Feature-less Stitching of Cylindrical Tunnel

Ramanpreet Singh Pahwa¹, Wei Kiat Leong², Shaohui Foong³, Karianto Leman¹, Minh N. Do⁴

Abstract—Traditional image stitching algorithms use transforms such as homography to combine different views of a scene. They usually work well when the scene is planar or when the camera is only rotated, keeping its position static. This severely limits their use in real world scenarios where an unmanned aerial vehicle (UAV) potentially hovers around and flies in an enclosed area while rotating to capture a video sequence. We utilize known scene geometry along with recorded camera trajectory to create cylindrical images captured in a given environment such as a tunnel where the camera rotates around its center. The captured images of the inner surface of the given scene are combined to create a composite panoramic image that is textured onto a 3D geometrical object in Unity graphical engine to create an immersive environment for end users.

Index Terms—Image Stitching, Cylindrical Projection, Unity Simulation

I. INTRODUCTION

Aging infrastructure is becoming an increasing concern in the developed countries. There is a growing need for automatic or user-assisted assessment, diagnosis and fault detection of old structures such as sewage tunnels, bridges and roof-tops [1], [2]. Some of these structures may also be inaccessible or too dangerous for human inspection. For example, manual inspection of deep tunnel networks is an extremely challenging and risky task due to the inaccessibility and potentially hazardous environment contained in these tunnels. Due to the health risks involved, UAVs coupled with scene understanding techniques [3], [4] provide a perfect choice as they are compact and can be automated or controlled by a user to remotely capture the necessary information.

This paper builds towards imaging and inspection of the Deep Tunnel Sewerage System (DTSS). DTSS is a massive integrated project currently being developed by the Public Utilities Board (PUB) in Singapore to meet the country's long-term clean water needs through the collection, treatment, reclamation and disposal of used water from industries, homes and businesses [5]. These DTSS tunnels are covered with a corrosion protection lining (CPL) for protection. This paper aims towards automatically stitching the images collected by the UAV into a cylindrical panoramic view of the tunnel and render the tunnel in 3D to inspect the physical conditions of the CPL as well as the structural integrity of the tunnel as a whole.

While UAVs provide a viable alternative for remote assessment of deep tunnels as they are unaffected by debris and sewage flow, they are primarily designed for high altitude aerial imagery and are not appropriate for short range detailed imaging of tunnel surfaces. An alternative is to attach a 360° camera in front of a UAV and capture the panoramic view of the tunnel. However, these 360° images have low resolution

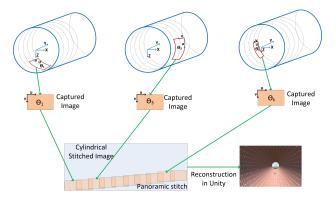


Fig. 1: The UAV is assumed to be moving horizontally during the capturing process. This constant horizontal motion of the UAV coupled with the rotating camera results in a panoramic spiral image.

that are not suitable for fault detection. Moreover, most of these cameras are too heavy and/or consist of odd shapes that render them difficult to attach to a UAV. Instead, we can use a lightweight and high resolution camera. After performing calibration [6]–[8], the camera can be mounted on a UAV. The camera rotates around the shaft of the UAV while it moves forward in a tunnel, in turn providing us with spiral-like images.

This paper presents a framework where we simulate a UAV to fly through a cylindrical sewage tunnel as shown in Fig. 1. We record the trajectory of the UAV moving in the scene and and integrate it with 2D color images captured by a rotating camera to stitch the images using cylindrical projection. Thereafter, the stitched cylindrical images are textured on a tunnel-like 3D object and displayed in Unity [9] to assist users in visualization, remote inspection, and fault detection of these tunnels.

In particular, we make the following contributions:

- A novel 360° revolving camera system is simulated in a virtual environment which, coupled with the maneuverability of the UAV, allows capturing high definition images of the tunnel surface efficiently in a spiraling motion.
- We develop a mathematical framework to combine recorded camera trajectory with known scene geometry of a tunnel to automatically combine the images captured and create a "panoramic" view of the tunnel.
- A geometrical visualization framework is developed using Unity to assist the users in visualizing the tunnel and room-like geometry for inspection and fault detection.

II. RELATED WORK

Image Stitching: A lot of work has been done in the computer vision [10]–[13] and photogrammetry community [14]–[17] to perform image stitching. Traditional image stitching techniques rely on an underlying transform, usually a 3×3 affine or homographic matrix that maps pixels from one coordinate frame to another. Typical image stitching techniques such as AutoStitch [18] assume the camera's location to be static, i.e. a pure camera rotation between captured images, or the captured scene to be roughly planar. In our panoramic spiral imaging system, the camera both rotates and translates while capturing the scene. This translation is often not negligible as compared to the distance of the tunnel surface to the camera. Moreover, the planar assumption of the scene is invalid for us since we capture images of a cylindrical tunnel. Dornaika and Chung [19] proposed a heuristic approach of piecewise planar patches to overcome this issue. Other recent methods [20]–[22] propose a different strategy of aligning the images partially in order to find a good seam to stitch different images in the presence of parallax. However, these methods rely heavily on reliable feature detection and matching, which might be difficult in the tunnel environment. Furthermore, these methods do not exploit known geometry of the scene. **Structure-from-Motion:** Structure-from-Motion (SfM) refers to the recovery of 3D structure of the scene from given images. A widely known application of SfM is where ancient Rome is

reconstructed using Internet images [23], [24]. SfM has made tremendous progress in the recent years [25], [26]. Since, we use a simulated environment for this work, we use the groundtruth camera pose recorded while capturing the dataset.

360° cameras: MIT Technology review [27] identified 360° cameras as one of the top ten breakthrough technologies of 2017. Usually two or more spherical cameras are used to stitch images and videos in real-time [28], [29] or offline [30], [31]. However, the current technology suffers from extensive motion blur and low resolution. Expensive and dedicated hardware is required to capture and post-process high resolution video [32]. To our knowledge, the best resolution for 360° video capture is provided by VIRB 360 [28] at 3,840 pixels per 360° resolution at 30 fps. This is not sufficient for anomaly inspection and identifying faults in sewage tunnel linings. We intend to use a light-weight GoPro camera [33] for our future data collection which will provide upto 7,500 pixels per 360° resolution at 60 fps. This will result in twice the resolution and twice the frame rate compared to the best solution available currently in the market.

III. CYLINDRICAL PROJECTION

In this section, we set up the cylindrical projection that models spiral imaging of tunnel surfaces. As shown in Fig. 2, a generic 2D pixel of an acquired image, $[u,v]^{\mathsf{T}}$, can be projected to a 3D point $x = [x, y, z]^T$ using a camera's

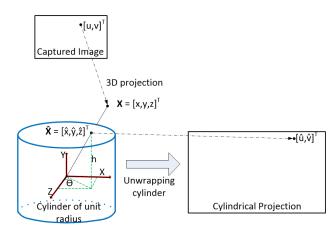


Fig. 2: Cylindrical projection - A 3D point is projected onto a cylinder with unit radius to obtain cylindrical coordinates. These cylindrical coordinates are then eventually flattened out onto a planar image.

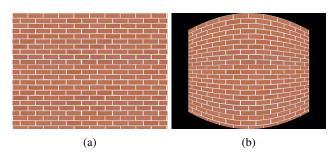


Fig. 3: (a) A sample brick image. (b) Image obtained after cylindrical projection.

intrinsic projection parameters - focal length, f, and optical center $([c_x, c_y]^{\mathsf{T}})$ - as follows:

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \lambda \boldsymbol{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} \frac{u - c_x}{f} \\ \frac{v - c_y}{f} \\ 1 \end{bmatrix}, \tag{1}$$

where K represents the internal calibration matrix of the camera and λ refers to the pixel's depth. This 3D point is projected onto a unit cylinder as follows:

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \frac{1}{\sqrt{x^2 + z^2}} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sin \theta \\ h \\ \cos \theta \end{bmatrix}, \tag{2}$$

where, angle, θ , and height, h, are two parameters required to represent a 3D point lying on a unit cylinder. The unit cylinder can be unwrapped onto a planar image as shown in Fig. 2 as follows:

$$\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = \begin{bmatrix} f\theta + c_x \\ fh + c_y \end{bmatrix}.$$
 (3)

An example of cylindrical projection of a 2D planar image captured by a camera is shown in Fig. 3.

IV. IMAGE STITCHING

We use Unity to generate our simulated dataset. A cylindrical hollow cylinder, resembling a tunnel, is created in Blender [34] and a texture is applied to it in Unity. The camera is placed near the center of tunnel and is rotated a certain degree every few milliseconds. The benefits of taking an image after the camera rotates a certain pre-defined rotation is to avoid issues such as motion blur and rolling shutter noise that would occur if we capture a video sequence in real environment.

As we use a simulated environment, we can generate the ground-truth measurements for the movement of the camera in the tunnel. We consider these rotation and translation measurements of the camera as the final camera pose for our framework. The image capturing and stitching design is shown in Fig. 4.

Using the camera pose enables us to project and transform the 2D pixel information per image into world coordinate frame. Let X_w represent the 3D points, per frame, in world coordinate frame. A global i^{th} point $x_w = X_w^i$ lying on the cylindrical tube of radius r can be represented by two parameters - θ^i and h^i as follows:

$$\boldsymbol{X}_{w}^{i} = \boldsymbol{x}_{w} = \begin{bmatrix} r \sin \theta^{i} \\ h^{i} \\ r \cos \theta^{i} \end{bmatrix}$$
 (4)

Let X_c represent the 3D points in camera frame of reference for every image captured by the rotating camera. Every i^{th} pixel (= $[u^i, v^i]^T$) can be projected onto 3D as:

$$\boldsymbol{X}_{c}^{i} = \boldsymbol{x}_{c} = \frac{z_{c}^{i}}{f} \begin{bmatrix} u^{i} - c_{x} \\ v^{i} - c_{y} \\ f \end{bmatrix}$$
 (5)

where z_c^i refers to the depth of i^{th} pixel which is unknown. X_c and X_w are related by:

$$X_w = RX_c + t \tag{6}$$

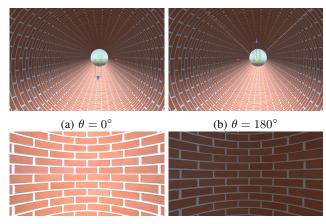
where ${m R}$ is a 3×3 orthonormal rotation matrix and ${m t}=[t_x,t_y,t_z]^{\sf T}$ represents the translation of the UAV in world coordinate frame. Initially the UAV is assumed to be at the center of the tunnel (${m t}=\vec{0}$). As the UAV moves horizontally across the tunnel, t_y increases. t_x and t_z denote the deviation of the UAV from center of the tunnel in \vec{x} and \vec{z} directions respectively.

$$m{R} = egin{bmatrix} r_{11} & r_{12} & r_{13} \ r_{21} & r_{22} & r_{23} \ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad m{t} = egin{bmatrix} t_x \ t_y \ t_z \end{bmatrix}$$

For every i^{th} pixel, we obtain three equations for three parameters - $\{z_c^i, \theta^i, h^i\}$. The two equations involving θ can be deterministically solved as follows:

$$r \sin \theta^{i} = (\mathbf{R}\mathbf{x}_{c})(1) + t_{x}$$

$$= \frac{z_{c}^{i}}{f} (r_{11}(u^{i} - c_{x}) + r_{12}(v^{i} - c_{y}) + r_{13}f) + t_{x}$$
(7)



(c) Image captured at $\theta = 0^{\circ}$ (d) Image captured at $\theta = 180^{\circ}$

Fig. 4: Top row shows the unity simulation when camera is rotated by 0° and 180° with respect to y axis (green arrow). Bottom row displays the images captured by the virtual camera at these two angles respectively.

$$r\cos\theta^{i} = (\mathbf{R}x_{c})(3) + t_{z}$$

$$= \frac{z_{c}^{i}}{f}(r_{31}(u^{i} - c_{x}) + r_{32}(v^{i} - c_{y}) + r_{33}f) + t_{z}$$
(8)

where $(\mathbf{R}\mathbf{x}_c)(1)$ and $(\mathbf{R}\mathbf{x}_c)(3)$ represent the first and third element of the column vector $\mathbf{R}\mathbf{x}_c$. Let

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \frac{1}{f} \begin{bmatrix} (r_{11}(u - c_x) + r_{12}(v - c_y) + r_{13}f) \\ (r_{21}(u - c_x) + r_{22}(v - c_y) + r_{23}f) \\ (r_{31}(u - c_x) + r_{32}(v - c_y) + r_{33}f) \end{bmatrix}$$
(9)

Solving for z_c using Eqs. (6,7,8,9), we obtain:

$$r^{2} = z_{c}^{2}(c_{1}^{2} + c_{3}^{2}) + 2z_{c}(c_{1}t_{x} + c_{3}t_{z}) + t_{x}^{2} + t_{z}^{2}$$
 (10)

Thus.

$$z_{c} = \frac{-(c_{1}t_{x} + c_{3}t_{z}) \pm \sqrt{(c_{1}t_{x} + c_{3}t_{z}) - (c_{1}^{2} + c_{3}^{2})(t_{x}^{2} + t_{z}^{2} - r^{2})}}{(c_{1}^{2} + c_{3}^{2})}$$

$$(11)$$

Thereafter, we can compute θ by finding an intersection of the following two solutions for Eqs. (7,8):

$$\theta = \left\{ \arcsin\left(\frac{c_1 z_c + t_x}{r}\right), \pi - \arcsin\left(\frac{c_1 z_c + t_x}{r}\right) \right\} \cap \left\{ \arccos\left(\frac{c_3 z_c + t_z}{r}\right), 2\pi - \arccos\left(\frac{c_3 z_c + t_z}{r}\right) \right\}$$
(12)

Thereafter, we can also estimate the y component of x_c by:

$$h = c_2 z_c + t_y \tag{13}$$

Once we compute the world coordinate of every pixel's location, we can obtain the cylindrical projection for it as described in Sec. III.

V. EXPERIMENTAL RESULTS

In this section, we perform synthetic experiments and compare our results in both noiseless and noisy scenarios. We also discuss our Unity framework and display a few examples of the rendered cylindrical scene in Unity for visualization.

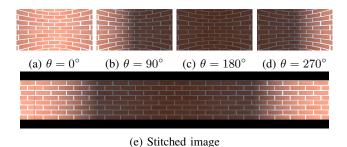


Fig. 5: Top row displays the images captured from unity when the camera is rotated from 0 to 360 degrees. Bottom row displays the stitched image obtained after our stitching process.

Unity is a game engine used mainly to create platform independent video game applications. However, it also provides us with a good set of tools to provide an immersive 3D visualization for various purposes. The cylinder of radius, r=3m, is positioned such that the geometrical center of the cylinder is located at $[0,0,0]^{\mathsf{T}}$. We texture-mapped a brick wall with a downward facing light source onto the inner face of the cylinder for visualization purposes. The light source is fixed to look vertically down for our experimental evaluation. Hence, images captured around 0° rotation are brightly lit while the images captured around 180° appear dark due to lack of illumination. A few images captured by the virtual camera are shown in Fig. 4(c-d), 5(a-d). Let us denote the cylindrical image to be synthesized as X_{vv}^{2D} .

Stationary Camera Positioned at Center: In our first experiment, we positioned the camera at the center of a cylindrical tunnel. The camera is held stationary throughout this experiment and only rotated rotated by $\beta=30^\circ$ per frame and images are captured consequently. An example of this process is shown in Fig. 5. It takes 12 images to complete the full 360° rotation. Thereafter, the images are stitched together as described in Sec. IV. In this experiment:

$$\mathbf{R} = \begin{bmatrix} \cos(\beta k) & 0 & \sin(\beta k) \\ 0 & 1 & 0 \\ -\sin(\beta k) & 0 & \cos(\beta k) \end{bmatrix}; \mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \forall k = \{0, 1, \dots, 11\}$$
(14)

For each image captured, we use Eqs. 12 and 13 to project each pixel onto the cylindrical stitched image \boldsymbol{X}_w^{2D} . However, performing this "forward warping" may leave some holes in the stitched image. Thus, instead of performing forward warping, we use the four corners of each image to obtain the forward warped boundary. Thereafter, for every pixel inside this boundary of \boldsymbol{X}_w^{2D} , we perform "inverse warping" to obtain its pixel location and intensity information in \boldsymbol{X}_c^i . The fully stitched image is shown in Fig. 5(e). We observe that the images align perfectly with each other and all the "curved" bricks are straightened after performing cylindrical projection.

Stationary Camera Positioned off Center: Under ideal conditions, the camera should be positioned at the center of cylindrical tunnel. However, in reality this will not

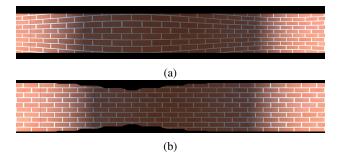
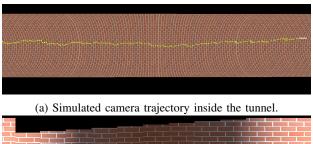


Fig. 6: (a) Stitched cylindrical image without accounting for camera location off center. (b) Stitched cylindrical image after accounting for camera's location off center.

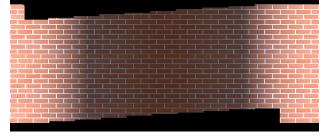
happen as UAVs tend to hover around and might have difficulty maneuvering to the center of the cylinder before each image is captured. We move the camera off center, i.e. $\boldsymbol{t} = [0.5m, 0m, 0.5m]^{\mathsf{T}}$. The camera is held stationary throughout this experiment and only rotated by $\beta = 30^{\circ}$ per frame and images are captured consequently.

A cylindrical image is a 360° view of the scene around the camera flattened on a planar image. Thus, even though the camera is off center, we can still view what the cylindrical projection of the scene looks like assuming the UAV's current position to be the center of the tunnel as shown in Fig. 6(a). While the stitching is perfect, the straight lines (bricks) are no longer straight and we can see the zoom in and zoom out effect when the camera is far or near to the tunnel boundary respectively. We use Eqs. 12 and 13 to synthesize what each image would look like if the camera was positioned at the center of the tunnel. The fully stitched image is shown in Fig. 6(b). We observe that the images align perfectly with each other and all the "curved" bricks are straightened after performing cylindrical projection.

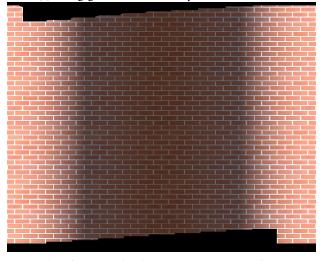
Freely moving camera in a simulated tunnel: In our last experiment, we aim to simulate UAV movements in realworld conditions. A UAV is expected to suffer from jitters and sideways movements while it tries to balance itself and move forward in the tunnel. This means that the camera's movement and rotation per image given by IMU may be unreliable for our stitching purposes. We simulate a tunnel of radius 3 m. We initially positioned the camera at the center of the cylindrical tunnel. Our baseline movement and rotation in the three orthogonal directions are $t = [0, 0.10, 0]^{\mathsf{T}}$ and $r = [0, 0.524, 0]^{\mathsf{T}}$ respectively. This implies that we expect the ideal movements in the tunnel to be 10 cm horizontally forward (v direction) with a 30° rotation across v axis. We add Gaussian noise with zero mean and standard deviation of [2, 2, 3]^T to our translation and Gaussian noise with zero mean and standard deviation of 2° to camera rotation per image. We run the simulation till the camera completes ten rotations and record the groundtruth translation and rotation of the camera per frame. In the baseline stitch, we assume that the camera does not suffer from any jitter or sideways movement and blindly trust the initially planned rotation



(b) Using baseline pre-estimated camera pose.



(c) Using groundtruth camera pose, four rotations.



(d) Using groundtruth camera pose, ten rotations.

Fig. 7: We add random jitter and movement to the camera after each image capture. Using baseline rotation and translation leads to an extremely inaccurate and incoherent stitch.

and translation per frame. In our second approach, we use groundtruth measurements to perform image stitching. The results of the baseline and groundtruth stitch are shown in Fig. 7. The baseline stitch, understandably, fails to provide us with a good stitch of the simulated tunnel. This validates our framework and showcases that we can obtain good results for stitching and visualizing long underground tunnels as long as we are able to obtain a good estimation for camera pose per image. This cylindrical projection stitch can be wrapped around a cylinder in the Unity engine to provide an immersive

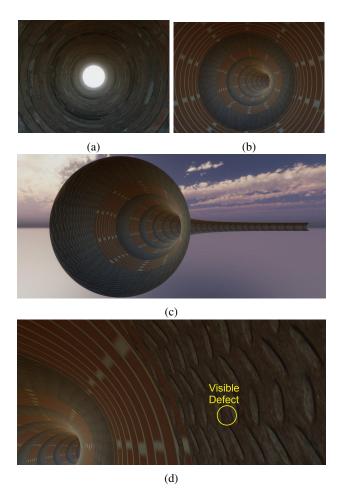


Fig. 8: A hollow cylinder is created in Blender and UV mapped. This cylinder is imported in unity to create both (a) straight and (b,c) curved tunnels based on the user requirements. (d) The user can move freely inside the tunnel and move closer to the boundaries to carefully inspect for any damage in the tunnel.

3D display of the panoramic view for users.

Visualization in Unity: We display the cylindrical images of the scene rendered as described in previous sections in Unity. We envision our system as a "fly-through" of the scene. The user controls the camera position using keyboard and mouse and just like a first-player shooter game, can float around in the 3D scene freely. This enables the user to not only view the rendered tunnel images but also move closer to the areas where the user suspects faults in the tunnel. Fig. 8 shows our setup for straight and curved tunnels. The user provides a text file with the curve of tunnel and the cylinders for the curved tunnel are rendered when the user starts the application. Video demos for Unity simulation and user inspection can be seen here and here.

VI. CONCLUSION

We presented a simple and accurate system to capture images of a given scene using a spirally moving camera system and display the panoramic stitched images in unity for an interactive 3D display. The presented method excels in scenes where prior geometrical information is available. This allows us to project the images in 3D and warp them onto a unit cylinder to obtain unit cylindrical images of the scene. This approach can be easily extended to other commonly found geometries such as underpasses, rooms, and train tracks.

In future, we plan to extract the geometrical information such as tunnel radius, curved angle of pipes automatically using depth sensors mounted on the UAV. We also intend to test our framework on real dataset in the future.

ACKNOWLEDGMENT

This research grant is supported by the Singapore National Research Foundation under its Environmental & Water Technologies Strategic Research Programme and administered by the PUB, Singapore's National Water Agency.

REFERENCES

- S. Samiappan, G. Turnage, L. Hathcock, L. Casagrande, P. Stinson, and R. Moorhead, "Using unmanned aerial vehicles for high-resolution remote sensing to map invasive phragmites australis in coastal wetlands," *International Journal of Remote Sensing*, pp. 1–19, 2016.
- [2] N. Metni and T. Hamel, "A UAV for bridge inspection: Visual servoing control law with orientation limits," *Automation in Construction*, vol. 17, no. 1, pp. 3–10, 2007.
- [3] R. S. Pahwa, J. Lu, N. Jiang, T. T. Ng, and M. N. Do, "Locating 3D Object Proposals: A Depth-Based Online Approach," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 28, no. 3, pp. 626–639, March 2018.
- [4] R. S. Pahwa, T. T. Ng, and M. N. Do, "Tracking objects using 3D object proposals," in Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Dec 2017, pp. 1657–1660.
- [5] PUB, DTSS Home, Singapore's National Water Agency, Singapore, 2017. [Online]. Available: https://www.pub.gov.sg/dtss/
- [6] R. S. Pahwa, M. N. Do, T. T. Ng, and B. S. Hua, "Calibration of depth cameras using denoised depth images," in *IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 3459–3463.
- [7] R. S. Pahwa, "Depth camera calibration using depth measurements," Master's thesis, UIUC, U.S.A, 2013.
- [8] —, "3D sensing and mapping using mobile color and depth sensors," Ph.D. dissertation, UIUC, U.S.A, 2017.
- [9] Unity, *Unity Game Engine*, urlhttps://unity3d.com/, Unity Technologies, San Francisco, USA, 2017, [Online; accessed 10-Jan-2017].
- [10] J. Lu, D. Min, R. S. Pahwa, and M. N. Do, "A revisit to MRF-based depth map super-resolution and enhancement," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 985–988.
- [11] T. T. Ng, R. S. Pahwa, B. Jiamin, K. H. T. an, and R. Ramamoorthi, "From the Rendering Equation to Stratified Light Transport Inversion," *International Journal of Computer Vision (IJCV)*, vol. 96, no. 2, pp. 235–251, Jan. 2012. [Online]. Available: https://doi.org/10.1007/ s11263-011-0467-6
- [12] T. T. Ng, R. S. Pahwa, B. Jiamin, T. Q. S. Quek, and K. H. Tan, "Radiometric compensation using stratified inverses," in *IEEE International Conference on Computer Vision (ICCV)*, Sept 2009, pp. 1889–1894.
- [13] X. Chu, T. T. Ng, R. S. Pahwa, T. Q. S. Quek, and T. S. Huang, "Compressive Inverse Light Transport," *British Machine Vision Conference* (BMVC), vol. 10, no. 16, p. 27, 2011.
- [14] R. Szeliski, "Image alignment and stitching: A tutorial," Foundations and Trends in Computer Graphics and Vision, vol. 2, no. 1, pp. 1–104, 2006
- [15] A. Zomet, A. Levin, S. Peleg, and Y. Weiss, "Seamless image stitching by minimizing false edges," *IEEE Transactions on Image Processing* (TIP), vol. 15, no. 4, pp. 969–977, 2006.
- [16] A. Levin, A. Zomet, S. Peleg, and Y. Weiss, "Seamless image stitching in the gradient domain," in *European Conference on Computer Vision* (ECCV). Springer, 2004, pp. 377–389.

- [17] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision (IJCV)*, vol. 74, no. 1, pp. 59–73, 2007.
- [18] M. Brown and D. Lowe, "Autostitch," 2008.
- [19] F. Dornaika and R. Chung, "Mosaicking images with parallax," *Signal Processing: Image Communication*, vol. 19, no. 8, pp. 771–786, 2004.
- [20] J. Zaragoza, T.-J. Chin, M. S. Brown, and D. Suter, "As-projective-as-possible image stitching with moving DLT," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2339–2346.
- [21] F. Zhang and F. Liu, "Parallax-tolerant image stitching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3262–3269.
- [22] W.-Y. Lin, S. Liu, Y. Matsushita, T.-T. Ng, and L.-F. Cheong, "Smoothly varying affine stitching," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011, pp. 345–352.
- [23] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, "Building rome in a day," *Communications of the ACM*, vol. 54, no. 10, pp. 105–112, 2011.
- [24] J.-M. Frahm, F.-G. Pierre, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik et al., "Building rome on a cloudless day," in *European Conference on Computer Vision (ECCV)*. Springer, 2010, pp. 368–381.
- [25] A. Kushal, B. Self, Y. Furukawa, D. Gallup, C. Hernandez, B. Curless, and S. M. S. M. Seitz, "Photo tours," in Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT). IEEE, 2012, pp. 57–64.
- [26] C. Wu, "Towards linear-time incremental structure from motion," in International Conference on 3DTV. IEEE, 2013, pp. 127–134.
- [27] E. Woyke, The 360-Degree Selfie: 10 Breakthrough Technologies 2017 MIT Technology Review, https://www.technologyreview.com/s/603496/10-breakthrough-technologies-2017-the-360-degree-selfie/, MIT Technology Review, 2017, [Online; accessed 10-April-2018].
- [28] Garmin Ltd., VIRB 360 Cameras Products Garmin Singapore — Home, http://www.garmin.com.sg/products/cameras/virb-360, 2017, [Online; accessed 10-April-2018].
- [29] Insta360, Insta360 ONE A camera crew in your hand, https://www.insta360.com/product/insta360-one, 2018, [Online; accessed 10-Apr-2018].
- [30] Xiaomi Inc., Mi Sphere Camera Kit 360 Degree Panoramic Camera, http://www.mi.com/us/mj-panorama-camera/, 2018, [Online; accessed 10-Apr-2018].
- [31] Rylo Inc., Rylo The powerful little 360 camera, https://www.rylo.com/, 2018, [Online; accessed 10-Apr-2018].
- [32] S. Kasahara and J. Rekimoto, "Jackin head: An immersive humanhuman telepresence system," in SIGGRAPH Asia 2015 Emerging Technologies, ser. SA '15. New York, NY, USA: ACM, 2015, pp. 14:1– 14:3. [Online]. Available: http://doi.acm.org/10.1145/2818466.2818486
- [33] GoPro, Inc., GoPro Official Website Capture + share your world - HERO4, https://shop.gopro.com/APAC/cameras/, 2017, [Online; accessed 10-Jan-2017].
- [34] B. O. Community, Blender A 3D Modelling and Rendering Package, Blender Foundation, Blender Institute, Amsterdam, 2016. [Online]. Available: http://www.blender.org