2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR) | 978-1-6654-9548-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/MIPR54900.2022.00057

Ultra-Sparse 360-Degree Camera View Synthesis for Immersive Virtual Tourism

Qian Zhou
Department of Computer Science
University of Illinois Urbana-Champaign
Email: qianz@illinois.edu

Klara Nahrstedt
Department of Computer Science
University of Illinois Urbana-Champaign
Email: klara@illinois.edu

Abstract

360-degree video based virtual tours are becoming more and more popular due to travel costs and restrictions. Existing solutions leverage teleport, 3D modeling or image morphing, but none of them offers satisfactory immersion and scalability. In this paper, we propose a morphing based ultra-sparse 360-degree camera virtual tourism solution. It uses a novel bus tour mode to improve immersion; besides, it uses a series of strategies to improve feature matching such that morphing works well for ultra-sparse (15 m apart) cameras and the system can be deployed on a large scale. The experimental results show that our work results in remarkably better feature matching and synthesized views.

1. Introduction

When watching a video produced by a camera, we are sharing the camera's vision and virtually standing at its position. Such video-based virtual tours have advantages over physical ones in time and travel costs, and are especially appreciated during pandemics due to travel restrictions.

A virtual tour will be immersive only if it allows the viewer to control her viewing position and direction in the virtual space and receive the corresponding view. Existing solutions fall into three types. 1) teleport [4, 8, 6, 11]: the system consists of arbitrarily sparsely deployed 360° cameras; the viewer can switch her view from one camera to another, changing her viewing position; at each position, she can control her viewing direction since a 360° camera has omnidirectional capture. Its drawback is *poor immersion*: the viewer's virtual position can only be at a point where there is a camera, so her movement is discrete. 2) 3D modeling [3, 9, 2]: the system uses a dense (centimeters or decimeters apart) camera constellation to conduct depth estimation and 3D reconstruction; it can synthesize the view in any direction and at any position within the space fenced by the constellation; thus, the viewer can continuously move in the virtual space. Its defect is *poor scalability*: because dense cameras are necessary, it cannot be deployed on a large scale, across a wide area to offer tourists rich scenery. 3) morphing [7, 5, 10]: it allows for sparser cameras, and uses image morphing to smoothly transform one camera's view to another's, making the viewer feel like moving continuously between two cameras' positions.

We think morphing is the overall best for virtual tourism due to its higher immersion than teleport and higher scalability than 3D modeling. Even so, its immersion and scalability are far from satisfactory. 1) Morphing can only synthesize views for the positions on a connection line segment between two cameras; the viewer will be frustrated when she tries and fails to walk out of a connection line. 2) To be deployed on a large scale, the cameras need to be ultra-sparse (tens of meters apart, like streetlights), but morphing requires the feature correspondences between two cameras' views as input, and we find that existing algorithms cannot find correspondences accurately for such sparse cameras.

In this paper, we aim to solve the two challenges of a morphing based virtual tourism system, where the distance between two adjacent 360° cameras is $\sim$15 m. To address the first challenge, we make each camera simulate a tour station and a connection line simulate a tour road between two stations; we elaborately position a tour as a bus tour instead of a pedestrian one, and use visual interfaces (e.g., steering wheel, bus window) to make the viewer realize that she is on a bus instead of walking in the virtual space. Since in real life a bus cannot go off a road and its passengers are still happy to see the scenery through the front or side windows, the viewer in our system should be fine that only on-road positions can be reached. To address the second challenge, we devise three filters to reduce the false positive feature correspondences produced by existing feature matching algorithms and one interpolator to alleviate the false negatives. They work in cascade and result in significantly improved feature correspondences and synthesized views. We claim our contributions as follows:

1. We propose a morphing based ultra-sparse 360° camera virtual tourism solution. It uses a novel bus tour mode to offer viewer-expected immersion.

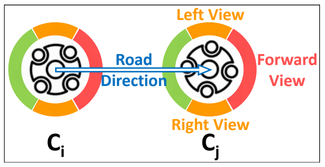
2. We devise three filters and one interpolator to improve the flawed feature correspondences produced by existing feature matching algorithms.
3. We conduct experiments using views from four tours, with four 360° cameras used in each tour. Our results show that the feature correspondences and synthesized views are remarkably improved.

2 Models and Assumptions

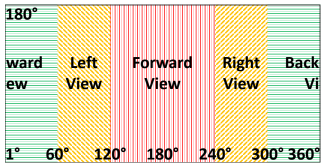
Model. We assume a graph formed by ultra-sparse (tens of meters apart) 360° cameras streaming 360° videos to the cloud. Each camera C_i is a vertex resembling a *tour station*; if two cameras C_i-C_j are less than 15 m apart and have a line of sight, an edge exists and resembles a *tour road*.

A viewer travels in the virtual space along a tour route (i.e. a sequence of connected edges) made beforehand or on demand. We assume that she provides the cloud with her virtual position (which spot of which road) and direction to the road, and receives the corresponding user view. If she is at station C_i, her view will be a subview of C_i's recorded omnidirectional view; if she is on road C_i-C_j, her view will be synthesized from C_i and C_j's views using morphing.

Forward View. For each camera pair C_i-C_j forming a road, the subviews along the road direction are called forward views, and those toward the left/right of the road are left/right views (Fig. 1a). View alignment [12] is conducted such that each camera pair has forward views located in the middle of the equirectangular panoramas (Fig. 1b).



(a) Camera pair seen from above.



(b) View locations in panorama.

Figure 1: 360° camera's views.

Morphing Based View Synthesis. Morphing takes C_i's view as I_0 (source image) and C_j's view as I_1 (destination image), and synthesizes intermediate view I_α for any $\alpha \in (0, 1)$. Here α increases from 0 to 1 as the viewer virtually moves from C_i to C_j. E.g., if she is virtually at the midpoint, $I_{0.5}$ will be synthesized and displayed to her.

Feature Arrow. A feature arrow (FA) is an arrow (denoted as $\overrightarrow{p_{src}p_{dst}}$) drawn on the overlay of I_0 and I_1, with its start point p_{src} (marked with $\bigcirc$) at a feature in I_0 and end point p_{dst} (marked with $+$) at the corresponding feature in I_1, as shown in Fig. 2. Thus an FA visualizes a feature correspondence. Like a vector, an FA's angle is the one it makes with the positive x-axis; its length is its magnitude.

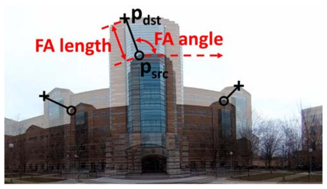


Figure 2: Feature arrow.

2.1 Challenges and Goals

Challenges. 1) Morphing can only synthesize views for the positions on a road; the viewer will be frustrated when she tries and fails to walk out of a road. 2) Morphing requires the feature correspondences between two cameras' views as input, but we find that existing algorithms cannot accurately detect and match correspondences for ultra-sparse cameras. Specifically, we test eight commonly used feature detection algorithms (e.g., SURF, ORB, KAZE [1]), and find that they generate either many correct FAs (true positives) with many wrong FAs (false positives) together, or fewer false positives but insufficient true positives. We denote the set of FAs produced by existing feature detection and matching algorithms as $\mathcal{FA}_0$, and show an example (produced by KAZE) in Fig. 3. Based on our manual labeling, $\mathcal{FA}_0$ is found to contain many false positives despite a lot of true positives. Its precision is only 0.56. Besides, numerous true positives are missed.

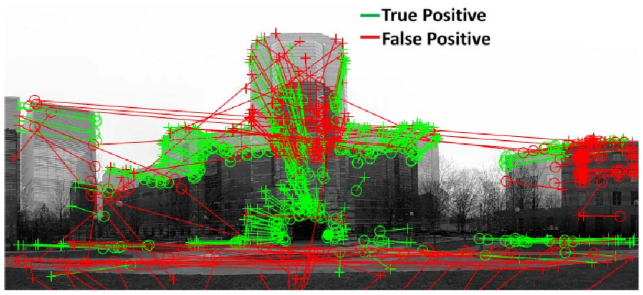


Figure 3: $\mathcal{FA}_0$ is highly flawed when two cameras are far apart from each other.

Goals. In this paper, we aim to offer viewer-expected immersion, and improve the highly flawed $\mathcal{FA}_0$ resulting from existing detection and matching algorithms and thus improve view synthesis. Due to space limitation, we focus on forward view synthesis; left/right views are also desired during tours and will be studied in the future.

3 Our Approach

Our approach consists of two parts: 1) a bus tour mode to improve immersion; 2) a series of strategies to improve $\mathcal{FA}_0$ under the ultra-sparse camera context.

3.1 Bus Tour Mode

Without indication, the viewer may think she is a pedestrian freely wandering in the virtual space, and will be frustrated when she tries to walk out of a road and gets rejected. The core of our solution is to proactively regulate the viewer's expectation to avoid disappointment.

We elaborately position a tour as a bus tour (Fig. 4) instead of a pedestrian one. Since in real life a bus cannot go off a road, but it still offers passengers good scenery through the front window or side windows, the viewer in our system should be fine that only on-road positions are reachable once she realizes that she is on a virtual bus.

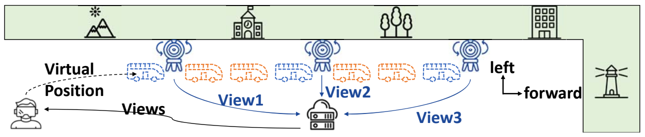
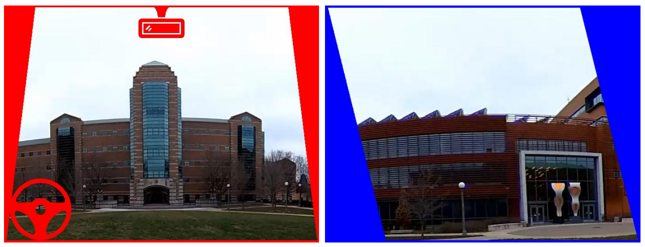


Figure 4: A virtual bus tour based on ultra-sparse 360° cameras. A viewer can obtain the cameras' views and virtually travel to the cameras' positions (denoted by blue buses); she can also obtain the synthesized intermediate views and virtually travel to the positions between cameras (orange).

Fig. 5 shows the visual interfaces we use to remind the viewer that she is on a virtual bus and to indicate which direction she is looking at. We overlay a shape of bus front window (including steering wheel and mirror) to the viewer's forward view; we overlay a shape of bus left/right window to the viewer's left/right view.



(a) Forward view. (b) Left view.

Figure 5: Visual interfaces.

3.2 Ultra-Sparse Camera View Synthesis

In this section we introduce our pipeline (Fig. 6) to generate feature correspondences between two cameras' views (denoted as $\mathcal{FA}_0$, highly flawed when cameras are ultra-sparse) using existing detection and feature matching algorithms, improve them with three filters and one interpolator, and apply the resulting correspondences $\mathcal{FA}_4$ to morphing for intermediate view synthesis.

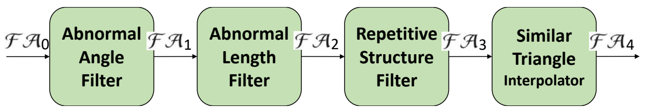


Figure 6: Pipeline.

3.2.1 Feature Detection & Matching (Output: $\mathcal{FA}_0$)

The algorithm first takes the equirectangular panoramas of two cameras as input: the source camera view is denoted as I_0 and the destination camera view is I_1. Originally they are both 360° horizontally by 180° vertically, we crop them, keep the central 180° by 180°, and feed the resulting I_0 and I_1 to feature detection and matching algorithms to automatically generate feature arrows $\mathcal{FA}_0$.

Our contribution here is not to design a new feature detector, but to find an existing one which fits our context the best. Specifically, a detector producing many correct FAs (true positives) with many wrong FAs (false positives) together is preferred to one producing fewer false positives but missing more true positives. It is because we find filtering out wrong FAs to be much easier than creating correct FAs missed by the detector. Our experiments on eight commonly used detectors show that KAZE [1] produces about 4 times as many FAs as SURF and ORB, and 10 times as many as others. Thus, we choose to use KAZE.

3.2.2 Abnormal Angle Filter ($\mathcal{FA}_0$ to $\mathcal{FA}_1$)

We observe an FA angle pattern that most true positives follow while false positives not. Based on this, we devise a filter to effectively remove the false positives which have abnormal angles. The pattern (Fig. 7a) is:

1) Radical in the Middle. The intuition is that as a person moves forward, the objects in the forward direction (with an x-coordinate of 180° in the panorama) will expand in her eyes, so an FA $\overrightarrow{p_{src}p_{dst}}$ should radiate outward from the image center c (i.e., $\angle\overrightarrow{p_{src}p_{dst}} = \angle\overrightarrow{cp_{src}}$) if x_{src} (the x-coordinate of p_{src}) is 180°.

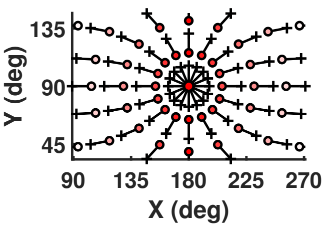
2) Horizontal on the Sides. The objects to the person's left (x-coordinate: 90°) or right (x-coordinate: 270°) will move parallel to her, so $\angle\overrightarrow{p_{src}p_{dst}}$ should be 180° if x_{src} is 90°, or be 0° if x_{src} is 270°.

As for an FA with $x_{src} \in (90°, 180°)$ or $(180°, 270°)$, the expected angle should be a weighted mean of $\angle\overrightarrow{cp_{src}}$ (radical) and 180° or 0° (horizontal); the closer it is to the center c, the more weight is given to $\angle\overrightarrow{cp_{src}}$. We find that a simple linear interpolation works well enough. The formula of the expected angle θ^* is:

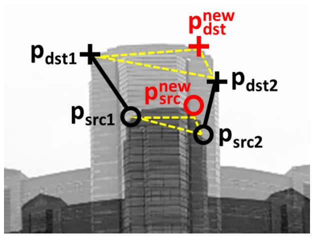
$$\theta^* = \begin{cases} (1-\lambda)\cdot\angle\overrightarrow{cp_{src}} + \lambda\cdot 0° & \text{if } x_{src} > 180° \\ (1-\lambda)\cdot\angle\overrightarrow{cp_{src}} + \lambda\cdot 180° & \text{otherwise} \end{cases} \tag{1}$$

$$\lambda = |x_{src} - 180°| \div 90° \tag{2}$$

Fig. 7a shows the overall look of the FAs with expected angles. An FA $\overrightarrow{p_{src}p_{dst}}$ will be regarded as abnormal and filtered out if $|\angle\overrightarrow{p_{src}p_{dst}} - \theta^*| > threshold_{\Delta\theta}$, a threshold for angle error. We empirically set the threshold as 20°.



(a) Angle filter.



(b) Interpolator.

Figure 7: Angle filter & interpolator.

3.2.3 Abnormal Length Filter ($\mathcal{FA}_1$ to $\mathcal{FA}_2$)

We observe that some false positive FAs have lengths which are dramatically different from their neighboring FAs. Thus, we put a filter after the angle one to remove the FAs with abnormal lengths. Specifically, we use a k-nearest neighbors filter: the filter computes the average length of an FA's k (we set k as 10) nearest neighbors as the FA's expected length, and will remove the FA if its actual length is too different (e.g., the relative error is above 60%).

3.2.4 Repetitive Structure Filter ($\mathcal{FA}_2$ to $\mathcal{FA}_3$)

By this stage quite some false positives still persist. We notice that many of them appear on repetitive structures like similar-looking windows, tiles and bricks, which are common on buildings. E.g., a wrong FA pairs the middle window of Floor 15 in I_0 with the middle window of Floor 14 or 16 in I_1. And it is extremely similar to the correct FA in terms of angle and length thus cannot be detected by the former two filters. To deal with this issue, we devise a filter targeting the FAs on repetitive structures.

There seems no good way to accurately distinguish repetitive structure false positives with true positives nearby, so highly targeted removal of only false positives is hard. But what if we aggressively kill those false positives at the expense of hurting true positives? *Our insight is that the improvement resulting from removing false positives surpasses the worsening resulting from removing **non-key** true positives.* We observe that: 1) many repetitive structures are located on a building image's interior rather than borders; 2) interior FAs are not key in image synthesis while border FAs are. This means that if we blindly remove interior FAs (both false and true positives are reduced), the synthesized image will: be improved due to fewer false positives; be worsened due to fewer true positives. And because those true positives are non-key, the improvement overrides the worsening. In this way, we no longer need to distinguish false and true positives; instead we just distinguish interior and border FAs, which is much more doable.

Based on this idea, we devise a repetitive structure filter which removes building interior FAs. First we use a ready-made segmentation CNN (Deeplab v3+) to partition I_0 into multiple regions labeled as sky, building, tree, etc. Then edge detection is applied to find building borders (e.g., the boundary between a building and the sky). The FAs which are on buildings but not close enough to building borders are regarded as interior FAs and removed.

3.2.5 Similar Triangle Interpolator ($\mathcal{FA}_3$ to $\mathcal{FA}_4$)

Now the number of false positives is in a low enough level, but false negatives remain to be improved: FAs are missed on some borders of some objects. Notice that such a lack of FAs barely results from the side effect of any of our three filters; FAs are lacked on some borders because the feature detector missed them and did no put them in $\mathcal{FA}_0$. To solve this problem, we choose interpolation, i.e. use existing FAs to generate new ones. This interpolator is designed based on the observation that the triangle formed by three neighboring feature points in I_0 is highly similar to the triangle formed by their corresponding points in I_1. It has two steps:

1) Interpolation Destination Determination. We need to determine which point in I_1 needs an interpolation. Such a point p_{dst}^{new} must meet two conditions: i) no feature exists nearby (this avoids overly dense features); ii) it is on a border, e.g., between the sky and a building (this avoids non-key interior features). Just like before, here we use the segmentation CNN to find borders.

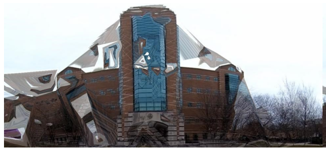
2) Interpolation Source Determination. Now we need to find the corresponding point of p_{dst}^{new} in I_0, which is denoted as p_{src}^{new}, then a new FA $\overrightarrow{p_{src}^{new}p_{dst}^{new}}$ is completely determined. In Fig. 7b, the algorithm searches p_{dst}^{new}'s nearest left neighbor p_{dst1} among the original feature points in I_1, and nearest right neighbor p_{dst2}. A triangle is formed with p_{dst1}, p_{dst2} and p_{dst}^{new}, and vertex angle $\angle 1$ and $\angle 2$ are obtained. Then the algorithm uses p_{src1}, p_{src2}, $\angle 1$ and $\angle 2$ to draw a similar triangle in I_0, whose third vertex is p_{src}^{new}.

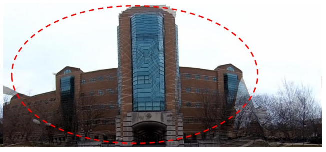
4 Evaluation

We conduct experiments with a self-collected dataset of 360° images corresponding to four outdoor tours. Each tour has four cameras deployed along a straight line, thus three pairs of adjacent cameras: C_i-$C_{i+1}, i = 1, 2, 3$. The spacing of two cameras in each pair is ∼15 m. Our algorithm takes each camera pair's views as input, finds feature correspondences, and synthesizes intermediate views.

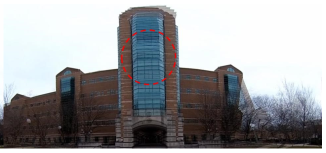
4.1 Qualitative Evaluation

Fig. 8 shows the synthesized views ($\alpha = 0.5$, i.e., the virtual position is the midpoint between two cameras) of one camera pair using different FAs, to demonstrate how the view evolves as $\mathcal{FA}_0$ passes our filters and interpolator and gets improved. A view has a red circle to mark the part which is remarkably better than the previous step.


(a) Synthesized view using $\mathcal{FA}_0$.


(b) Synthesized view using $\mathcal{FA}_1$.

(c) Synthesized view using $\mathcal{FA}_2$.


(d) Synthesized view using $\mathcal{FA}_3$.

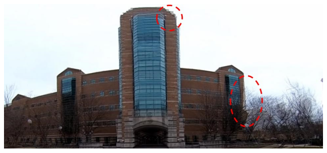
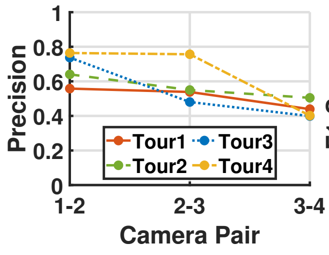
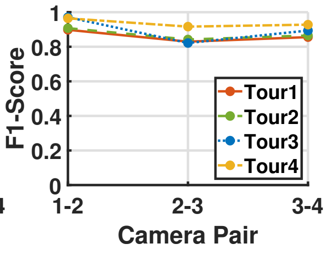

(e) Synthesized view using $\mathcal{FA}_4$.

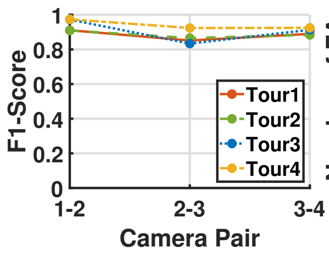
Figure 8: Synthesized views evolve as FAs get improved.



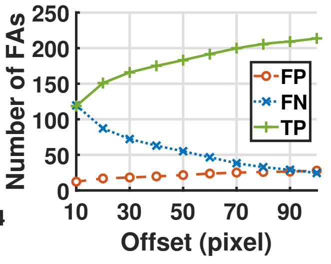
(a) $\mathcal{FA}_0$



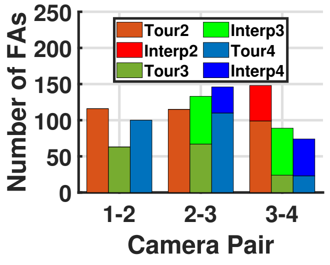
(b) $\mathcal{FA}_1$



(c) $\mathcal{FA}_2$



(d) $\mathcal{FA}_3$



(e) $\mathcal{FA}_4$

Figure 9: Performance evaluation.

As is seen, the view synthesized using $\mathcal{FA}_0$ has unacceptably low quality because $\mathcal{FA}_0$ is highly flawed. $\mathcal{FA}_1$ greatly improves the view as it removes most false positives with abnormal angles; $\mathcal{FA}_2$ makes the view better by further removing false positives with abnormal lengths; $\mathcal{FA}_3$ effectively reduces the ripples on repetitive structures; $\mathcal{FA}_4$ also alleviates artifacts and brings remarkable improvement.

4.2 Quantitative Evaluation

Here we measure the precision or number of FAs in each step. Fig. 9a shows that $\mathcal{FA}_0$'s precision varies from 0.40 to 0.76, and is poorly 0.56 in average, which means that about half of them are false positives. The average number of false positives per camera pair is 174.2.

Fig. 9b shows that after the angle filter is applied, we get a good F1-score of 0.89 (precision 0.82 and recall 0.98). It reduces the average number of false positives from $\mathcal{FA}_0$'s 174.2 to $\mathcal{FA}_1$'s 51.3. Also, the scores of all camera pairs are similar, so the angle filter is universally effective.

Fig. 9c shows that the length filter results in an F1-score of 0.90 (precision 0.86 and recall 0.96), only slightly higher than $\mathcal{FA}_1$'s 0.89 since most false positives have already been removed by the angle filter. It reduces the average number of false positives from $\mathcal{FA}_1$'s 51.3 to $\mathcal{FA}_2$'s 39.2.

The repetitive structure filter preserves building border FAs and removes interior ones. If an FA is on a building but its distance to the building's border is above a threshold $offset$, then it is detected as interior. A larger $offset$ makes the filter treat more FAs as border ones to preserve. Note that to this stage F1-score is no longer a good metric for performance evaluation because the repetitive structure reduces both true and false positives and $\mathcal{FA}_3$ may have a lower F1-score than $\mathcal{FA}_2$; but we have explained in Section 3.2.4 and shown in Section 4.1 that $\mathcal{FA}_3$ is better than $\mathcal{FA}_2$. So in Fig. 9d we show the numbers of false positives, false negatives and true positives. We set $offset$ as 40 pixels because it makes the average number of false positives drop to 19.8 in contrast to offset 100's 27.8; an offset of 10 pixels leads to even fewer false positives, but meanwhile too few true positives for view synthesis.

Fig. 9e shows that due to the interpolator, $\mathcal{FA}_4$ has 53.4 more FAs than $\mathcal{FA}_3$ in average.

5 Related Work

A virtual tour will be immersive only if it allows the viewer to control her viewing position and direction in the virtual space and receive the corresponding view. Existing solutions fall into three types.

1) Teleport. A system of this type [4, 8, 6, 11] consists of arbitrarily sparsely deployed 360° cameras; the viewer can switch her view from one camera to another, changing her viewing position; at each position, she can control her viewing direction since a 360° camera has omnidirectional capture. However, the viewer's virtual position can only be at a point where there is a camera, so her movement in the virtual space is discrete and unnatural.

2) 3D Modeling. A system of this type [3, 9, 2] uses a dense (centimeters or decimeters apart) camera constellation to conduct depth estimation and 3D reconstruction; it can synthesize the view in any direction and at any position within the space fenced by the constellation; thus, the viewer can continuously move in the virtual space. E.g., in [2] 46 cameras form a 92 cm diameter rig; in [9] 16 cameras form a 1 m diameter rig). However, because dense cameras are necessary, it cannot be deployed on a large scale, across a wide area to offer tourists rich scenery.

3) Morphing. Image morphing [7] is a visual effect technique transforming a source image I_0 to a destination image I_1. It allows for sparser cameras, and can synthesize two cameras' intermediate views, making the viewer feel like moving continuously between two cameras' positions. Lipski et al. present a hybrid approach [5] which combines 3D modeling with morphing but it does not aim at ultra-sparse cameras like ours. Photo Tourism [10] uses structure from motion (SfM) to recover the cameras' poses and the 3D geometry of the scenes, for browsing unstructured photos; it also uses morphing when viewers move between photos, but that is for making the switching process slightly smoother, neither targets nor achieves seamless transition. We choose morphing as our base due to its higher immersion than teleport and higher scalability than 3D modeling, but we improve its immersion and scalability to a new level.

6 Conclusion

In this paper, we introduce the design and evaluation of a morphing based ultra-sparse 360° camera virtual tourism solution. It uses a novel bus tour mode to achieve viewer-expected immersion. Besides, it improves the highly flawed feature correspondences produced by existing feature detection and matching algorithms when two cameras are far apart. As a result, our solution fits the ultra-sparse camera context and can be deployed on a large scale. The experimental results show that our work results in remarkably better feature matching and synthesized views.

Acknowledgment

This research was funded by the NSF CNS 1900875, by the Postdoctoral Fellowship program at CS UIUC and by the Grainger College of Engineering funding. The presented views in the article are of the authors and do not represent the views of the funding organizations.

References

[1] P. F. Alcantarilla, A. Bartoli, and A. J. Davison. Kaze features. In *European Conference on Computer Vision*, pages 214–227. Springer, 2012.

[2] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. Duvall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)*, 39(4):86–1, 2020.

[3] I. Choi, O. Gallo, A. Troccoli, M. H. Kim, and J. Kautz. Extreme view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7781–7790, 2019.

[4] X. Corbillon, F. De Simone, G. Simon, and P. Frossard. Dynamic adaptive streaming for multi-viewpoint omnidirectional videos. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 237–249, 2018.

[5] C. Lipski, F. Klose, and M. Magnor. Correspondence and depth-image based rendering a hybrid approach for free-viewpoint video. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(6):942–951, 2014.

[6] T. Maugey, L. Guillo, and C. L. Cam. Ftv360: a multiview 360° video dataset with calibration parameters. In *Proceedings of the 10th ACM Multimedia Systems Conference*, pages 291–295, 2019.

[7] P. K. Oswal and P. Y. Govindaraju. Image morphing: a comparative study. *Department of Electrical and Computer Engineering, Clemson University, Clemson*, 1998.

[8] H. Pang, C. Zhang, F. Wang, J. Liu, and L. Sun. Towards low latency multi-viewpoint 360 interactive video: A multimodal deep reinforcement learning approach. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 991–999. IEEE, 2019.

[9] A. P. Pozo, M. Toksvig, T. F. Schrager, J. Hsu, U. Mathur, A. Sorkine-Hornung, R. Szeliski, and B. Cabral. An integrated 6dof video camera and system design. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019.

[10] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006.

[11] K. K. Sreedhar, I. D. Curcio, A. Hourunranta, and M. Lepistö. Immersive media experience with mpeg omaf multi-viewpoints and overlays. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 333–336, 2020.

[12] Q. Zhou, B. Chen, Z. Yang, H. Guo, and K. Nahrstedt. 360viewpet: View based pose estimation for ultra-sparse 360-degree cameras. In *2021 IEEE International Symposium on Multimedia (ISM)*, pages 1–8. IEEE, 2021.