Image Stabilization for Hololens Camera in Remote Collaboration

Gowtham Senthil *, Siva Vignesh Krishnan *, Annamalai Lakshmanan *, Florence Kissling *

Abstract

With the advent of new technologies, Augmented Reality (AR) has become an effective tool in remote collaboration. Narrow field-of-view (FoV) and motion blur can offer an unpleasant experience with limited cognition for remote viewers of AR headsets. In this article, we propose a two-stage pipeline to tackle this issue and ensure a stable viewing experience with a larger FoV. The solution involves an offline 3D reconstruction of the indoor environment, followed by enhanced rendering using only the live poses of AR device. We experiment with and evaluate the two different 3D reconstruction methods, RGB-D geometric approach and Neural Radiance Fields (NeRF), based on their data requirements, reconstruction quality, rendering, and training times. The generated sequences from these methods had smoother transitions and provided a better perspective of the environment. The geometry-based enhanced FoV method had better renderings as it lacked blurry outputs making it better than the other attempted approaches. Structural Similarity Index (SSIM) and Peak Signal to Noise Ratio (PSNR) metrics were used to quantitatively show that the rendering quality using the geometry-based enhanced FoV method is better. Link to the code repository - https://github.com/MixedRealityETHZ/ImageStabilization

Keywords - Image Stabilization, Remote Collaboration, Scene Reconstruction, NeRFs

1. Introduction

AR has become an effective tool for real-time remote assistance and collaboration due to its immersive visual technology. However, the existing AR devices have many drawbacks [1], including limited field-of-view (64.69° for Microsoft Hololens 2 (HL) [2]) and motion blur. The narrow FoV negatively impacts the perception of the overall context in a scene. Moreover, during remote collaboration, these devices can provide blurry images due to rapid head movements and this is rather an unpleasant experience for the
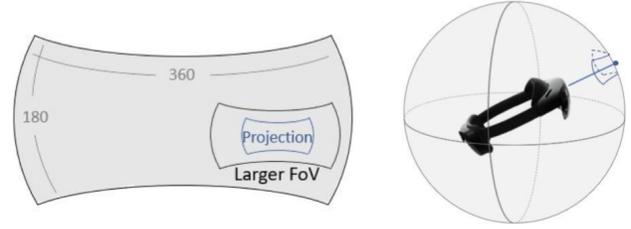

Figure 1. Concept of Enhanced FoV in HoloLens2

remote viewer. In this paper, we attempt to address these challenges by providing a more stable experience with a larger FoV as shown in Fig. 1 for the remote viewers of the AR device.

To provide a stable visual experience for remote viewers, we require a realistic and detailed geometric representation of the environment. However, generating a precise and realistic reconstruction of the 3D world in real time is infeasible with the current processing power in the AR headsets. Thus, we resort to a two-stage solution wherein the 3D reconstruction is performed offline, while an enhanced view is rendered from this representation during runtime. We capture a localized RGB-D sequence of the target environment and make use of this dataset to perform offline 3D reconstruction using the following two methods.

In the first method, we use depth information to build a point cloud and mesh representation of the environment. In the second method, we train a deep Neural Radiance Field (NeRF) representation using only RGB images. During runtime, the localized poses of the HL device are transmitted to the remote device, from where we can render the FoV-enhanced images from either of these representations to the remote user. Additionally, we post-process the images obtained from the first approach with image-enhancement techniques to generate realistic-looking images. The proposed pipeline is illustrated in Fig. 2.

2. Related Works

There are several methods for generating a coloured point cloud representation of a scene from multiple RGB

---

*G. Senthil and A. Lakshmanan are with the Department of Information Technology & Electrical Engineering, ETH Zürich. S. V. Krishnan is with the Department of Mechanical & Process Engineering, ETH Zürich. F. Kissling is with Department of Computer Science, ETH Zürich.
* These authors contributed equally to this work.
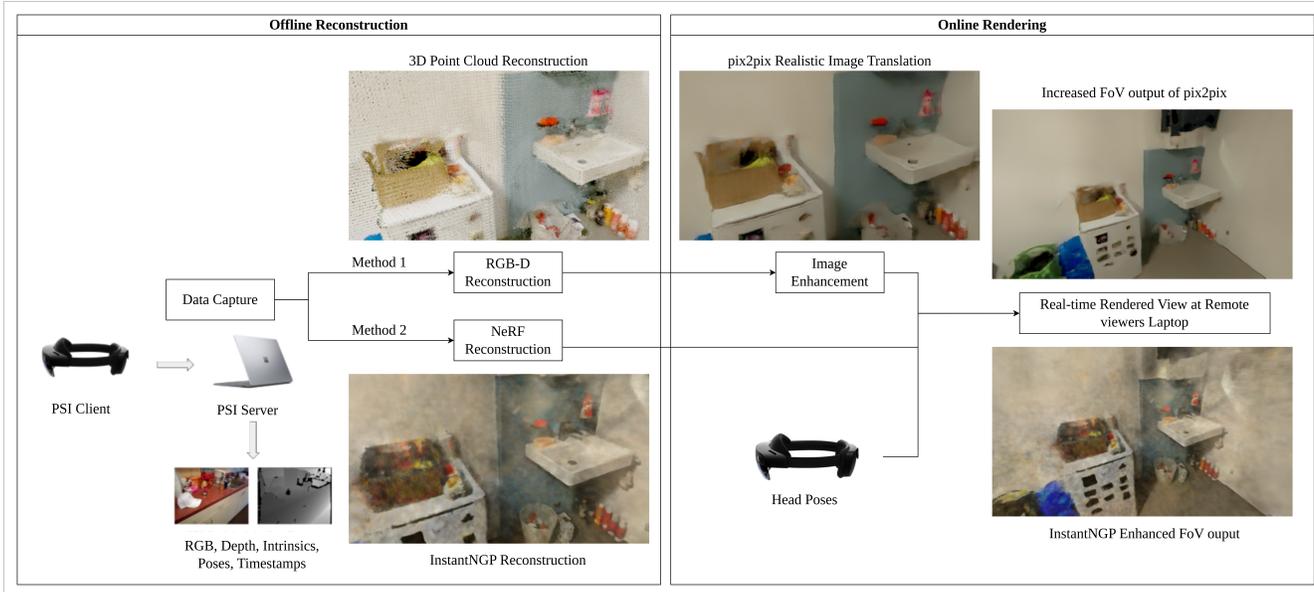E-mail of the corresponding author: sivavigneshk@gmail.com

Figure 2. Illustration of the proposed pipeline: Data is captured using HoloLens and PSI framework. In the first method, both RGB and depth images are used to get a 3D point cloud of the scene. Then an image translation model translates the point cloud to realistic-looking images and then an enhanced FoV image is rendered based on the head pose of the user of HoloLens. In the second method, InstantNGP based approach is used to reconstruct a scene and then based on the headposes of the user in realtime, enhanced FoV images are rendered

images. The most popular Structure-from-motion (SfM) [3] involves taking a sequence of images of an object or scene from different viewpoints and using them to reconstruct a 3D model. SfM works by extracting features from the images, such as points or lines, and matching them across different views. The relative positions and orientations of the cameras can then be estimated using these correspondences, and the 3D structure of the scene can be reconstructed. Multi-view Stereo (MVS) [4] involves taking multiple images of the same scene from different viewpoints and using them to estimate a dense 3D point cloud with or without calibrated camera poses. MVS works by finding correspondences between image pixels in different views and using them to reconstruct the 3D structure of the scene.

Both SfM and MVS can obtain high-precision estimates in textured areas. However, they face difficulty in feature matching of texture-less areas which are prominent in indoor scenes. This leads to the incompleteness of point clouds or a large number of outliers. Learning-based MVS approaches [5] attempt to overcome this problem with advantages in terms of accuracy and completeness in reconstruction. Methods such as planar priors [6], multiresolution [7] and depth map completion [8] also improve the results of the texture-less region, but they still cannot obtain satisfactory point clouds.

Introducing depth information along with RGB images will simplify the process of 3D reconstruction by a significant amount as discussed in works like 3D reconstruction of a scene with RGB-D information like accurate geometric registration [9], joint appearance and geometry optimiza-

tion [10].

Since renders of point cloud, despite having a lot of the details, do not look like images captured from the real world. So, we explore techniques to transfer views from the point cloud to the image domain. For this purpose, image-to-image translation and style transfer Generate Adversarial Networks (GANs) can provide good results. There are numerous such networks, both supervised and unsupervised, including some well-known supervised ones *pix2pix* [11], *pix2pixHD* [12], and *pSp* [13].

While [11] introduced the concept of generic image-to-image translation, its results were rather blurry and unsuitable for high-resolution images. The follow-up [12] has multiscale generator and discriminator architectures to generate high-resolution images. [14] introduces a new spatially-adaptive normalization layer to improve the quality of translating semantic maps into high-resolution images. [13] encodes the input image into a latent vector which is then passed to a pre-trained Style GAN [15] to obtain the translated image output, and was shown to generate high-quality image outputs. In our work, we demonstrate that the simplest *pix2pix* already provides reasonable quality results for view enhancement.

An alternative for rendering images at very high resolution is Neural Radiance Fields (NeRFs) [16] which recently shot into prominence. The model involves an implicit network of Multi-Layer Perceptron (MLP) that can reconstruct complex 3D scenes. However, training involves several hours if not days while rendering a scene would take around 1 minute. This is highly impractical for our appli-

cation. Fast-NeRFs [17] reduced this inference time to 5ms by efficiently caching the position's map in space. More recent works like NICE-SLAM [18] and Instant-NGP [19] can reconstruct scenes within a few minutes.

While other NeRF approaches work with just RGB images and poses, NICE-SLAM augments the depth information to give an environment with less noise. However, the images don't look real as it uses a hierarchical representation, first reconstructing it sparsely and then doing a fine-level refinement. Instant-NGP uses a multi-resolution hash-based encoding on MLPs and this reduces the reconstruction time to around 5 minutes while the inference time is brought down to 200ms. We proceed to implement the Instant-NGP model because of the faster training and almost real-time inference while also rendering realistic high-quality images.

3. Proposed Pipeline

3.1. Data Capture

HoloLens2 (HL) has an RGB camera sensor and a range (depth) sensor, essentially, having RGB-D information. We require a sequence of synchronized RGB images and depth maps with known poses for 3D reconstruction. HL's *Research Mode API* [20] exposes this data, which we access through a wired connection using Microsoft's Platform for Situated Intelligence (PSI) [21] applications. Due to the limited bandwidth of the wired connection, we were limited to recording a reduced resolution of RGB images than the maximum that the hardware supports. We record RGB images at 1280×720 RGB images at 25Hz, 320×288 depth maps at 5Hz, and also their corresponding extrinsic poses and intrinsics.

PSI's HoloLens CaptureApp is deployed on the HL with the above settings, and data is transferred to a wired Windows laptop running the PSI's HoloLens CaptureServer. The user walks around the target environment extremely slowly to ensure images are not very blurry and tries to cover all viewing angles for the entire environment. This takes approximately 5 minutes for a room of size $10 - 15 \ m^2$. This data is then processed with PSI's HoloLens Exporter to create a *train* dataset which is used to reconstruct the room. The *test* dataset is typically much shorter and contains fast motion and is used for both qualitative and quantitative evaluations.

3.2. RGB-D Mesh Reconstruction

Internally, HoloLens2 is able to output a *SpatialMesh* of the scanned scene. But we do not make use of it as it lacked colour information and as it was not very detailed. Hence, we tried to reconstruct a coloured mesh of the scene for rendering. We use the depth images and RGB images, along with the camera intrinsics, extrinsics (also refered to
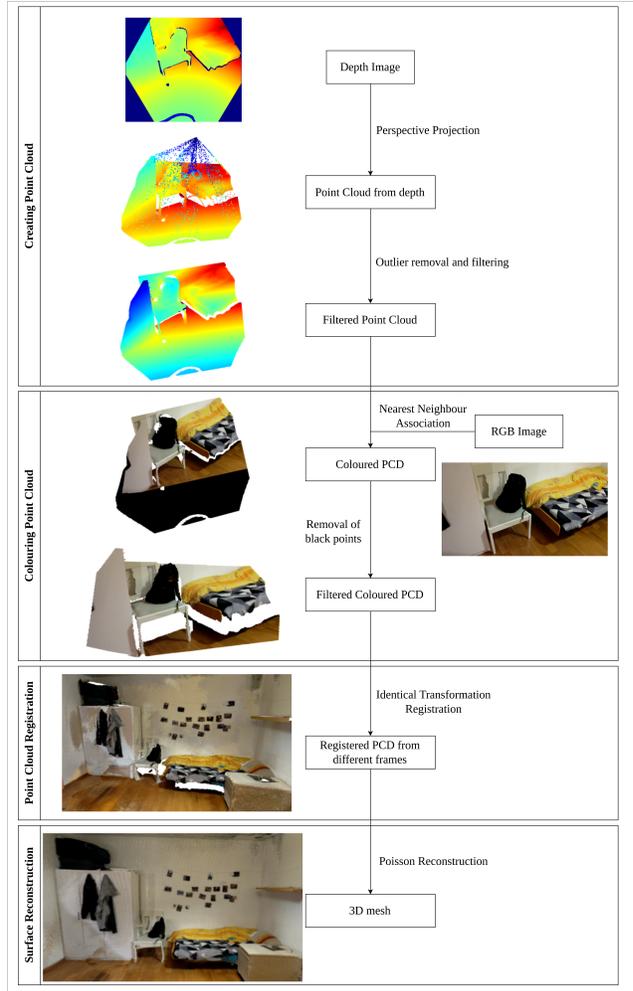

Figure 3. Illustration of the RGB-D mesh reconstruction

as poses in this article), and time stamps at which the image is captured to create the point cloud representation of the scene, in our case, a room. We follow the four steps below to do the same. Fig. 3 illustrates how a mesh is generated from RGB-D information from HoloLens.

Step 1: Creating the point cloud. We create a set of point clouds from each depth image in the dataset. For this, we first undistort the images and for each pixel in the undistorted image, we use the intrinsic parameters such as the focal length and the principal point and the depth value to calculate the 3D position of the point corresponding to that pixel in the camera's coordinate system. We use the Open3D [22] library and the location of the points to create a point cloud. Since the point cloud was noisy, we tried to remove them using a radius outlier filter and a statistical outlier filter from Open3D. Using these two filters, we were effectively able to reduce the noise. Depth cameras generally do not work well with highly reflective, transparent, or black surfaces and hence some of the points were very far away from or very close to the camera. This is an inherent

problem of the depth sensor as it uses time-of-flight to measure the distance. To overcome this, we used the fact that the data was recorded inside a room and away from various artefacts. Hence, we filtered the points and removed the ones are either very close to or very far away from the camera. Essentially, only the points that lie within a certain depth range were chosen for further processing. The distance range was manually fine-tuned to work well with various frames in various rooms.

Step 2: Colourizing the point cloud. For every depth image, we find the associating RGB image and raycast to get the colourized point cloud. Since the depth image and the RGB images are from different sensors, we associate them using the timestamp information employing the nearest neighbour approach. The point cloud obtained from that particular depth image is transformed into a world frame and then projected onto the image plane using the RGB camera intrinsics, and extrinsics, using the standard projective transformation function from OpenCV. We choose the depth points that have some colour assigned to them using the indices. We then colourise the point cloud using Open3D utilities. We remove the points that do not have a colour associated with them from the point cloud. Additionally, we removed the points that are close to the image edges (farther away from the optical center) as the distortion is higher in those regions. By doing so, we reduce the colour noise in the point cloud.

Step 3: Point Cloud Registration. Since all the point clouds from depth images are processed in a particular world frame, we stitch them together through identical transformation, after voxel-wise downsampling. We then apply statistical outlier removal once again to remove points that are further away from their neighbours exploiting the fact that the scene is a room. Since the overlapping regions were darker than the non-overlapping regions and since it affected the output of the later part of the pipeline (image translation network training), we stitched the points that were not already present or present very close to existing points, hence avoiding overlap. The colours of the point clouds were not very detailed for smaller objects present in the room. To address this, we implemented coloured-Interior Closest Point (coloured-ICP) algorithm [23] for point cloud registration instead of identical transformation to improve the association of colour and reduce the localisation errors of the camera poses. The coloured ICP was removed from the pipeline as we figured it does not improve the scene by even a negligible amount.

Step 4: Surface Reconstruction from Point Cloud. There are many standard algorithms available to create a 3D surface model from a set of 3D points such as Delaunay triangulation [24], Ball pivoting [25], Poisson Surface Reconstruction [26], and Isosurface extraction [27]. Since we require a detailed and accurate 3D model that is similar to the real-world ground truth, we used Poisson surface reconstruction. This method creates a watertight triangular mesh surface model by fitting a polynomial surface to the input points and using an octree data structure to reconstruct the surface. We employ the Open3D utility function to create a 3D mesh. For this, we first estimate the normals for the point cloud by locally fitting a plane per 3D point and then propagating the normal orientation using a minimum spanning tree to ensure consistency. Further, density-based filtering was employed to remove vertices and triangles that have low support from the 3D mesh. The mesh was then filtered to include only the region inside a bounding box defined by the point cloud. In this way, we reduced the triangles that were outside the room.

3.3. NeRF Reconstruction

For NeRF-based reconstruction, we use the Instant NGP model. Since we do not require depth maps for this model, we consider only the colour images (RGB) and camera poses.

Step 1: Selecting high-clarity images. NeRFs require high-resolution images to give a good representation of a 3D complex scene. As this model is trained with just RGB images, even having a few blurry images can result in a lot of noise in the model. In total 150-200 images are usually required for the reconstruction of a scene. We chose the number of images as 200 and the sharpness threshold as 150. To enforce these constraints, all the images were first checked for their sharpness. Based on their sharpness within a group, the sharpest image is selected. If the sharpness is lesser than the threshold for a certain image, the sharpness is increased to 150 using an unsharp mask. The number of images per group is decided inherently using an image selection function so as to obtain 200 images in total.

Step 2: Pose Refinement. The poses for the selected images are taken and the axis is transformed. Then they are converted to NeRF input-type poses which require a central alignment based on which the images are subject to transformations. Both the train dataset and test dataset are subjected to this pose refinement. The test dataset however does not require a separate central alignment-based transformation as we can use the transformation generated for the train dataset.

Step 3: Training and Rendering. Now, the images and the transformed poses of the train dataset are fed to the model which is able to reconstruct a scene. From the scene, we now give the test poses to render the images. Rendering is usually done by ray-casting across all pixels of an image to get the RGB values along the ray which essentially gives out the images required. We also render views with an enhanced FoV of $100°$ so as to increase the perspective. Both training and rendering were done using an RTX3090 with 8GB memory as Instant NGP requires a cuda compute

Figure 4. Qualitative comparison of renders with enhanced FoV from different reconstruction techniques for dataset 1. Top Row: Original image, point cloud, Mesh. Bottom Row: point cloud + Pix2pix enhancement, NeRF

capability greater than 7.0 to use Fullyfused MLP (responsible for bringing down the training time). For our scenes, the training time was between $5 - 7\ min$ and the rendering time per image was observed to be $200\ ms$.

3.4. View Enhancement

We use Open3D [22] framework to render image views from the point cloud or mesh representations using poses given by HL. However, the rendered images do not look like realistic images (see Fig. 4). We perform the discussed learned view enhancement step to make them look like realistic images. Here, we make use of a standard image-to-image translation architecture, *pix2pix* [11] which is trained on the *train* dataset.

During training time, the input to the network is a rendered view of the point cloud or mesh using HL pose, and the corresponding original image is used to train the GAN in a supervised fashion. The input images are cropped to 512×512 size and trained with random flips for 100 epochs with 2×10^{-4} learning rate. The learning rate is linearly reduced step-wise to 0 over another 100 epochs. The whole training takes around 36 hours on an RTX 2080 GPU.

During runtime, the HL poses from *test* dataset are used to render views from point cloud representation with increased FoV of about $100°$ (compared to $64.69°$ from HL2). This view is passed as input to the network to obtain the realistic image for the current frame. The inference time per frame is $7.4\ ms$ on a RTX 2080 GPU.

4. Implementation

4.1. Demo Application

We propose an application that deploys the image stabilization pipeline for the purposes of remote assistance. The application consists of a component on the HL and one on a Windows laptop.

On the HL side, the local user enables Research Mode such that the app can obtain the current head pose of the device. The app streams the live sequence of head poses over a TCP connection to the remote viewer's Windows laptop.

On the Windows laptop, a script listens to this TCP stream. Whenever a packet with a new pose arrives, the script will feed the pose to either the trained Instant-NGP model or the point cloud representation to obtain an enhanced image of the current HL view on the remote laptop. Thus the remote viewer can experience a better viewing quality in real-time given the 3D reconstruction that was performed offline.

Due to time constraints, we did not implement the proposed application. Nevertheless, we believe a real-time solution is feasible as the network delays introduced by our proposed application would be significantly lower than streaming the entire images from HL to the remote viewer.

4.2. Experimental Results

We capture two environments, each of which is an approximately $10 - 12\ m^2$ room, and capture a *train*, and *test* sequence for each room. We use the *train* dataset to train the NeRF network, perform RGB-D point cloud reconstruction,

Figure 5. Qualitative comparison of renders with enhanced FoV from different reconstruction techniques for dataset 2. Top Row: Original image, point cloud, Mesh. Bottom Row: point cloud + Pix2pix enhancement, NeRF

Dataset	point cloud	Mesh	point cloud + pix2pix	NeRF
Room1	11.71 | 0.31	10.83 | 0.15	**16.39** | **0.67**	12.18 | 0.56
Room2	11.73 | 0.30	10.77 | 0.16	**17.24** | **0.70**	14.30 | 0.65

Table 1. Quantitative comparison of renders of 2 test environments from different reconstruction techniques (PSNR (dB) | SSIM)

and train the view enhancement network. The train dataset Room1 after 3D reconstruction has 12418960 points in the point cloud and 2600068 vertices in the 3D mesh. The train dataset for Room2 after 3D reconstruction has 14236652 in the point cloud set and 2561906 vertices in the mesh. It is to be noted that the reconstruction of 3D coloured point cloud for a frame

We report the PSNR and SSIM [28] metrics of each representation's rendering (with the same FoV as HL) against the original images of *test* dataset in Tab. 1. While these are good metrics for image comparison and help us in evaluating the results, it is to be noted that the original ground-truth images from HL have a lot of motion blur for this dataset as it consists rapid motion. Hence, the numbers are not completely representative of the image quality.

The point cloud and mesh representations have limited details and do not resemble real images and hence have poor PSNR and SSIM scores on both datasets. The pix2pix enhancement improves PSNR by upto $5.5dB$ and SSIM more than doubles and renders more accurate images. The NeRF-based approach scores lower than pix2pix-enhanced renders, despite having more details on the background and a sharper image. This is mainly due to the smokey nature (as

seen in Fig. 5 of the NeRF reconstruction which in turn is due to the blurry images used to train the model.

In Fig. 4, the limited FoV of HL leads to missing out on the overall context of the room. This is solved by rendering views with more than 1.5 times the original FoV. In Fig. 5, we can see a simple case of motion blur from HL, which is even worse in general. Since our renderings are done frame-by-frame from offline reconstructions, we don't face the issue of motion blur and retain most of the detail of the scene. In both figures, it is clear that the point cloud or mesh representation does not resemble real-world lighting, despite having most of the details. NeRF representation while accurately representing the background is too noisy to be usable.

5. Limitations and Future Scope

Data transfer rates from HL to a wired Windows laptop were limited, and we could only transfer RGB images at a lower resolution and framerate than the maximum supported. This also meant lower resolution for training the DNNs, affecting the reconstruction quality and the viewer experience. In addition, we found that individual frames from the HL's video stream had lot of motion blur despite very slow movement. This greatly reduced the quality of NeRF reconstruction and supervised image-to-image translation and resulted in either smokey or blurry outputs. An alternative to solve these issues is to capture high-resolution localized images (instead of video) from the HL and use them for reconstruction, which should provide better-quality output.

The quality of the point cloud reconstruction appears to be limited by the accuracy of the depth camera in the HoloLens as the data is noisy and they fail catastrophically near edges and on objects or surfaces that are highly reflective or black. Additionally, the depth cameras cannot handle moving objects. For the former, we improved the point cloud by removing the outliers and removing the points that are very close and very far away from the camera. It is to be noted that those distance parameters were hand-tuned exploiting the fact that the scene reconstructed is a room with static elements. The same problem could be tackled more intelligently, in the sense that one could use the RGB image corresponding to the depth image and use deep learning approaches to either get a depth prior [29] or fit planes [30] to get more smooth depth values on highly reflective or black plane surfaces. We have ignored the latter as it is out of the scope of this project where we are working with static scenes.

Another factor that limits the quality of the scene reconstruction is the detailing in colour. To create a realistic and accurate 3D reconstruction, it is important to have high-quality colour information for the points in the point cloud. This can be difficult to obtain, especially if the points are far away from the camera (loss of detail) or if the lighting conditions are poor or changing. Invariant illumination could be tackled by faster sensor motion [31]. There are other factors that diminish the quality of colours in the reconstructed scenes such as: 1. the geometric 3D model being noisy and inaccurate; 2. the RGB camera not being in perfect correspondence with the depth camera, thereby increasing the misalignment of the projected images; 3. the RGB images suffering from blurring and ghosting. To address these sources of errors, one could try to maximize the photometric consistency by jointly optimizing for the camera poses and non-rigid correction function for all images as in [32].

Processing the point cloud for a room is another bottleneck due to its size. In order to increase the processing speed, one could employ feature-based down sampling approaches [33]. For example, downsample less in the region where there are many features, and downsample more in the textureless, planar regions (such as a wall).

The view enhancement *pix2pix* network generates high-quality results for the cropped image size it was trained on. However, when testing on full-resolution image sizes, it results in blurry outputs with less detailing. This can be resolved by either training on full-resolution images which in turn would require more GPU memory and training time or trying out more recent supervised image-to-image translation architectures such as *pix2pixHD* [12], *SPADE* [14], and *pSp* [13]. These tackle the problem of fewer details by working at multiple scales to obtain high-resolution images.

The major disadvantage of a NeRF model is the presence of noise clouds in any scene. While the NeRF model [19] is able to generate seemingly high-quality images, the amount of noise here is higher. This can be mainly attributed to the 8-megapixel camera of Hololens 2, the low-resolution captured data and the blurriness of the images. One way to reduce the noise while not compromising the image quality was discussed in NeRF-SLAM [34] which uses depth information in the Instant-NGP model to reduce the noise.

6. Conclusion

To address the problem of image stabilisation for remote collaboration using AR devices, we proposed a two-stage pipeline. The solution includes an offline 3D reconstruction of an indoor environment, followed by enhanced and smoothed renderings of the reconstructed scene during real-time collaboration. Offline reconstruction is chosen because the current AR headsets do not provide high-computational requirements. For the 3D reconstruction, a geometry-based and a fast NeRF-based (InstantNGP) methods were employed. Experimental results highlight that the rendered views using these approaches can provide a smooth experience for the remote viewer while offering more contextual awareness of the environment than the actual AR device output. The rendering times for both approaches are small and can be done in real-time during remote collaboration. Geometry-based reconstruction with image translation for rendering was found to be better than the NeRF-based method because of faster rendering and lack of noise.

7. Acknowledgements

References

[1] Wnag Shiyao et al. Augmented reality as a telemedicine platform for remote procedural training. *Sensors*, 2017. 1

[2] Microsoft. Locatable camera overview. *https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/locatable-camera-overview*. 1

[3] Onur Ozyesil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion, 2017. 2

[4] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 1:519–528, 2006. 2

[5] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo, 2021. 2

[6] Qingshan Xu and Wenbing Tao. Planar prior assisted patch-match multi-view stereo. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):12516–12523, Apr. 2020. 2

[7] Qingshan Xu and Wenbing Tao. Multi-scale geometric consistency guided multi-view stereo. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5478–5487, 2019. 2

[8] Andreas Kuhn, Christian Sormann, Mattia Rossi, Oliver Erdler, and Friedrich Fraundorfer. Deepc-mvs: Deep confidence prediction for multi-view stereo reconstruction. In *2020 International Conference on 3D Vision (3DV)*, pages 404–413, 2020. 2

[9] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2

[10] Junho Jeon, Yeongyu Jung, Haejoon Kim, and Seungyong Lee. Texture map generation for 3d reconstructed scenes. *Vis. Comput.*, 32(6–8):955–965, jun 2016. 2

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017. 2, 5

[12] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2, 7

[13] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 2, 7

[14] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2, 7

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. pages 4396–4405, 06 2019. 2

[16] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2

[17] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021. 3

[18] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 3

[19] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 3, 7

[20] Hololens research mode. https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/research-mode. Accessed: 2010-09-30. 3

[21] Dan Bohus, Sean Andrist, Ashley Feniello, Nick Saw, Mihai Jalobeanu, Patrick Sweeney, Anne Loomis Thompson, and Eric Horvitz. Platform for situated intelligence, 2021. 3

[22] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 3, 5

[23] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *Proceedings of the IEEE international conference on computer vision*, pages 143–152, 2017. 4

[24] Der-Tsai Lee and Bruce J Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980. 4

[25] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999. 4

[26] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 4

[27] Y. Livnat, Han-Wei Shen, and C.R. Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84, 1996. 4

[28] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 6

[29] Ibraheem Alhashim and Peter Wonka. High quality monocular depth estimation via transfer learning, 2018. 7

[30] Alican Mertan, Damien Jade Duff, and Gozde Unal. Single image depth estimation: An overview. *Digital Signal Processing*, 123:103441, apr 2022. 7

[31] Zunjie Zhu, Zhefeng Xu, Ruolin Chen, Tingyu Wang, Can Wang, Chenggang Yan, and Feng Xu. Fastfusion: Real-time indoor scene reconstruction with fast sensor motion. *Remote Sensing*, 14(15), 2022. 7

[32] Qian-Yi Zhou and Vladlen Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. 33(4), 2014. 7

[33] A. Al-Rawabdeh, F. He, and A. Habib. Automated feature-based down-sampling approaches for fine registration of irregular point clouds. *Remote Sensing*, 12(7), 2020. 7

[34] Antoni Rosinol, John J Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. *arXiv preprint arXiv:2210.13641*, 2022. 7