20	left view preferred	0.4363	0.2708	0.2632	-7.35%	-8.10%	-8.10%	-0.0947	-0.0585	-0.0559	1.69%	2.01%	1.97%
	center view preferred	0.3123	0.2131	0.2104	-5.27%	-6.39%	-6.54%	-0.1029	-0.0567	-0.0548	1.83%	1.80%	1.79%
	right view preferred	0.1781	0.1179	0.1283	-3.04%	-3.63%	-4.00%	-0.1268	-0.0795	-0.0711	2.20%	2.67%	2.43%

as the storage and bandwidth capacities allow, a larger f_e is preferred because it leads to wider navigation segments and increases the efficiency of compression and transmission. On the other hand, more data is transmitted at each request for the larger f_e, which however increases the system delay for transmitting and processing the data. Therefore, f_e can not be arbitrarily large in order to enable low-delay navigation.

C. Complete System Evaluation

We now evaluate the performance of the complete system, i.e. a joint evaluation of partition and allocation of the navigation segments, where we further apply the allocation solution in Eq. (21) for each data request based on the different partitioning methods discussed above.

Navigation paths along real views

We first carry out the experiments on navigation paths composed of real camera viewpoints, where the ground-truth images are always available and we can compute the distortion directly. In our experiment, we generate simulated navigation paths from a pre-defined view popularity distribution. In particular, the first camera viewpoint in the navigation path is generated based on this distribution. In order to choose the next viewpoint, we build a navigation ball with navigation speed Δ centred at the previous viewpoint, and then randomly pick up a viewpoint from the normalized view popularity distribution within the navigation ball. This process is repeated until the last viewpoint in the navigation path is reached. Then interpolation is applied between consecutive viewpoints according to the frame rate. Fig. 16 shows the simulated navigation paths along camera viewpoints in our experiments. We further fix $t_S = t^\star$ in the allocation solution Eq. (21) to avoid the rendering of camera views caused by insufficient data, and therefore the distortion is uniquely affected by quantization. In this case, the allocation solution is indeed S_0. We then adjust the QP value in order to derive the rate-distortion (RD) curves. Four QP values $\{25, 30, 35, 40\}$ for I-frame are tested, where we rerun the partitioning algorithm for each QP value and apply the allocation algorithm afterwards. Table IV summarizes the RD performances of different partitioning methods, which is averaged over 100 simulated navigation paths. The Bjonteggard metric [40] is adopted for RD comparison.

It is observed that the proposed NBPA achieves better RD performance than the baseline method in all configurations, and the gain is larger as the navigation speed increases. This is mainly due to the great rate reduction achieved by the proposed method as discussed before. On the other hand, NBPA and NBPU have quite similar RD performances. If we compare Fig. 8 and Fig. 10, it is clearly seen that, under the same Δ, the segments are quite similar in popular views for different popularity distributions, while the main differences lie in the unpopular views. Since the popular views are required more frequently, it makes a primary contribution to the final RD performance. Therefore the similar segment patterns lead to the close RD performances between NBPA and NBPU. It should be reminded that, although the gain in RD performance is limited, considering the view popularity brings benefit to the system in terms of lower resource consumptions as demonstrated previously.

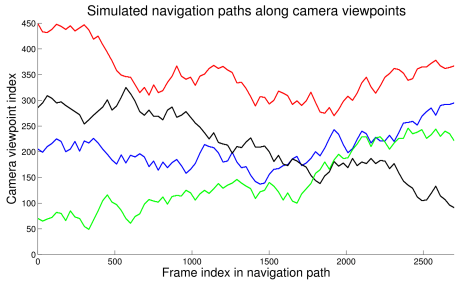


Fig. 16. Simulated navigation paths along camera viewpoints (pre-defined view popularity: center view preferred, navigation speed: $\Delta = 10$). The colors represent different navigation paths.

Navigation paths along virtual views

We next conduct experiments on navigation paths composed of virtual viewpoints which surrounds the real camera viewpoints. We generate them by adding a 6-D (position plus orientation) random shift onto the previously derived navigation paths composed of camera viewpoints. Fig. 17 shows the resulting navigation paths. Different from the camera viewpoints, the virtual viewpoints require view synthesis and the distortion is in general difficult to compute due to the unavailability of the ground-truth images. Instead, we estimated the view synthesis distortion using Eq. (16). We further fix QP $= 25$ so that the distortion only comes from inpainting, and we adjust t_S in Eq. (21) in order to obtain a series of RD points, based on which we plot the RD curves.

Fig. 18 illustrates the RD curves of different partitioning methods. Here the distortion is represented by the number of pixels in the hole regions, because it is proportional to the view synthesis distortion as we assume a constant inpainting distortion in Eq. (16). We further label the view synthesis as *successful* when the size of the hole regions is

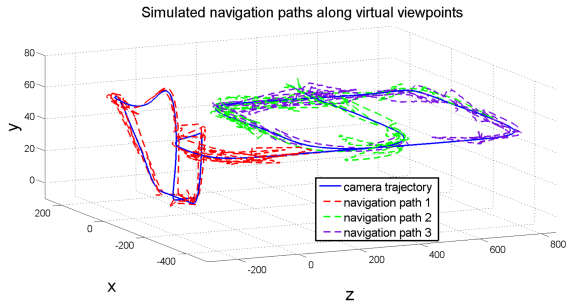
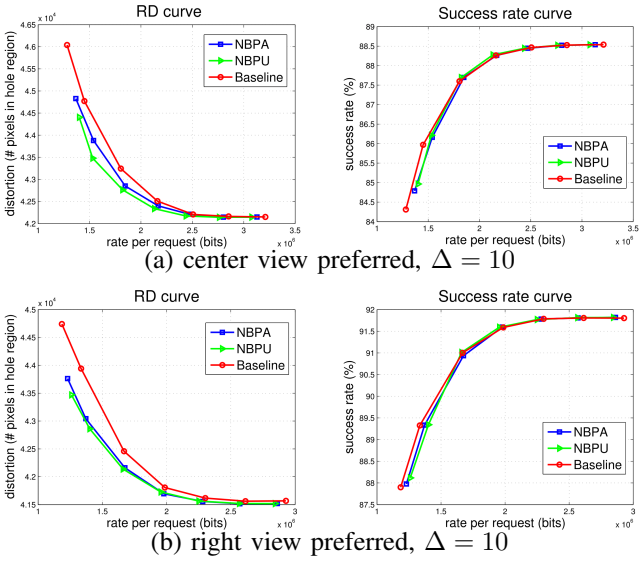


Fig. 17. Simulated navigation paths along virtual viewpoints plotted by their positions in 3-D space (center view preferred, $\Delta = 10$). The solid curve denotes the camera trajectory, while the dashed curves denote different simulated navigation paths.



(a) center view preferred, $\Delta = 10$

(b) right view preferred, $\Delta = 10$

Fig. 18. RD performances of different partitioning methods averaged over 100 simulated navigation paths along virtual viewpoints.

less than a threshold (half the image size in our experiment). Otherwise our distortion model is ineffective because the inpainting distortion drastically increases and the virtual views are heavily distorted. We therefore discard the unsuccessful rendering with extremely large hole regions and evaluate the RD performance only for successful ones. We further plot the success rate curves on the right side of Fig. 18 for reference. It is observed that all methods provide similar success rate curves. It indicates that similar percentages of successful rendering are evaluated for different methods and therefore the RD comparison between them is fair. The proposed NBPA clearly outperforms the baseline partitioning method in different configurations of navigation speed and view popularity. Similarly to the experiments on navigation paths along camera views, the RD gain mainly comes from a lower transmission rate provided by the proposed partitioning methods. We also note that NBPA and NBPU have very close RD performances, and sometimes NBPU is even better than NBPA. This is because the segment partition is optimized only for the fixed allocation solution $\mathcal{S}_0$. However, in this experiment, we adopt the complementary allocation solutions in order to derive different rate-distortion combinations. There is no guarantee that NBPA always has better RD performance than NBPU in this case. On the other hand, as we investigate the trade-offs between different navigation costs of the system, the consideration of the better solution should be evaluated in all different aspects. Although NBPA is less efficient than NBPU in terms of RD performance in some cases, it is validated in previous experiments that NBPA always achieves lower rate and storage costs than NBPU in different situations.

The above evaluations of the complete system demonstrate the effectiveness of the proposed navigation segment representation for a practical navigation system with low resource consumptions and high navigation quality.

VIII. Conclusion

In this paper, we study the optimal navigation segment representation for an end-to-end interactive multiview navigation system. Experimental results verify the effectiveness of the proposed data representation for practical navigation systems. Our method provides lower resource consumption, higher navigation quality and higher adaptation to different navigation parameters when compared to the baseline representation method. The idea of navigation segment representation can be further extended to more challenging navigation scenarios like the dynamic environment or more complex camera arrangements like the 2-D manifold camera arrays. These problems will be considered in our future work.

References

[1] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, "Multiview imaging and 3dtv," *IEEE Signal Process. Mag.*, vol. 24, pp. 10–21, Nov. 2007.

[2] M. Tanimoto, "Overview of free viewpoint television," *Signal Process. Image Commun.*, vol. 21, no. 6, pp. 454 – 461, 2006.

[3] K. Muller, P. Merkle, and T. Wiegand, "3-d video representation using depth maps," *Proc. IEEE*, vol. 99, pp. 643–656, Apr. 2011.

[4] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo, "Free-viewpoint tv," *IEEE Signal Process. Mag.*, vol. 28, pp. 67–76, Jan. 2011.

[5] Y. Chen, Y.-K. Wang, K. Ugur, M. M. Hannuksela, J. Lainema, and M. Gabbouj, "The emerging mvc standard for 3d video services," *EURASIP J. Appl. Signal Process.*, vol. 2009, pp. 8:1–8:13, Jan. 2008.

[6] K. Muller *et al.*, "3d high-efficiency video coding for multi-view video and depth data," *IEEE Trans. Image Process.*, vol. 22, pp. 3366–3378, Sept. 2013.

[7] A. De Abreu, L. Toni, N. Thomos, T. Maugey, F. Pereira, and P. Frossard, "Optimal layered representation for adaptive interactive multiview video streaming," *J. Visual Commun. Image Represent.*, vol. 33, pp. 255–264, 2015.

[8] L. Toni, T. Maugey, and P. Frossard, "Optimized packet scheduling in multiview video navigation systems," *IEEE Trans. Multimedia*, vol. 17, pp. 1604–1616, Sept. 2015.

[9] X. Xiu, G. Cheung, A. Ortega, and J. Liang, "Optimizing frame structure for interactive multiview video streaming with viewsynthesis," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, pp. 1–6, July 2011.

[10] X. Xiu, G. Cheung, and J. Liang, "Delay-cognizant interactive streaming of multiview video with free viewpoint synthesis," *IEEE Trans. Multimedia*, vol. 14, pp. 1109–1126, Aug. 2012.

[11] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, pp. 1649–1668, Dec. 2012.

[12] T. Maugey, I. Daribo, G. Cheung, and P. Frossard, "Navigation domain representation for interactive multiview imaging," *IEEE Trans. Image Process.*, vol. 22, pp. 3459–3472, Sept. 2013.

[13] M. Peris, S. Martull, A. Maki, Y. Ohkawa, and K. Fukui, "Towards a simulation driven stereo vision system," in *Proc. Int. Conf. Pattern Recogni. (ICPR)*, pp. 1038–1042, Nov. 2012.

[14] S. Martull, M. Peris, and K. Fukui, "Realistic cg stereo image dataset with ground truth disparity maps," in *Proc. Int. Conf. Pattern Recogni. (ICPR) workshop TrakMark2012*, vol. 111, pp. 117–118, 2012.

[15] M. Karczewicz and R. Kurceren, "The sp- and si-frames design for h.264/avc," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 637–644, July 2003.

[16] N. M. Cheung, A. Ortega, and G. Cheung, "Distributed source coding techniques for interactive multiview video streaming," in *Proc. Picture Coding Symp. (PCS)*, pp. 1–4, May 2009.

[17] G. Petrazzuoli, M. Cagnazzo, F. Dufaux, and B. Pesquet-Popescu, "Using distributed source coding and depth image based rendering to improve interactive multiview video access," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, pp. 597–600, Sept. 2011.

[18] G. Cheung, A. Ortega, and N. M. Cheung, "Generation of redundant frame structure for interactive multiview streaming," in *Proc. Int. Packet Video Workshop (PV)*, pp. 1–10, May 2009.

[19] G. Cheung, A. Ortega, and N.-M. Cheung, "Bandwidth-efficient interactive multiview live video streaming using redundant frame structures," in *Asia-Pacific Signal and Inform. Process. Assoc., Annu. Summit and Conf. (APSIPA ASC)*, pp. 498–501, 2009.

[20] Y. Liu, Q. Huang, S. Ma, D. Zhao, and W. Gao, "Rd-optimized interactive streaming of multiview video with multiple encodings," *J. Visual Commun. Image Represent.*, vol. 21, no. 5, pp. 523–532, 2010.

[21] E. Kurutepe, M. R. Civanlar, and A. M. Tekalp, "Client-driven selective streaming of multiview video for interactive 3dtv," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, pp. 1558–1565, Nov. 2007.

[22] A. M. Tekalp, E. Kurutepe, and M. R. Civanlar, "3dtv over ip," *IEEE Signal Process. Mag.*, vol. 24, pp. 77–87, Nov. 2007.

[23] H.-Y. Shum, S. B. Kang, and S.-C. Chan, "Survey of image-based representations and compression techniques," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 1020–1037, Nov. 2003.

[24] S.-T. Na, K.-J. Oh, C. Lee, and Y.-S. Ho, "Multi-view depth video coding using depth view synthesis," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 1400–1403, May 2008.

[25] T. Maugey and P. Frossard, "Interactive multiview video system with low decoding complexity," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, pp. 589–592, Sept. 2011.

[26] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. SIGGRAPH '96*, (New York, NY, USA), pp. 31–42, ACM, 1996.

[27] T. Maugey, P. Frossard, and G. Cheung, "Consistent view synthesis in interactive multiview imaging," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, pp. 2717–2720, Sept. 2012.

[28] U. Takyar, T. Maugey, and P. Frossard, "Extended layered depth image representation in multiview navigation," *IEEE Signal Process. Lett.*, vol. 21, pp. 22–25, Jan. 2014.

[29] I. Bauermann and E. Steinbach, "Rdtc optimized compression of image-based scene representations (part i): Modeling and theoretical analysis," *IEEE Trans. Image Process.*, vol. 17, pp. 709–723, May 2008.

[30] I. Bauermann and E. Steinbach, "Rdtc optimized compression of image-based scene representations (part ii): Practical coding," *IEEE Trans. Image Process.*, vol. 17, pp. 724–736, May 2008.

[31] C. Fehn, "Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv," *Proc. SPIE*, vol. 5291, pp. 93–104, 2004.

[32] G. Tech, K. Wegner, Y. Chen, and S. Yea, "3d-hevc draft text 1," in *Joint Collaborative Team on 3D Video Coding Extensions (JCT-3V) Document JCT3V-E1001, 5th Meeting: Vienna, Austria*, 27 July-2 Aug. 2013.

[33] G. J. Sullivan, J. M. Boyce, Y. Chen, J. R. Ohm, C. A. Segall, and A. Vetro, "Standardized extensions of high efficiency video coding (hevc)," *IEEE J. Sel. Topics Signal Process.*, vol. 7, pp. 1001–1016, Dec. 2013.

[34] R. Ma, T. Maugey, and P. Frossard, "Optimized data representation for interactive multiview navigation," *ArXiv e-prints*, 2017, arXiv:1705.02940 [cs.MM].

[35] R. Ma, N. M. Cheung, O. C. Au, and D. Tian, "Novel distortion metric for depth coding of 3d video," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, pp. 1714–1718, Sept. 2013.

[36] T. Maugey, P. Frossard, and C. Guillemot, "Guided inpainting with cluster-based auxiliary information," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, pp. 1702–1706, Sept. 2015.

[37] T. Maugey, A. Ortega, and P. Frossard, "Graph-based representation for multiview image geometry," *IEEE Trans. Image Process.*, vol. 24, pp. 1573–1586, May 2015.

[38] T. Maugey, G. Petrazzuoli, P. Frossard, M. Cagnazzo, and B. Pesquet-Popescu, "Reference view selection in dibr-based multiview coding," *IEEE Trans. Image Process.*, vol. 25, pp. 1808–1819, Apr. 2016.

[39] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, no. 3, pp. 596–615, 1987.

[40] G. Bjontegaard, "Calculation of average psnr differences between rd-curves," in *Video Coding Experts Group (VCEG) Document VCEG-M33, 13th Meeting: Austin, Texas, USA*, 2-4 Apr. 2001.