FROM PHOTOGRAMMETRIC RECONSTRUCTION TO IMMERSIVE VR ENVIRONMENT

Maxime Lhuillier

Université Clermont Auvergne, CNRS, Institut Pascal, F-63000 Clermont Ferrand, France

ABSTRACT

There are several steps to generate a VR environment from images: choose experimental conditions (scene, camera, trajectory, weather), take the images, reconstruct a textured 3D model thanks to a photogrammetry software, and import the 3D model into a game engine. This paper focuses on a post-processing of the photogrammetry step, mostly for outdoor environments that cannot be reconstructed by UAV. Since VR needs a 3D model in a good coordinate system (with a right scale and an axis that is vertical), a simple method is proposed to compute this. In the experiments, we first reconstruct both urban and natural immersive environments by using a helmet-held Gopro Max 360 camera, then import into Unity the 3D models in good coordinate systems, last explore the scenes like a pedestrian thanks to an Oculus Quest.

***Index Terms*—** VR Environment, Vertical Direction, Scale, Photogrammetry, 360 Camera, Hough Transform.

1. INTRODUCTION

Although several photogrammetry software exist today, they produce 3D models that are not directly usable by VR. Indeed VR needs a 3D model in a good coordinate system: the scale must be physically plausible and a coordinate axis must be vertical. A good scale is needed for stereoscopic rendering, realistic speed and motions of the user in the VR environment. Furthermore the vertical direction must be known for drawing as vertical lines in the head-mounted display vertical lines in 3D when the user looks the scene with usual poses of the head (no roll, no pitch). Otherwise, vertical scene components like walls and facades appear to be oblique, and the immersion is unrealistic. Even worse, cybersickness can increase since these visual stimuli are not those of the daily life and since there is a sensor conflict between the vertical seen by the eyes and the vertical detected by the vestibular system [17]. There are other reasons why a photogrammetric reconstruction is not directly usable by VR (e.g. simplify and clean up the mesh), but they are not the topics of the paper.

Unfortunately, a good coordinate system is not guaranteed by standard structure-from-motion included in photogrammetry software, which can only reconstruct a scene up to a similarity transform of the 3D space, if the camera has only one centre like a pinhole camera [9]. (We remind that a similarity transform has 3+3+1 DoF: rotation, translation and scale.) If the camera has several centres, like most 360 cameras composed of several monocular cameras, and if the baseline between the component cameras is known, the scene is theoretically reconstructible up to an Euclidean transform. (Which has 3+3 DoF: rotation and translation.) But the estimated scale can be inaccurate if the baseline is small.

In practice, both scale and vertical are manually estimated thanks to the GUI of a software like Blender [2], but this may be tedious for large environments. Alternatively, the vertical direction can be computed up to orientation (toward sky or ground ?) by assuming that the camera motion is roughly on a horizontal plane during the acquisition of the images. Then the orientation is defined by the user or an additional assumption (e.g. the ground is projected in the bottom of the images) by multiplying the vertical by -1 if needed. Last the 3D model is rotated such that the vertical direction becomes a vector of the canonical basis of $\mathbb{R}^3$. Although this is often OK, there are scenes whose such a vertical estimation is too inaccurate. The paper presents examples: a touristy city build on an inclined ground and a non-planar motion in basalt canyons.

Here we propose a simple but new method to compute both scale and vertical direction (with orientation) from the output of standard photogrammetry assuming that

1. the height of the camera (with respect to the ground surface) during acquisition is roughly constant and known
2. the surface reconstructed by photogrammetry is complete and closed: its triangles cover both ceiling (in a large sense: foliage, sky ...) and ground
3. we know whether the height of the camera is smaller than the difference of heights between the ceiling and the camera (It is for most outdoor scenes.)
4. the scene is anisotropic as follows: the density of the normal of its surface is higher near the horizontal directions and near the vertical direction than elsewhere.

The first assumption is easy to meet in practice for a given image acquisition setting, e.g. for a helmet-held camera by adding an offset to the height of the people who takes the images. The second one requires that the ceiling triangles are not removed by the photogrammetry software, even if they are in the sky. If the images are taken at the ground level with a 360 camera, 3D points are reconstructed all around the camera and the camera is in the convex hull of the 3D points. As a consequence, the surface reconstructed by the standard methods are closed. This also implies that the third assumption is true for most outdoor scenes: the ground-camera distance is smaller than the ceiling-camera distance thanks to high scene components (buildings, trees, ...). For an indoor scene, an estimate of the ceiling height is needed. Last we justify the anisotropy assumption. On the one hand, the ground has large area and its slope is moderated. On the other hand, other scene components have major (almost) vertical areas like facades in urban scenes or trunks in natural scenes. Thus the expected normals are mostly near the vertical direction or near the horizontal directions, respectively.

2. PREVIOUS WORK

Photogrammetry software provides neither scale nor vertical direction, unless its structure-from-motion step integrates measures of an additional sensor (Sec. 2.1). We also compare our work to a previous one that also uses Hough transforms in Sec. 2.2.

2.1. Additional sensor for photogrammetry

GPS and IMU measures can be integrated with both terrestrial [10] and aerial [8] imagery. Bundle adjustment [18] simultaneously refines parameters of both camera poses and 3D points by minimising a sum of reprojection errors, which can be augmented by a GPS term [11] and an IMU term [14]. Bundle adjustment needs a parameter initialisation, which can also be done thanks to GPS [7] and IMU [10]. GPS provides georeferenced reconstruction, which implies that both scale and vertical direction are known. Unfortunately, GPS has unreliable measures in the terrestrial case at locations where GPS satellites are occluded by the scene, e.g. by buildings in urban canyons. IMU measures translation acceleration (including gravity) and rotation acceleration. The vertical can be computed when the IMU does not move, then the scale can be computed by integrating over time (only a few seconds) the IMU measures between two locations of the camera. The method in the paper does not need such additional sensors.

2.2. Hough transform for detecting cylinders

In a different context, [16] estimates cylinders from a cloud of 3D points, that is obtained by scanning an industrial site. This method is based on Hough transforms to deal with outliers and multiple instances. However each cylinder has 5 parameters (2 for the cylinder direction, 2 for the axis position, and 1 for the radius) and a 5D Hough transform is intractable in practice. Thus this method has a two steps: first find potential cylinder directions using a 2D Hough transform, then estimate positions and radii using 3D Hough transforms. Our method also uses a 2D Hough transform but with differences: there is only one direction (the vertical), we don't need a set of circular cylinders in the scene, the input is a triangulated surface, the votes are weighted, the search space of the direction is only a segment of an unit hemisphere, last the vertical and scale estimations are linked.

3. RESET THE COORDINATE SYSTEM

Let $\mathbf{v} \in \mathbb{R}^3$ be the vertical direction pointing toward the sky in the coordinate system used by photogrammetry. First Sec. 3.1 defines a search space for $\mathbf{v}$. Then Sec. 3.2 estimates $\mathbf{v}$ up to its orientation, i.e. it estimates a $\epsilon\mathbf{v}$ where $\epsilon \in \{-1, +1\}$ in the search space. Last Sec. 3.3 finds the good orientation and Sec. 3.4 updates the coordinate system of the 3D model.

3.1. Search space for unoriented vertical

The locations $\mathbf{l}_i \in \mathbb{R}^3$ of the camera during the image acquisition are also known in the coordinate system used by photogrammetry. We first obtain a rough estimate $\mathbf{v}_0$ of $\epsilon\mathbf{v}$ by a principal component analysis (PCA): $\mathbf{v}_0$ is the singular vector with the smallest singular value of the covariance matrix of the $\mathbf{l}_i$. We have $\mathbf{v} \in \{+\mathbf{v}_0, -\mathbf{v}_0\}$ if the camera motion is planar and horizontal. Then the search space of $\epsilon\mathbf{v}$ is a subset of the unit sphere of $\mathbb{R}^3$: the unit vectors $\mathbf{u}$ forming an angle with $\mathbf{v}_0$ that is less than a threshold α. Since we would like to deal with roughly planar ground surfaces with moderate slope angles, a large enough α is needed. Furthermore, $\alpha \leq \pi/2$ since a hemisphere contains all unoriented directions.

3.2. Unoriented Vertical

The vertical direction is detected by a 2D Hough transform. First the search space of $\epsilon\mathbf{v}$ is sampled in a set of vertical candidates. Then every triangle of the surface votes for all vertical candidates that are almost parallel to the triangle. (Triangle normal and vertical candidate are almost orthogonal.) Thus the vertical candidates of a triangle form a segment of a great circle in the unit sphere. Furthermore, the votes of each triangle are weighted by the triangle area, such that the vote result does not change if the triangles are subdivided. Last $\epsilon\mathbf{v}$ is the vertical candidate that maximises the votes.

Here we explain how this method provides the expected result thanks to the anisotropy assumption in Sec. 1. An important area of the surface is formed by roughly vertical triangles. Each vertical triangle votes for a segment of a great circle, and all these great circles intersect at two opposite vectors which have the target vertical direction, whatever the normals of the vertical triangles. (Since the great circle is in a plane that is parallel to that of the triangle.) Thus the votes near the target vertical direction are high. Furthermore, the votes near the horizontal directions are also high since the ground is roughly horizontal and has a large area too. (Each horizontal triangle votes for all horizontal directions). Thus the risk is to obtain a bad $\epsilon\mathbf{v}$ if one of the "horizontal votes" (i.e. votes for roughly horizontal directions) is greater than the vertical votes (similarly). However, the angle between $\mathbf{v}_0$ and every horizontal direction is greater than the angle between $\mathbf{v}_0$ and the vertical, since $\mathbf{v}_0$ is approximately vertical. Now we see that a small enough α rejects the horizontal votes and retains the vertical votes. This implies that such a bad (horizontal) $\epsilon\mathbf{v}$ does not appear. Since the only remaining high votes are vertical, $\epsilon\mathbf{v}$ is vertical.

In the experiments, our trade-off is $\alpha = \pi/4$ to allow a moderate ground slope (Sec. 3.1) and to discard the votes of the ground triangles. We accumulate the votes for vertical candidates corresponding to pixels in an image, such that the vertical candidate of a pixel projects on that pixel by a pinhole camera [9]. The camera is centred at zero like the unit sphere, its field-of-view is 2α, it maps $\mathbf{v}_0$ at the centre of the image. Thus each triangle votes for the pixels along a line segment.

3.3. Oriented Vertical

Now $\epsilon\mathbf{v}$ is known but both ϵ and $\mathbf{v}$ are unknown. Thus there is a function $\epsilon \mapsto \mathbf{v}_\epsilon$. For each possible value of $\epsilon \in \{-1, +1\}$, we examine the triangles that are below the camera trajectory (during the acquisition) in the sense of $\mathbf{v}_\epsilon$. In more details, for each camera location $\mathbf{l}_i$, we collect in a list L_i the few triangle(s) that intersect(s) the half-line hl_i started at $\mathbf{l}_i$ with direction $-\mathbf{v}_\epsilon$. Then we compute a mean m_ϵ, for all i, of the distance between $\mathbf{l}_i$ and the intersection(s) between hl_i and every triangle in L_i. In practice, we reduce such 3D calculations into 2D calculations by rotating both surface and camera trajectory such that the z-axis becomes parallel to $\mathbf{v}_\epsilon$. We also use buckets to accelerate the intersection tests.

Thanks to the second assumption (Sec. 1), m_{-1} and m_{+1} are camera-ground or camera-ceiling distances. According to the third assumption, the camera-ground distance is the smallest one. We obtain $\mathbf{v} = \mathbf{v}_\epsilon$ such that ϵ meets $m_\epsilon < m_{-\epsilon}$.

3.4. Change of the Coordinate System

We rotate the 3D model such that the z-axis has the same direction as $\mathbf{v}$ (using a rotation with axis $\mathbf{r}/||\mathbf{r}||$ and angle $\arcsin(||\mathbf{r}||)$ where $\mathbf{r} = \mathbf{v} \wedge (0 \quad 0 \quad 1)^\top$). We also rescale it by multiplying all vertices by $h/\min\{m_{-1}, m_{+1}\}$ where h is the height of the acquisition camera in meters, thanks to the first assumption.

4. OTHER DETAILS

These details are not the main topic of the paper, but they are interesting for re-implementers and experimenters. Sec. 4.1 describes specificities of our 360 camera. Sec. 4.2 focuses on motion capabilities of the VR user with the help of a motion surface (Sec. 4.3) for general grounds.

4.1. Photogrammetry using the Gopro Max 360 Camera

The Gopro Max is low cumbersome and has a good resolution (Details are below.) with a moderated price. It is composed of two fisheye cameras that point in opposite directions. However it uses a special image format [15] to improve the compression efficiency: it compresses neither equirectangular nor original fisheye images[1] as the other 360 cameras, but compresses an equiangular cubemap image with redundancy between borders of cube faces (Fig. 1). The firmware

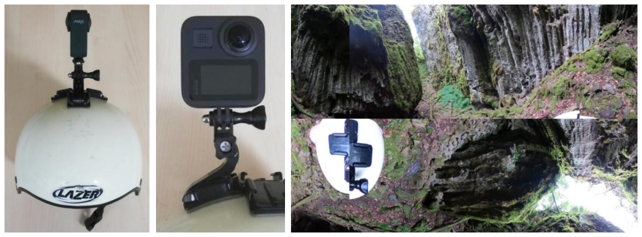


Fig. 1. A Gopro Max 360 camera and a cubemap image.

of the camera stitches each pair of original fisheye images to an equiangular cubemap image using an approximate knowledge of the Gopro Max calibration (with a single camera centre: that of the cube). Then we have to convert each equiangular cubemap image in a format supported by photogrammetry software (The list is non-exhaustive here.): an equirectangular image e.g. by [5], or a traditional cubemap image e.g. by [1], or several fisheye images using a classical projection model with radial distortions e.g. by [4]. Thus the photogrammetry input and output can be degraded.

In practice, the equiangular cubemap images are obtained from the files (with suffix ".360") provided by the Gopro Max as follow. Each file is limited to 4Go (8 minutes) and contains two synchronised 4096×1344 videos at 30Hz. A cubemap image is a concatenation of two synchronised images, each of them has three faces of the cube. We first extract two MP4 files from the input files with concatenation over time by using FFmpeg [3] only twice without loss of quality:

```
ffmpeg -f concat -i list.txt -map 0:0 -c copy cube1.MP4
ffmpeg -f concat -i list.txt -map 0:5 -c copy cube2.MP4    (1)
```

where the file list.txt includes lines such as: file 'GS010050.360'. Then the successive cubemap images are extracted by decompressing simultaneously the two MP4 files.

4.2. Motion Capabilities of the User

The goal is visualisation by VR of immersive 3D models reconstructed by photogrammetry, which can be large in the sense of the trajectory length of the acquisition camera: from several hundreds to a few kilometres in our experiments. Furthermore, we would like that the user of VR

[1]There is also a file for the original fisheye images with a reduced resolution and very high compression rate, but it is unusable by photogrammetry.

1. freely moves on the ground like a pedestrian
2. can follow the trajectory of the acquisition camera for a naive exploration, where the visual quality is good
3. has strong spatial awareness and weak cybersickness.

We would like to meet (1) with a ground that is neither horizontal nor planar. This is done by enforcing the locations of the user eyes to be in a "motion surface" that approximates an offset of the ground surface. Thanks to the first assumption in Sec. 1, we extrapolate the motion surface from the trajectory of the acquisition camera (more details in Sec. 4.3).

For (2), we augment the 3D model by a small cube with gray levels at every acquisition location so that the user of VR can look and follow the acquisition trajectory without spending time to find where are the good reconstructed parts of the scene. The user can also visually assess the quality of the photogrammetry result by moving away from the acquisition trajectory. The greater the distance between the user and this trajectory, the greater the visual artifacts due to photogrammetry inaccuracies or incompleteness.

Last we should meet (3) and choose a motion method for large virtual environments [19] among steering and teleportations. Unfortunately, the two conditions in (3) seem to be contradictory and a trade-off must be done. In short, both spatial awareness and cybersickness are usually greater for a continuous motion (steering) than for a discrete motion (teleportation). The best trade-off is outside the scope of this paper and we only implement the following capabilities inspired by VR games: a continuous motion with the left joystick and a discrete rotation with the right joystick (add $\pm\pi/12$ to the yaw angle). The former is also useful for users who want to assess the photogrammetry result thanks to the motion parallax. The latter resets a mean for the next directions of the exploration in progress. The user remains seated most of time and can also rotate the head as usual.

4.3. Motion Surface

The motion surface is extrapolated from the trajectory of the acquisition camera computed by photogrammetry as follows. Assume that the successive locations (x_i, y_i, z_i) of the acquisition camera are in the coordinate system used by the game engine for the 3D model. In the case of Unity [6] which we use, the y-axis is vertical and points toward the sky. Furthermore, the ground is roughly horizontal. Thus we define the motion surface as a function $y = f(x, z)$, then the left joystick updates the values of x and z, which moves the virtual camera on the motion surface thanks to f. Let

$$f(x,z) = a + \frac{\sum_i w_{ci} y_{ci}}{\sum_i w_{ci}} \text{ where } w_i = e^{-\frac{\sqrt{(x-x_i)^2+(z-z_i)^2}}{b}}, \quad (2)$$

a is a vertical offset of the virtual camera with respect to the acquisition camera, b is a scale factor to set the decrease of the weights w_i with respect to the distance between (x, z) and (x_i, z_i), and c is used to skip locations and accelerate the calculations of f at each frame (assuming that i increases by acquisition time). Both a and b are in meters. The value of b is a trade-off: small enough for extrapolation accuracy, large enough to avoid staircase effects. A non-zero a is useful to avoid cubes that occlude the field-of-view of the user of VR.

We finish by a note about the coordinate systems. Ours is right-handed (as usual) with a vertical z-axis toward sky. That of Unity is left-handed with a vertical y-axis toward sky. There are two consequences. First, we have to convert the table of locations (x_{ci}, y_{ci}, z_{ci}) in the C# program needed by Unity that computes f: there is a $-\pi/2$ rotation around the x-axis and a sign change of the x-coordinate to switch from right-handed to left-handed. Thus we convert (x_{ci}, y_{ci}, z_{ci}) to $(-x_{ci}, z_{ci}, -y_{ci})$. Second, we set Unity to rotate by $-\pi/2$ around the x-axis while importing our 3D models. (The sign change of x is implicitly done here). Then the acquisition locations and 3D models are consistent in Unity.

5. EXPERIMENTS

5.1. Dataset

The first 3D model "Basalt" is reconstructed from a 360 video taken by walking during 27 minutes in a geologic site using one Gopro Max camera mounted on the top of a helmet. The two sides of the path is composed of basalt prisms, whose heights are in range 5-20m, and which are degraded by vegetation and high humidity. The texture is favourable for photogrammetry, except at a few locations with low light due to narrow canyon (less than 1m) or where the vegetation is too close to the camera (less than 50cm). The path is mostly composed of rocks, that can be slippery and are bordered by small vegetation. Its slope angle is less than $\pi/4$, except at a few small sections where walking is replaced by climbing with hands or by going down on the buttocks. This path was carefully selected before the acquisition since the site is complicated (a kind of labyrinth with obstacles and dangerous sections) and since the trajectory must include closed loops so that the structure-from-motion step reduces the drift.

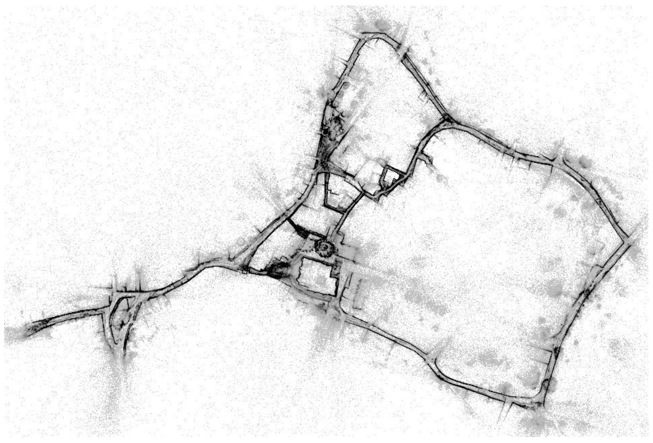
The second 3D model "City" is reconstructed from a 360 video taken by walking during 21 minutes in roads of a medieval city using two Gopro Max cameras, that are rigidly mounted on both sides of the head near the ears (only one fisheye is used by each Gopro). This model is more standard than the first one, but it is also interesting for both VR and experimenting our method on a roughly planar ground with a moderated slope. It includes small castles, a church, trees and medieval porches. The path is chosen before acquisition thanks to a map, so that it includes several closed loops. Both day and hour of acquisition are chosen for reducing the number of tourists in the roads, which are prohibited to most cars. As in the first model, the weather is chosen so that there is

Name	#tri	#loc	h	length	slope-length
Basalt	2.9M	3840	1.8m	819m	>30%,496m
City	3.4M	3132	1.55m	1.6km	>10%,240m

Table 1. Dataset's characteristics: number of triangles, number of selected camera locations $\mathbf{l}_i$, camera-ground distance (i.e. camera height) h, trajectory length, slope-length (e.g. at least 10% slope along 240m of the acquisition trajectory).

enough light and the vegetation does not deform.

Tab. 1 shows characteristics of Basalt and City, including trajectory lengths and slope angles which are estimated thanks to our 3D coordinate system change. Note that these environments are not trivial: they are several hundred meters long with millions of triangles (reconstructed using [12] and [13]) and non-negligible slope angles. Fig. 2 shows global and top views of reconstructed points and camera trajectories.


(a) City's points

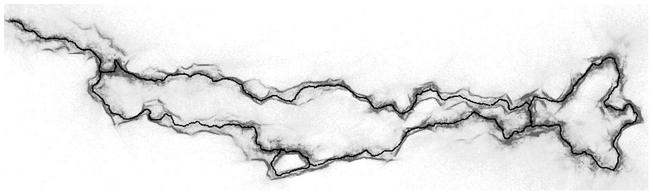
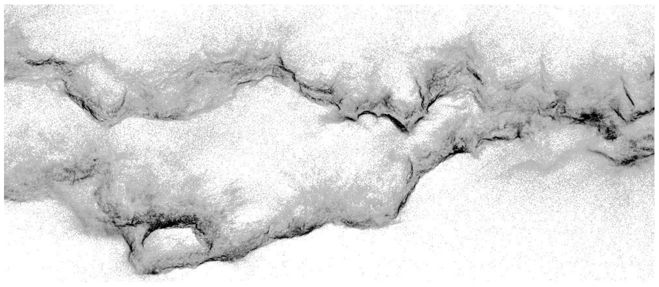

(b) Basalt's points and camera trajectory

Fig. 2. Global and top views of the reconstructed points of City and Basalt. The darker the gray level, the greater the point density. The trajectory of the acquisition camera is also shown in black for Basalt.

5.2. Comparisons

We first compare methods that compute the vertical direction up to its orientation: the PCA method in Sec. 3.1 and the Hough method in Sec. 3.2. (The former initialises the latter.) Their estimates are $\mathbf{v}_0$ and $\epsilon\mathbf{v}$, respectively. The comparison is done by orthogonal projections of the reconstructed cloud of points such that the projection direction is $\mathbf{v}_0$ or $\epsilon\mathbf{v}$, and by examining the scene components that have vertical surfaces. These components must be projected into lines or curves. Fig. 3 shows the projections for Basalt, which is composed of corridors (or canyons) whose sides are mostly vertical. The corridor sides are more parallel to the direction


(a) Projection parallel to the PCA direction ($\mathbf{v}_0$)

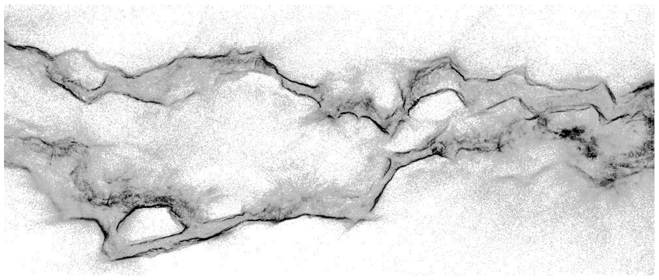
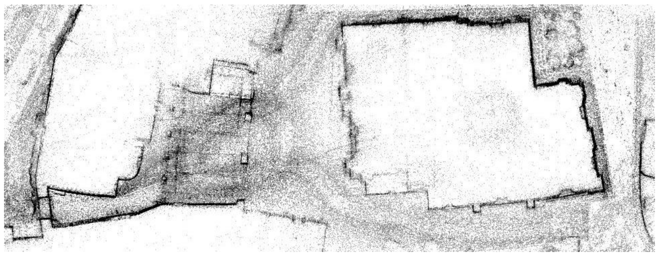

(b) Projection parallel to the Hough direction ($\epsilon\mathbf{v}$)

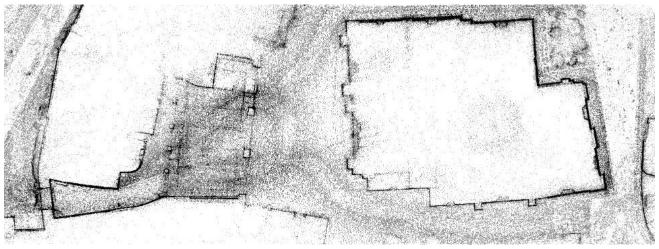
Fig. 3. Top views of reconstructed points of Basalt. The canyon sides are more parallel to the direction estimated by the Hough transform than those estimated by PCA.

estimated by the Hough transform than those estimated by PCA, since new dark curves appear with $\epsilon\mathbf{v}$. The comparison is less easy for City in Fig. 4 because the line segments of $\mathbf{v}_0$ and $\epsilon\mathbf{v}$ look the same. A more careful observation is needed, e.g. for orthogonal walls (left and bottom) of the main building in the figure: the line segments are less noisy and darker for $\epsilon\mathbf{v}$ than for $\mathbf{v}_0$. Furthermore, there are other line segments (or curves for Basalt) that have a smaller thickness for $\epsilon\mathbf{v}$ than for $\mathbf{v}_0$. The smaller the thickness, the more accurate the vertical. Thus the conclusion is the same for City and Basalt: $\epsilon\mathbf{v}$ is better than $\mathbf{v}_0$. The angle between $\mathbf{v}_0$ and $\epsilon\mathbf{v}$ is equal to 18 degrees for Basalt and 4 degrees for City. This explains why this comparison is easier for Basalt than for City.

Second there is another comparison to show that $\epsilon\mathbf{v}$ is better than $\mathbf{v}_0$. In Fig. 5, each image of the City's 3D model is drawn by a pinhole camera [9] that has zero pitch and zero roll angles with respect to a vertical direction of reference among $\mathbf{v}_0$ and $\epsilon\mathbf{v}$. Then the vertical lines in 3D are projected to vertical lines in 2D if the direction of reference is accurate. They are for $\epsilon\mathbf{v}$ but they are not for $\mathbf{v}_0$. (The images can be rotated



(a) Projection parallel to the PCA direction ($\mathbf{v}_0$)



(b) Projection parallel to the Hough direction ($\epsilon\mathbf{v}$)

Fig. 4. Top views of reconstructed points of City. Walls and facades are more parallel to the direction estimated by the Hough transform than those estimated by PCA.

by π to see the sky on the top.)

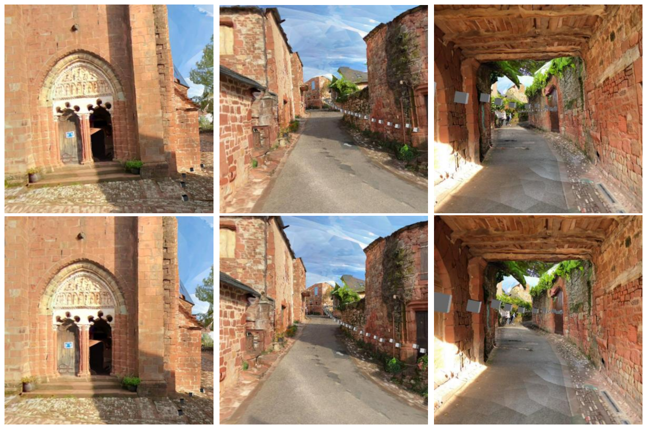
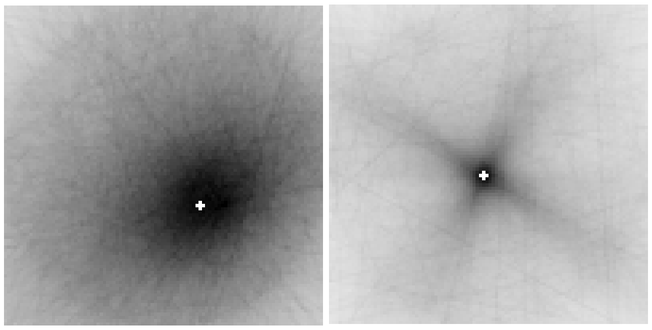


Fig. 5. City's views with zero pitch and zero roll. The vertical lines in 3D are projected to vertical lines in 2D if the reference vertical direction is $\epsilon\mathbf{v}$ (bottom). They are not for $\mathbf{v}_0$ (top).

Third we discuss the accumulator spaces of Basalt and City thanks to Fig. 6. We remind that they are used by the Hough transform to count the votes for the vertical candidates. The dark spots look like uncertainty ellipsoids and suggest that the computation of $\epsilon\mathbf{v}$ is more reliable in City than in Basalt. This confirms the observation that the basalt walls are less vertical than the building walls. The resolution of the accumulator spaces is a trade-off: not too large due to noise, not too small for accuracy. Here we use 100×100 images for angle ranges $[-\pi/4, +\pi/4]^2$, i.e. there is about 0.9 pixel per degree. We also see that the City accumulator is more anisotropic than the Basalt one: the former has two slightly dark strips that are roughly orthogonal. This can be explained as follow: the roads shown in Fig. 2 have roughly two main orthogonal directions, then most wall/facade surfaces (which are parallel or orthogonal to their roads) have two orthogonal normals whose great circles vote in the strips.

Name	m_{-1}	σ_{-1}	m_{+1}	σ_{+1}	σ_{+1}/m_{+1}
Basalt	28.2	15.1	5.6	0.99	0.177
City	24.7	11.0	7.71	0.23	0.0298

Table 2. Mean and standard deviations of ground-camera and ceiling-camera distances in the photogrammetric coordinate system. The ground-camera distance is m_{+1}.



(a) Basalt's accumulator (b) City's accumulator

Fig. 6. Accumulator spaces ($[-\pi/4, +\pi/4]^2$) of the Hough transform. The darker the gray level, the greater the vote. We add white crosses to show $\epsilon\mathbf{v}$. The centre of the space is $\mathbf{v}_0$.

Fourth we use Tab. 2 to detail the estimations of the orientation and scale in Sec. 3.3. We remind that we distinguish the camera-ground and the camera-ceiling distances in $\{m_{-1}, m_{+1}\}$ by comparing m_{-1} and m_{+1}. Since $m_{+1} < m_{-1}$ for both City and Basalt and thanks to the third assumption (Sec. 1), m_{+1} is the camera-ground distance and the 3D models will be rescaled such that m_{+1} becomes equal to h. We also provide the standard deviation σ_ϵ associated to m_ϵ for discussion. Since $\sigma_{+1} < \sigma_{-1}$, σ_ϵ can replace m_ϵ for distinguishing ground and ceiling. Furthermore, σ_{+1}/m_{+1} is smaller for City than for Basalt. Thus the assumption of constant camera height (Sec. 1) is better for City than for Basalt. This confirms the observation: the people who takes the images in the City case is always standing, while in the Basalt case he sometimes walks squatting to pass under obstacles (e.g. tree branches) or goes down on the buttocks.

Fifth, Figs. 7 and 8 show screenshots taken by a standalone Oculus Quest during explorations (Sec. 4.2) of Basalt and City, after our reset of their coordinate systems (Sec. 3.4). The outward-pointing normal direction is also encoded by colours: white for vertical toward sky or black toward ground, red-green-blue for horizontal. The reader can check that the ground of Basalt is not horizontal thanks to normal colours and cubes of the trajectory. For City, this can be seen in the last row thanks to horizontal levels of bricks with different colours. The setting of the user of VR is ($a = +0.3$, $b = 1$, $c = 4$) for Basalt and ($a = -0.2$, $b = 4$ and $c = 10$) for City.

VR applications for immersive explorations are available from `http://maxime.lhuillier.free.fr` in the section "Photogrammetry for VR".

5.3. Limitations of the method

A surface with a single normal direction, i.e. segment(s) of parallel planes, votes evenly for all vertical candidates along a single great circle of the unit sphere. Thus the unoriented vertical cannot be estimated as the maximiser of these votes. A piecewise planar reconstructed scene must have at least two segments of vertical planes with distinct unoriented normals. In practice, we must avoid a small acquisition trajectory that only reconstructs a single vertical plane with the ground. We should also avoid a trajectory limited to a straight road if its walls and facades are mostly parallel to it.

6. CONCLUSION

This paper estimates the vertical direction and the scale of a triangulated surface reconstructed by photogrammetry from terrestrial imagery, with explicit assumptions and limitations. The method is simple but new: first compute the unoriented vertical by a principal component analysis of the acquisition trajectory followed by a 2D Hough transform, then obtain the oriented vertical direction and the scale by projecting the trajectory onto the surface with respect to the two possible opposite directions. The paper also provides details on the acquisition and visualisation of two immersive VR environments (obtained by photogrammetry and the proposed method): a basalt canyon and an urban canyon with trajectory lengths of about 800m-1.6km. Future work includes comparisons of the results with those obtained by using other sensors like GPS and IMU, and the improvement of the travel method to reduce the cybersickness without sacrificing the spatial awareness.

7. REFERENCES

[1] 3df zephyr. https://www.3dflow.net.

[2] Blender. https://www.blender.org.

[3] Ffmpeg. https://ffmpeg.org.

[4] Meshroom. https://alicevision.org.

[5] Metashape. https://www.agisoft.com.

[6] Unity. https://unity.com.

[7] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *Conference on Computer Vision and Pattern Recognition*. IEEE, 2011.

[8] B. Grayson, N.T. Penna, J.P. Mills, and D.S. Grant. Gps precise point positioning for uav photogrammetry. *The Photogrammetric Record*, 33(164):427–447, 2018.

[9] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision (chapter: 3D reconstruction of cameras and structure)*. Cambridge University Press, 2000.

[10] B. Klingner, D. Martin, and J. Roseborough. Street view motion-from-structure-from-motion. In *International Conference on Computer Vision*. Computer Vision Foundation, 2013.

[11] M. Lhuillier. Fusion of gps and structure-from-motion using constrained bundle adjustments. In *Conference on Computer Vision and Pattern Recognition*. IEEE, 2011.

[12] M. Lhuillier. Surface reconstruction from a sparse point cloud by enforcing visibility consistency and topology constraints. *Computer Vision and Image Understanding*, 175, 2018.

[13] M. Lhuillier. Local-convexity reinforcement for scene reconstruction from sparse point clouds. In *International Conference on 3D immersion*, 2019.

[14] J. Michot, A. Bartoli, and F. Gaspard. Bi-objective bundle adjustment with application to multi-sensor slam. In *International Symposium on 3D Data Processing, Visualization, and Transmission*, 2010.

[15] D. Newman and D. Stimm. This is gopro max: Tech, specs + more, 2019. https://gopro.com/en/us/news/max-tech-specs-stitching-resolution.

[16] T. Rabbani and F. Van Den Heuvel. Efficient hough transform for automatic detection of cylinders in point clouds. In *Laser Scanning*. ISPRS, 2005.

[17] N. Tian, R. Clement, P. Lopes, and Boulic R. On the effect of the vertical axis alignment on cybersicknes and game experience in a supine posture. In *IEEE Conference on Games*. IEEE, 2020.

[18] B. Triggs, P.F. McLauchlan, R.I. Hartley, and A.W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Vision Algorithms: Theory and Practice*, pages 298–372. Springer-Verlag, 1999.

[19] T. Weissker, A. Kunert, B. Frohlich, and A. Kulik. Spatial updating and simulator sickness during steering and jumping in immersive virtual environments. In *IEEE Conference on Virtual Reality and 3D User Interfaces*. IEEE, 2018.

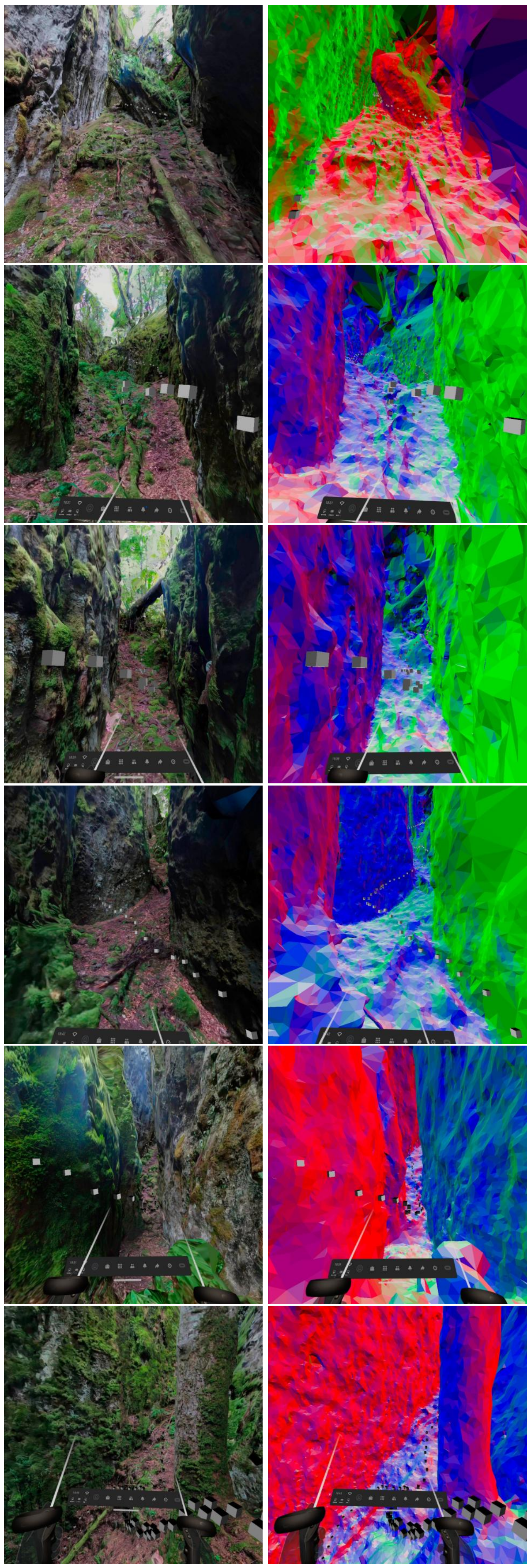


Fig. 7. Basalt's screenshots by a standalone Oculus Quest.

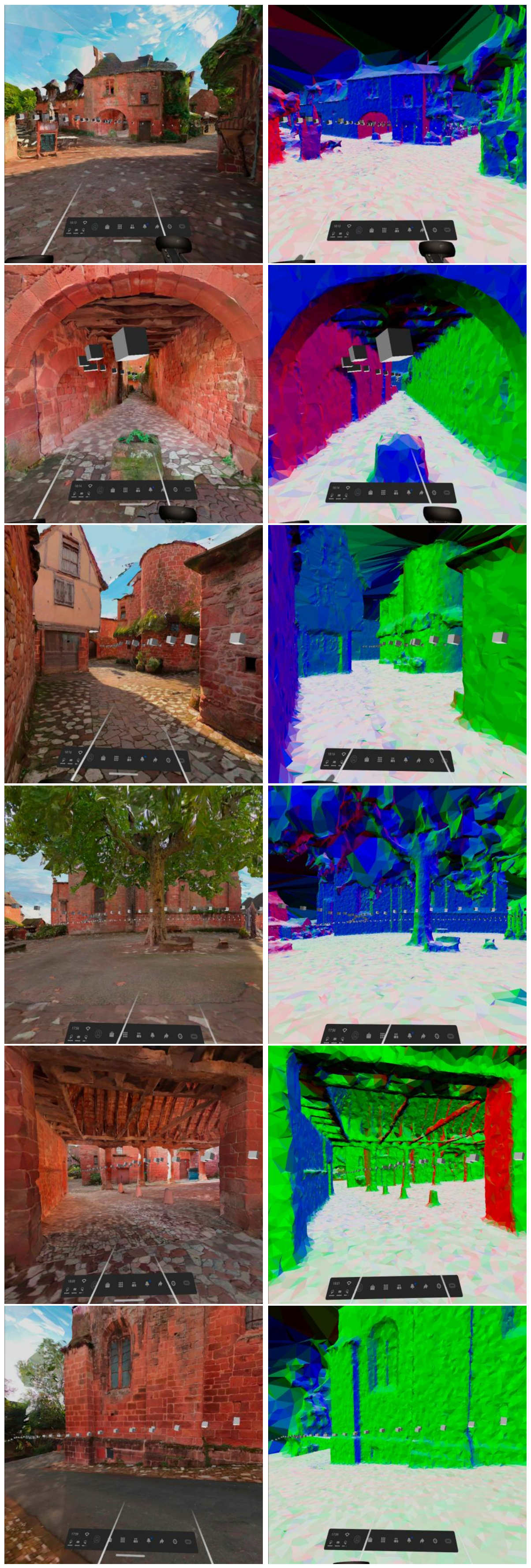


Fig. 8. City's screenshots by a standalone Oculus Quest.