ACTA IMEKO
ISSN: 2221-870X
June 2022, Volume 11, Number 2, 1 - 8

Omnidirectional camera pose estimation and projective texture mapping for photorealistic 3D virtual reality experiences

Alessandro Luchetti[1], Matteo Zanetti[1], Denis Kalkofen[2], Mariolino De Cecco[1]

[1] *Department of Industrial Engineering, University of Trento, Sommarive 9, 38123 Trento, Italy*
[2] *Institute of Computer Graphics and Vision, Graz University of Technology, Rechbauerstraße 12, 8010 Graz, Austria*

ABSTRACT
Modern applications in virtual reality require a high level of fruition of the environment as if it was real. In applications that have to deal with real scenarios, it is important to acquire both its three-dimensional (3D) structure and details to enable the users to achieve good immersive experiences. The purpose of this paper is to illustrate a method to obtain a mesh with high quality texture combining a raw 3D mesh model of the environment and 360° images. The main outcome is a mesh with a high level of photorealistic details. This enables both a good depth perception thanks to the mesh model and high visualization quality thanks to the 2D resolution of modern omnidirectional cameras. The fundamental step to reach this goal is the correct alignment between the 360° camera and the 3D mesh model. For this reason, we propose a method that embodies two steps: 1) find the 360° cameras pose within the current 3D environment; 2) project the high-quality 360° image on top of the mesh. After the method description, we outline its validation in two virtual reality scenarios, a mine and city environment, respectively, which allows us to compare the achieved results with the ground truth.

Section: RESEARCH PAPER

Keywords: Omnidirectional cameras; mesh reconstruction; camera pose estimation; optimization; enhanced comprehension

Citation: Alessandro Luchetti, Matteo Zanetti, Denis Kalkofen, Mariolino De Cecco, Omnidirectional camera pose estimation and projective texture mapping for photorealistic 3D VR experiences, Acta IMEKO, vol. 11, no. 2, article 24, June 2022, identifier: IMEKO-ACTA-11 (2022)-02-24

Section Editor: Alfredo Cigada, Politecnico di Milano, Italy

Received May 26, 2021; **In final form** March 21, 2022; **Published** June 2022

Funding: This work was developed inside the European project MiReBooks: Mixed Reality Handbooks for Mining Education, a project funded by EIT Raw Materials.

Corresponding author: Alessandro Luchetti, e-mail: alessandro.luchetti@unitn.it

1. INTRODUCTION

Media acquired by 360°cameras (also known as omnidirectional, spherical, or panoramic cameras) is becoming increasingly important to many applications. Compared to conventional cameras, images taken by 360° cameras offer a larger field of view, which is why they are traditionally useful to applications that derive their state from information about the environment. Examples include robot localization, navigation, and visual servoing [1]. However, omnidirectional cameras have recently also become an essential tool for content creation in Virtual Reality (VR) applications because spherical photographs and videos can provide a high level of realism. For example, applications for real estate agents already make use of omnidirectional images and video data within a VR head mounted display to improve the realism of virtual customer inspections and research domains span widely from 360° tourism [2] to education in 360° classrooms [3].

VR applications using omnidirectional media allow their users to change the view within the boundaries of a 360° image that has been captured at a specific Point of Interest (POI). Thus, VR users are commonly restricted to head rotations only while translations require transitioning into a 360° image that has been captured at a different POI [4]. Thus, motion parallax is missing in VR applications, which use omnidirectional data. Furthermore, view transitions are limited to where omnidirectional images or videos exist. These shortcomings limit the benefit of omnidirectional media in VR. For example, the missing 3D information restricts the usage of advanced exploration techniques [5], [6] and the missing motion parallax can cause visual discomfort [7].

To overcome these limitations, we propose combining omnidirectional photorealistic image data with its corresponding 3D representation. Since 3D reconstructions commonly suffer from poor color representations, we apply projective texture mapping of omnidirectional images. Our approach supports photorealistic image fidelity at the POIs and motion parallax at viewpoints nearby. To enable projective texture mapping of 360° image data, we present an approach for omnidirectional camera pose estimation that automatically finds the position and orientation of the 360° camera relative to the 3D representation of the environment.

To put our work in context, we first outline related work in Section 2, before we describe our approaches to omnidirectional camera pose estimation and projective texture mapping in Section 3. We evaluate our system in Section 4 and discuss possible directions for future work in Section 5.

2. RELATED WORK

Camera pose detection has always been a key problem in computer vision. For example, Makadia et al. [8] proposed a method useful for the alignment of large rotations with potential impact on 3D shape alignment to estimate the rotation directly from images defined on the sphere and without correspondence. Unfortunately, this approach is quite resistant only to small translations of the camera [9]. Another work [10] addresses the problem of camera pose recovery from spherical panoramas using pairwise essential matrices. In this case, the exact position of each panorama was an important step to ensure the consistency of visual information about a database of geo-referenced images. Here the pose recovery works with a two-stage algorithm for rotations and after for translations with a bad result if the camera starting pose is very far from the correct one.

The above-mentioned problems have been overcome by our method because it works also for large variations of translation as well as rotations. Also Levin et al. present in [11] a method to compute camera pose from a sequence of spherical images through the use of an essential matrix for initial pairwise geometry. Differently from our work and the work of [10], they also use a rough estimate of the camera path as an additional system input to calculate camera positions.

An example of generating a texture map of a 3D model with 2D high-quality images is given in [12]. In particular, it is a specific application in the e-commerce presentation of shoes. It consists of a texture mapping technique that comprises several phases: mesh partitioning, mesh parameterization and packing, texture transferring, and texture correction and optimization. In particular, in the texture transferring step, each mesh is allocated to a front image, and all meshes that use the same front image are put in a group. Finally, the pixels from the front image corresponding to the 3D mesh are extracted. Differently, our method uses only a spherical image to recreate the high-resolution 3D model by projecting each pixel of the image from the correct camera pose previously found. The obtained results are faster and good if the user's field of view rotates without large displacements with respect to the camera pose.

A similar approach but for another application related to realizing surveying tasks in architectural, archaeological, and cultural landscapes conservation is provided by Abmayr et al. [13]. They developed a laser scanner, which offers high accuracy measurements of object surfaces, combined with a panoramic color camera to achieve precise and accurate monitoring of the actual environment employing colored point clouds. The camera rotates according to the same tripod as the laser scanner. Many similarities with the method described in the present article can be found. The main difference resides in the use of a single 360° camera instead of a rotating unit, the use of an automatic pose estimation method instead to use the same tripod for laser scanner and camera during the acquisition process. Our method is faster, and the 3D model reconstruction can be more complete because it doesn't need to be at a fixed distance from the camera during the scanning process. This aspect becomes more important if it is necessary to reconstruct a high-resolution model with different cameras from unknown positions.

Finally, an interesting study was provided by Teo et al. [14], where, in the context of remote collaboration, helpers shared 360° live videos or 3D virtual reconstructions of their surroundings from different places to work together with local workers. The results showed that participants preferred having both 360° and 3D modes, as it provides variation in controls and features from different perspectives. Our work provides a combination of a 360° live video and 3D virtual reconstruction to combine their advantages without the need to switch between them.

3. METHOD

In this section, the localization algorithm to estimate the camera pose (i.e., its positions and orientations in the environment), and the method used to project the texture mapping on a 3D representation of the environment are explained.

3.1. Camera pose estimation

A good alignment between the virtual environment and the captured image is fundamental for the final texture projection that will be covered in the next chapter. For example, this step is necessary when an operator needs to place the camera in a predefined position and orientation. Some human errors may be made during this operation and a method to find an accurate camera pose is necessary. Moreover, for large distances, even a small angle or small position errors can compromise the final result.

The large-scale automatic camera pose identification algorithm has been implemented in Matlab 2019b using a ZMQ communication protocol [15] between Matlab and Unity 3D. A Particle Swarm Optimization (PSO) was used. The procedure of the camera pose estimation is shown in Figure 1.

Starting from the reconstructed 3D model with its low-quality texture but with depth information of the environment and given as input a high quality equirectangular photorealistic image taken by an omnidirectional camera, the localization algorithm finds the pose that gives a 360° image taken with a simulated camera that is as similar as possible to the input one.

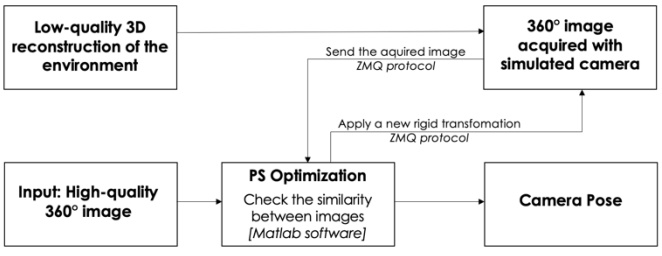


Figure 1. Schematic diagram of the camera pose detection algorithm.

In particular:

i. A new camera position is set for each iteration of the PSO algorithm.

ii. The equirectangular image corresponding to the set camera pose, at the previous step, is acquired.

iii. The algorithm checks the similarity between the new image and the input one that has to be used as a new texture for the 3D mesh; the parameters to be optimized are the translation and the Euler angles to be applied to the 3D model to generate an equirectangular image that matches the one in the input. The cost function for comparing the two equirectangular images uses the following quantities:

- The structural similarity (*SSIM*) index of the equirectangular images.
- The mean-squared error (*MSE*) between the two equirectangular images.
- *SSIM* of the approximation coefficients ($SSIM_A$) of level 1 of the wavelet decomposition.
- *SSIM* of the horizontal detail coefficients ($SSIM_H$) of level 1 of the wavelet decomposition.
- *SSIM* of the vertical detail coefficients ($SSIM_V$) of level 1 of the wavelet decomposition.
- *SSIM* of the diagonal detail coefficients ($SSIM_D$) of level 1 of the wavelet decomposition.

The final cost function *C* obtained by adding the above-mentioned quantities is:

$$C = SSIM + MSE + SSIM_A + SSIM_H + SSIM_V + SSIM_D. \quad (1)$$

The *MSE* represents the cumulative squared error between two images *x(i,j)* and *y(i,j)*:

$$MSE(x,y) = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}[x(m,n)-y(m,n)]^2\,, \quad (2)$$

where *M* and *N* are the number of rows and columns of *x* and *y*.

SSIM is used for measuring the similarity between two images *x* and *y* [16]. The *SSIM* Index quality assessment index is based on the computation of three terms, namely the luminance term *l*, the contrast term *c* and the structural term *s*. The overall index is a multiplicative combination of the three terms:

$$SSIM(x,y) = [l(x,y)]^{\alpha}\cdot[c(x,y)]^{\beta}\cdot[s(x,y)]^{\gamma}\,, \quad (3)$$

where:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{{\mu_x}^2 + {\mu_y}^2 + C_1}, \quad (4)$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{{\sigma_x}^2 + {\sigma_y}^2 + C_2}, \quad (5)$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}. \quad (6)$$

μ_x, μ_y, σ_x, σ_y and σ_{xy} are the local means, standard deviations, and cross-covariance for images *x* and *y*. C_1, C_2, C_3 are constants to avoid instability for image regions where the local mean or standard deviation is close to zero. Choosing $\alpha = \beta = \gamma = 1$ and $C_3 = \frac{C_2}{2}$, the index simplifies to:

$$SSIM(x,y) = \frac{(2\mu_X\mu_Y + C_1)(2\sigma_x\sigma_y + C_2)}{({\mu_x}^2 + {\mu_y}^2 + C_1)({\sigma_x}^2 + {\sigma_y}^2 + C_2)}. \quad (7)$$

iv. The PSO optimization runs until convergence, giving as output the best camera pose (translation and Euler angles) that makes the two images as similar as possible.

3.2. Texture projection

In this chapter, the method to apply the high-quality texture mapping will be described. Essentially, a merge of the high-quality 360° image with the 3D mesh is performed.

Firstly, the 3D Cartesian coordinates and colors of each 360° image's pixel were obtained by projecting the equirectangular image on the surface of a unitary radius sphere.

Given an equirectangular image with *N* rows and *M* columns, each image's pixel in 2D Cartesian coordinates *(n,m)* was transformed in spherical coordinates, computing the corresponding azimuth *a* and elevation *e*, setting the radius *R* equal to 1. The equations used for the conversion are:

$$a = -\left(\frac{m}{M} - 0.5\right)\cdot 2\,\pi\,, \quad (8)$$

$$e = -\left(\frac{n}{N} - 0.5\right)\cdot \pi\,, \quad (9)$$

$$R = 1. \quad (10)$$

Finally, the 3D Cartesian coordinates are obtained to be visualized in Matlab software like a 3D point cloud.

The mapping from spherical coordinates to 3D Cartesian coordinates is:

$$x = R\cdot\cos(e)\cdot\cos(a) \quad (11)$$

$$y = R\cdot\cos(e)\cdot\sin(a) \quad (12)$$

$$z = R\cdot\sin(\mathrm{e}). \quad (13)$$

This "spherical" point cloud was imported inside Unity and placed with the position and orientation found in the previous pose estimation step chapter.

The Raycasting technique was used: through the Ray class, it is possible to emit or "cast" rays in a 3D environment and control the resulting collisions. The rays used in Raycasting are invisible lines that have the center of the image sphere as the origin and are oriented in each pixel's direction. The important point is that these invisible lines or rays that are cast into the scene can return information about GameObjects that have been hit by the rays.

Attached to the environment's mesh as GameObject in Unity, there is a Mesh Collider to register a hit with the ray. When a ray intersects or "hits" a GameObject, the event is referred to as a RaycastHit. This hit provides details about the GameObject and where it was hit, including a reference to the GameObject's Transform, the length of the ray when it hits something, the point in the world where the hit happened.

Once the collision of each pixel is detected, their new position is saved with color properties.

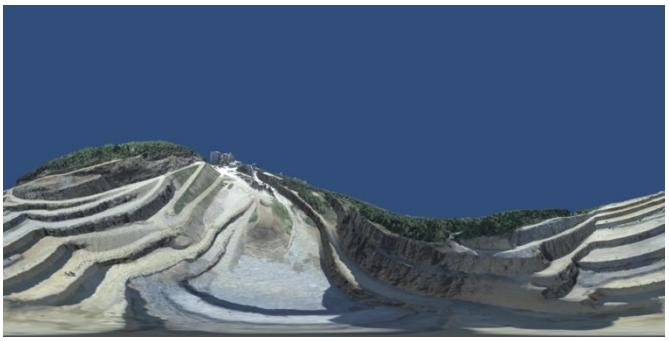
Lastly, the new point cloud was used to reconstruct a high-quality photorealistic texture, using the Screened Poisson Surface Reconstruction algorithm [17] implemented in Meshlab [18]. This algorithm is particularly useful when the model to reconstruct is very big with very fine details to be preserved. The reconstruction of the 3D model was done setting the Reconstruction Depth parameter (i.e., the maximum depth of the octree that is used to make the reconstruction) to 13. The default value of Meshlab for this parameter is 8, we increased it because in general, the higher this value is the more time will be needed for reconstitution, the more details will be preserved [17]. We did not increase it more than 13 because after 14 it is not possible to see a real change in the final result.

The Minimum Number of Samples was set to 1.5 and the Interpolation Weight to 4 as default values of Meshlab. Since the Poisson algorithm tends to "close" the reconstructed mesh, the triangles whose area was above a certain threshold were deleted to preserve the original form of the reconstructed environment.

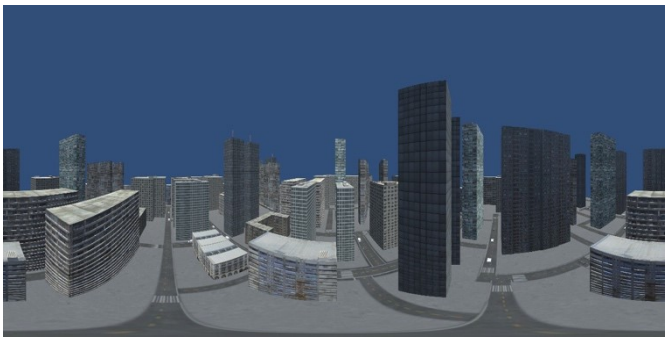
4. EVALUATION

For the validation of the camera pose localization algorithm and the high-quality texture mapping projection, a Wavefront 3D Object File (OBJ file extension) of two 3D high-quality virtual outdoor environments, one for a mine and one for a city, were imported into Unity 3D platform. An original script was also written to simulate a 360° camera. The 360° capture technique is based on Google's Omni-directional Stereo (ODS) technology using Cubemap rendering [19]. After the Cubemap is generated, it is possible to convert this Cubemap to an equirectangular map which is a projection format used by 360° video players.

After placing the simulated camera at a specific pose inside the scene of a specific scenario, a high-quality equirectangular image was acquired, Figure 2. This will be the input images whose pose has to be detected by the developed algorithm.

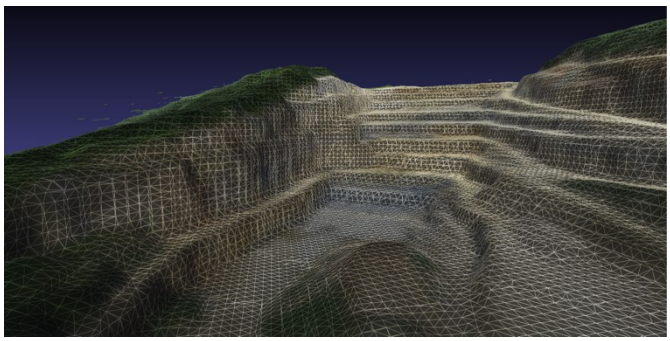


(a)

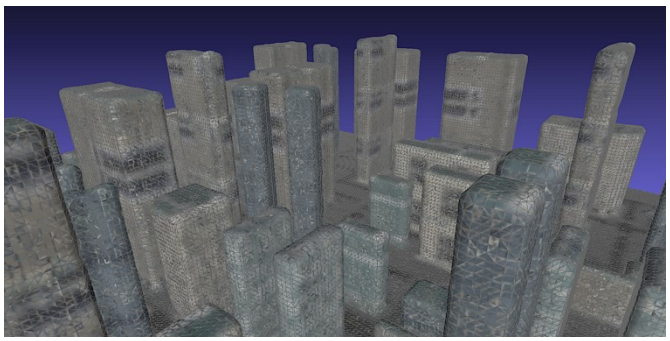


(b)

Figure 2. High-quality equirectangular images whose detection poses must be identified for a mine (a) and city (b) environments.



(a)



(b)

Figure 3. The 3D downsampled models used by the localization algorithm for a mine (a) and city (b) environments.

To simulate the acquisition of the environment through a 3D scanner, a point cloud for each analyzed environment was extracted from the 3D high-quality models using the Cloud Compare software [20]. These point clouds were downsampled to simulate a 3D model with less detail than the input model, and new reconstructions were performed in MeshLab [18] to obtain new low-quality 3D models, Figure 3. New scenes were then recreated in Unity with the downsampled 3D models.

Figure 4 shows the schematic diagram of our camera pose detection algorithm proposed in Figure 1 applied to the specific example of the mine environment.

The input omnidirectional image has a resolution of 4096 × 2048 pixels. However, to improve the calculation time speed, the comparison between images is done by downsampling them to 256 × 128 pixels for both the analyzed environments. The bounding box dimensions of the scenario with the mine are 113 m × 169 m × 37 m for the x, y, z coordinates, respectively. Instead, the dimensions of the city environment are 440 m × 100 m × 435 m.

The same analysis was done for both environments using the same approach and shifting the camera pose by the same values. Table 1 shows the position and orientation for 10 random trials.

The initial starting position was set to the origin (0, 0, 0) with null rotations for each trial. The research limits were set to ± 20 m for translations and ± 80° for rotations.

By default, Unity applies the following rotation order: Extrinsic Rotation around the z-axis (γ), then around the x-axis (α), and finally around the y-axis (β).

The average time spent by the PSO algorithm is around 20 minutes. The tests were run on a PC with an Intel i7-9700KF processor and 64 GB of RAM.

For each of the 10 trials of Table 1, the PSO algorithm has been run changing 5 times the numbers of generations, i.e., 200,

Table 1. Camera poses chosen for 10 trials (ground truth).

Trial	*x* (m)	*y* (m)	*z* (m)	*α* (°)	*β* (°)	*γ* (°)
1	-4.00	10.00	15.00	10.00	15.00	18.00
2	5.00	-2.00	5.00	10.00	-60.00	1.00
3	-8.00	5.00	-6.00	30.00	45.00	15.00
4	2.00	-7.00	15.00	-10.00	-45.00	-20.00
5	10.00	10.00	10.00	20.00	-15.00	5.00
6	0.00	15.00	8.00	25.00	-15.00	5.00
7	-5.00	2.00	-5.00	-10.00	60.00	-1.00
8	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00
9	-15.00	10.00	10.00	40.00	70.00	40.00
10	-19.00	19.00	-19.00	2.00	80.00	-5.00

250, 300, 350, 400, keeping the number of particles fixed to 100, and 5 times changing the number of particles, i.e. 60, 70, 80, 90, 100, keeping the number of generation fixed to 400. The number of generations and particles was changed to force the algorithm to increase variability.

To compute the error in pose detection, we decided to separate the translation and the rotation part. The translation error is computed by performing the Euclidean distance between the position of the camera found by the PSO algorithm and the ground truth. For what concerns the rotations, firstly, the rotations found by the optimization process and the ground truth were decomposed in Axis and Angle notation. Consequentially, the error, in the case of rotation, has 2 terms: the error in the axis orientation with respect to the ground truth and the amount of rotation around such axis.

Figure 5 shows the cost function score for the various error components explained above, while Figure 6 shows the three possible couple combinations of the error components with respect to the final score optimization value.

As can be noticed, sometimes, a higher score of the cost function at the end of the optimization does not mean that an incorrect pose was found. This fact is probably due to the mesh reconstruction process. Indeed, after this process, there could be portions of the environment that can be less accurate compared to the real model. For this reason, the meaning of the final reached score values is not absolute or easily comparable considering different camera poses.

This generates the need to quantify the accuracy of the camera localization measurement within a scene.

Despite the uncertainty concerning the accuracy in the pose found by the algorithm with respect to the final cost function score, Figure 5 and Figure 6 show that, for the mine environment, a score below 1.6 means that, for the trial performed, the error in translation is below 0.7 m, the difference in the amount of rotation is below 1°, and the difference in the rotation axis orientation is below 2°. For the city environment, the same amount of errors corresponds to a cost function score of 2. The score is higher because the city environment is a scenario with much more detail than a mine. Many of these details, through initial downsampling, are lost and the initial reconstructed mesh is much less detailed, as can be seen in Figure 3b. The final score, therefore, which measures the similarity between the input high-quality equirectangular image and that obtained from this low-quality model, turns out to be higher. However, the errors, especially those related to rotations (Figure 5b and Figure 5c), are lower for the city environment even at high levels of the cost function score because the environment is more diverse. Because of this relationship of the cost function threshold from the level of detail of the reconstructed 3D model,

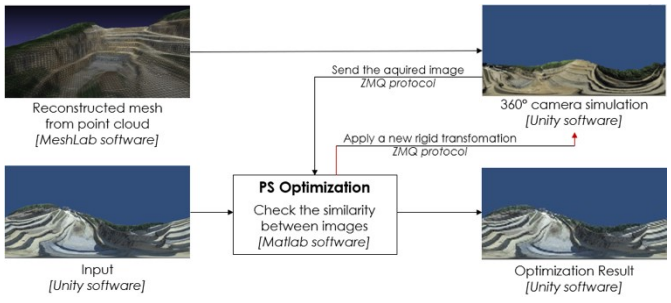
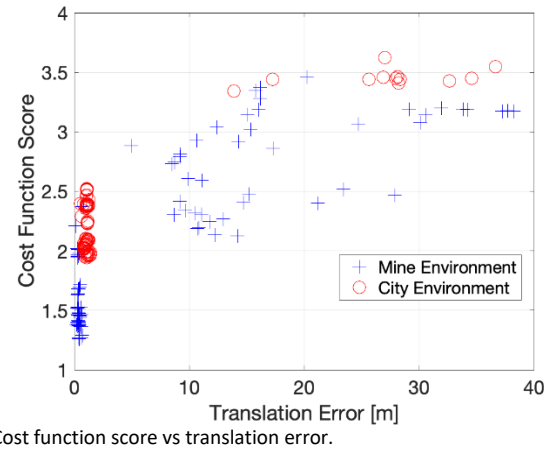
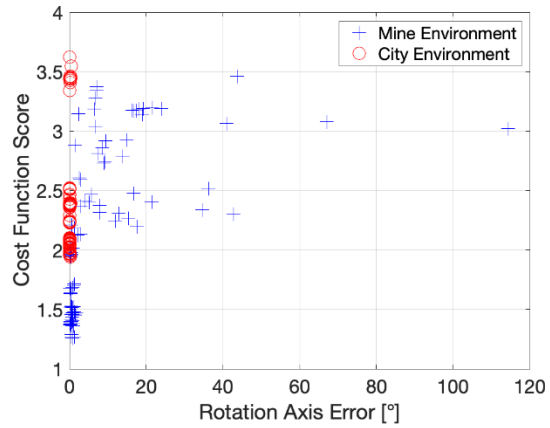


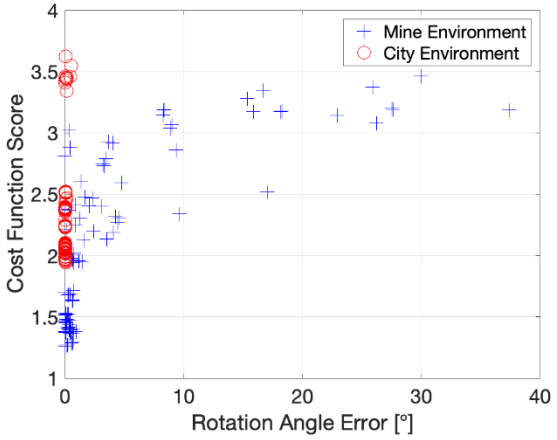
Figure 4. Example of the camera pose detection algorithm flow for the mine environment.



(a) Cost function score vs translation error.

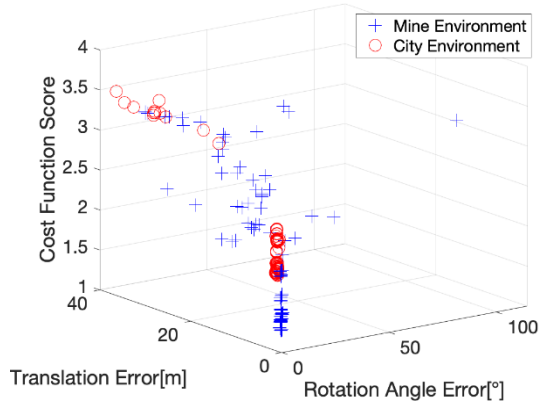


(b) Cost function score vs axis orientation error.

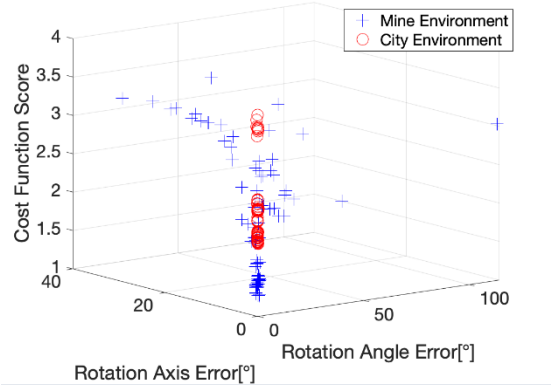


(c) Cost function score vs rotation angle error.

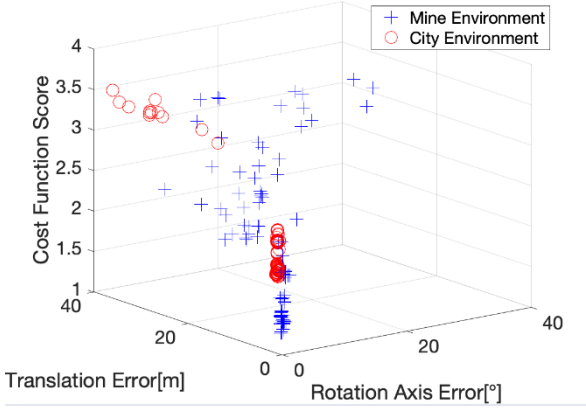
Figure 5. 2D plots of the cost function score vs the errors in translation (a), axis orientation (b), and rotation angle (c).



(a) Cost function score vs translation error vs rotation angle error.



(b) Cost function score vs axis orientation error vs rotation angle error.



(c) Cost function score vs translation error vs axis orientation error.

Figure 6. 3D plots of the cost function score and the errors in translation, rotation angle, and axis orientation.

there is a need for further analysis to investigate possible acceptance criteria and multidimensional models capable of finding a correlation between the different terms of the cost function and the uncertainty in translation and rotation. For example, Figure 7 shows that *MSE* could be a possible discriminant factor for accuracy. Indeed, in this case, the accurate solutions are all centered around the 0.005 value for both exanimated environments.

Once the camera poses were found for each environment, this information is used to set the 360° image projected on the surface of a unitary radius sphere in the correct position and orientation, Figure 8a. After that, using the Raycasting technique, the 3D mesh, Figure 8b, is hit by 360° image pixels (Figure 8c).

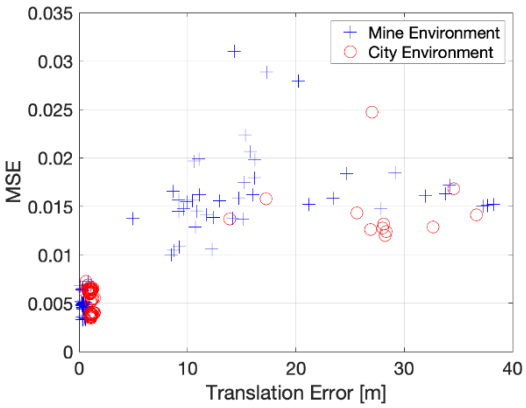
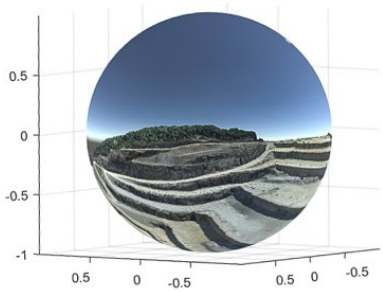
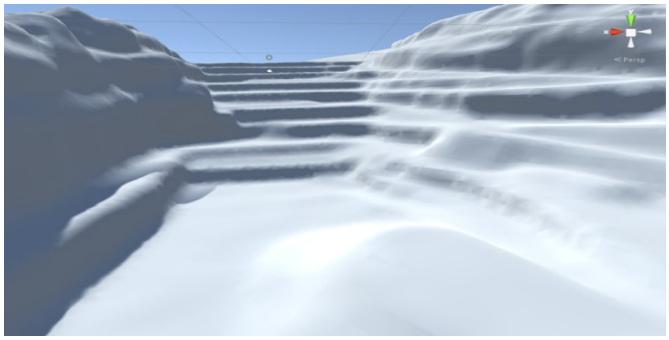


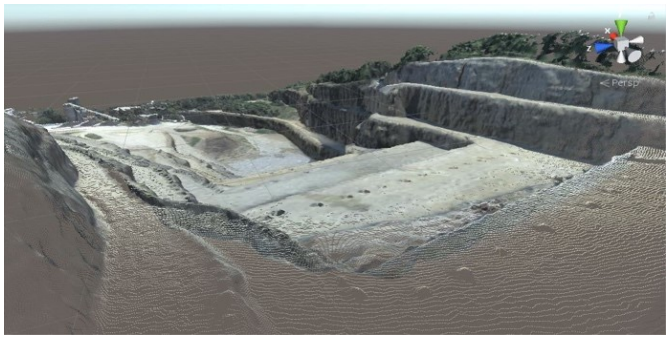
Figure 7. MSE score vs translation error.



(a) 360° image placed on the surface of a unitary sphere (Matlab Software).



(b) Raw 3D mesh (Unity Software).



(c) Point cloud obtained projecting the pixels of the 360° image on the raw 3D mesh (Unity Software).

Figure 8. The pixels of the 360° image of the mine environment are projected on a sphere surface (a), which is put in the correct camera pose found by our algorithm inside the raw 3D mesh (b). The pixels are then projected using the ray cast technique on the raw mesh, obtaining a new dense point cloud (c).

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented an approach for combining photorealistic with 3D environment representations using a 360° high-quality image and a 3D model of an environment with low-

quality. At the core of our system, we have developed an approach for automatic large-scale 360° camera pose estimation within a 3D environment and a method for projective texture mapping spherical images.

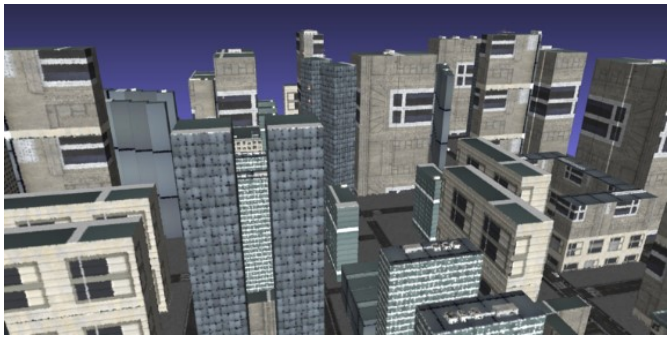
Contrary to previous work, outline in the related work section, the camera pose estimator developed in this paper works both for significant differences in rotation and displacement, and it works without the need to start from a known point of view. The positions and orientations of the camera were estimated with a translation error below 0.7 m, and below 1° and 2° for the difference in the amount of rotation, and the difference in the rotation axis orientation, respectively.

These results were obtained for both environments analyzed at full size and with search limits of ± 20 m for translations and ± 80° for rotations using an *MSE* of 0.005 as a possible discriminant factor for accuracy. While this work was validated using a 360° camera simulation in virtual scenes, we plan to test its capability on real scenes as well. In such situations, the light conditions could be very different between the model and equirectangular image which is why the luminance has to be carefully considered.

Furthermore, the approach here presented is valid until the view of the user rotates without large displacements from the camera's initial position because not all the mesh areas are covered after the pixel projection. To overcome this problem, the same method presented in this paper can be applied with more than one camera, but in the case of the final reconstruction of the texture, there is not a discriminating parameter that allows us to choose which pixels to use from one or another camera for the final reconstruction. This choice can be useful if the field of view of one camera is better for some mesh areas than another one to obtain a better result and it can be implemented in future work.

Finally, in the optimization camera pose process, a further study can be done to find a correlation between the different terms of the cost function and the uncertainty in translation and rotation by investigating other possible acceptance criteria through a multidimensional analysis.

(a)



(b)

Figure 9. Final results after the 3D reconstruction for the mine (a) and the city (b) environments.

REFERENCES

[1] R. Benosman, S. Kang, O. Faugeras, Panoramic vision, Springer Verlag, 2000, ISBN 978-0387951119.

[2] J. Hakulinen, T. Keskinen, V. Mäkelä, S. Saarinen, M. Turunen, Omnidirectional video in museums–authentic, immersive and entertaining, in International Conference on Advances in Computer Entertainment, Springer, 2017, pp. 567–587. DOI: 10.1007/978-3-319-76270-8_39

[3] D. Kalkofen, S. Mori, T. Ladinig, L. Daling, A. Abdelrazeq, M. Ebner, M. Ortega, S. Feiel, S. Gabl, T. Shepel, J. Tibbett, T. H. Laine, M. Hitch, C. Drebenstedt, P. Moser, Tools for Teaching Mining Students in Virtual Reality based on 360° Video Experiences, Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), IEE, Atlanta, GA, USA, 2020, pp. 455-459. DOI: 10.1109/VRW50115.2020.00096

[4] A. MacQuarrie, A. Steed. The effect of transition type in multi-view 360 media, IEEE Transactions on visualization and computer graphics 24(4) (2018), pp. 1564-1573. DOI: 10.1109/TVCG.2018.2793561

[5] M. Tatzgern, R. Grasset, D. Kalkofen, D. Schmalstieg, Transitional augmented reality navigation for live captured scenes, Virtual Reality (VR), IEEE, 2014, pp. 21-26. DOI: 10.1109/VR.2014.6802045

[6] M. Tatzgern, R. Grasset, E. Veas, D. Kalkofen, H. Seichter, D. Schmalstieg, Exploring real world points of interest: Design and evaluation of object-centric exploration techniques for augmented reality. Pervasive and mobile computing 18 (2015), pp. 55-70. DOI: 10.1016/j.pmcj.2014.08.010

[7] J. Thatte, B. Girod, Towards perceptual evaluation of six degrees of freedom virtual reality rendering from stacked omnistereo representation, Electronic Imaging, 2018. DOI: 10.2352/ISSN.2470-1173.2018.05.PMII-352

[8] A. Makadia, K. Daniilidis, Rotation recovery from spherical images without correspondences, IEEE transactions on pattern analysis and machine intelligence 28(7) (2006), pp. 1170–1175. DOI: 10.1109/TPAMI.2006.150

[9] A. Makadia, K. Daniilidis, Direct 3d-rotation estimation from spherical images via a generalized shift theorem, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, Madison, WI, USA, 2003, pp. II–217. DOI: 10.1109/CVPR.2003.1211473

[10] R. Laganiere and F. Kangni, Orientation and pose estimation of panoramic imagery, Mach Graph Vis 19(3) (2010), pp. 339–363.

[11] A. Levin, R. Szeliski, Visual odometry and map correlation, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1 (2004) Washington, DC, USA. DOI: 10.1109/CVPR.2004.1315088

[12] J.-Y. Lai, T.-C. Wu, W. Phothong, D. W. Wang, C.-Y. Liao, J.-Y. Lee, A high-resolution texture mapping technique for 3d textured model, Applied Sciences, vol. 8, no. 11, 2018, p. 2228. DOI: 10.3390/app8112228

[13] T. Abmayr, F. Härtl, M. Mettenleiter, I. Heinz, A. Hildebrand, B. Neumann, C. Fröhlich, Realistic3d reconstruction–combining laserscan data with RGB color information, Proceedings of ISPRS Internation Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 35 (2004), pp. 198–203.

[14] T. Teo, L. Lawrence, G. A. Lee, M. Billinghurst, M. Adcock, Mixed reality remote collaboration combining 360 video and 3D reconstruction, in Proceedings of the 2019 CHI conference on human factors in computing systems, 2019, pp. 1–14. DOI: 10.1145/3290605.3300431

[15] P. Hintjens, ZeroMQ: messaging for many applications. O'Reilly Media, Inc., 2013, ISBN: 9781449334062.
[16] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility tostructural similarity, IEEE Transactions on Image Processing 13(4) (2004), pp. 600–612.
DOI: 10.1109/TIP.2003.819861
[17] M. Kazhdan, H. Hoppe, Screened poisson surface reconstruction, ACM Transactions on Graphics (ToG) 32(3) (2013), pp. 1–13.
DOI: 10.1145/2487228.2487237
[18] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, Meshlab: an open-source mesh processing tool, in Eurographics Italian chapter conference, Salerno, 2008, pp. 129–136.
DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136
[19] Google Inc., Rendering omni-directional stereo content. Online [Accessed 21 March 2022] https://developers.google.com/vr/jump/rendering-ods-content.pdf
[20] D. Girardeau-Montaut, CloudCompare, 2016. Online [Accessed 21 March 2022] https://www.danielgm.net/cc